한국정보과학회
KOREAN INSTITUTE OF INFORMATION SCIENTISTS AND ENGINEERS

제 22 권 제 1 호
Vol. 22 No. 1

SOFTWARE ENGINEERING SOCIETY

KIPS 소프트웨어공학소사이어티

2020

제 22 회 한국 소프트웨어공학 학술대회 논문집

Proceedings of the 22nd Korea Conference on Software Engineering (KCSE 2020)

- 일시: 2020년 2월 3일(월) ~ 2월 5일(수)
- 장소: 강원도 평창 한화리조트(휘닉스파크점)

주최: 한국정보과학회, 한국정보처리학회
주관: 한국정보과학회 소프트웨어공학 소사이어티
한국정보처리학회 소프트웨어공학연구회

후원: ETRI 한국전자통신연구원 Electronics and Telecommunications Research Institute, SOLUTIONLINK ㈜비트컴퓨터, ㈜모아소프트, ㈜이에스지, 한국소프트웨어기술진흥협회(KOSTA), T3Q㈜, ㈜다한테크, 슈어소프트테크㈜, ㈜온페이스, STA 테스팅컨설팅㈜, TTA 소프트웨어시험인증연구소, ㈜에스피아이디, 신뢰적지능형 CPS 연구단

초대의 글

소프트웨어공학인의 축제인 제 22 회 한국 소프트웨어공학 학술대회(KCSE 2020) 참가자 여러분을 환영합니다.

기업, 연구소 및 학계에서 활동하고 계신 소프트웨어공학 분야 전문가들의 모임인 한국정보과학회 소프트웨어공학 소사이어티와 한국정보처리학회 소프트웨어공학연구회는 소프트웨어공학 기술의 발전 및 적용확산을 위하여 산, 학, 연과의 협력으로 한국 소프트웨어공학 학술대회를 개최하게 되었습니다.

최근 인공지능(AI)이 널리 확산되어 스스로 사람처럼 학습하고 판단하여 그 결과를 빠르고 정확하게 보여주어 세상을 변화시키는 중요한 기술과 방법으로 인식되고 있습니다. 이러한 인공지능의 핵심은 소프트웨어 기술이므로 KCSE 2020 학술대회에서는 "AI 를 완성하는 소프트웨어공학 기술"을 주제로 하여 소프트웨어공학 분야의 각계에서 제출한 81 편의 논문이 발표됩니다.

먼저, 스마트 SW 개발, 인공지능과 인간의 공존, 그리고 포스트 휴머니즘, 데이터 중심 혁신을 통한 AI 응용 구축 등 소프트웨어공학의 중요한 이슈와 미래방향에 대하여 세 분의 저명인사께서 기조강연을 해 주십니다. 또한 ADOxx Meta-Modeling Platform, Concolic 유닛 테스팅, SW 안전 확보를 위한 기법, 블록체인, 의미 기반 프로그램 수정 등의 최근 기술 동향과 소프트웨어공학의 핵심기술을 학습할 수 있는 튜토리얼이 준비되어 있습니다. 특히, 우수국제학술대회 논문 3 편, 우수국제저널 논문 2 편, 소프트웨어공학 소사이어티 논문지 논문 2 편을 초청하여 프로그램을 구성했습니다. 그리고 소프트웨어 테스트 환경 순위화, 자동 패치 생성, 프로그램 분석을 위한 휴리스틱 학습 등의 최근 연구 주제에 대하여 세 분의 박사께서 세미나를 해 주십니다.

본 학술대회가 소프트웨어공학의 학문적 발전과 소프트웨어 산업기술 발전의 장이 되고, 학술 교류 및 기술 협력을 위한 활발한 토론장이 될 수 있도록 여러분의 적극적인 참여를 부탁드립니다.

본 KCSE 2020 행사를 위해 수고해 주신 조직위원회와 학술위원회 위원들과 여러 후원 기관에 깊이 감사드립니다.

한국정보과학회 소프트웨어공학 소사이어티 회장 이병정
한국정보처리학회 소프트웨어공학연구회 운영위원장 김정아

학술대회 준비 위원회

공동대회장: 이병정 교수(서울시립대), 김정아 교수(가톨릭관동대)

조직위원장: 김정아 교수(가톨릭관동대)
조직위원: 홍장의 교수(충북대), 백종문 교수(KAIST), 이정원 교수(아주대),
고인영 교수(KAIST), 김순태 교수(전북대), 한종대 교수 (상명대),
이선아 교수(경상대),
정효택 박사(ETRI), 민상윤 대표(솔루션링크), 전진옥 사장(비트컴퓨터),
이해서 대표(이에스지), 박병훈 대표(티쓰리큐), 장주수 대표(모아소프트)

학술위원장 유준범 교수(건국대)
학술위원: 김문주 교수(KAIST), 김영철 교수(홍익대), 남재창 교수(한동대),
류덕산 교수(전북대), 박수진 교수(서강대), 배경민 교수(포항공대),
서영석 교수(영남대), 서주영 교수(아주대), 이관우 교수(한성대),
이우진 교수(경북대), 이지현 교수(전북대), 이찬근 교수(중앙대),
윤회진 교수(협성대), 염근혁 교수(부산대), 지은경 교수(KAIST),
최윤자 교수(경북대), 채흥석 교수(부산대), 홍신 교수(한동대)

문의사항 연락처

학술대회 홈페이지 : http://www.sigsoft.or.kr/KCSE2020/
조 직 : 김정아 교수 (clara@cku.ac.kr, 033-649-7801)
학 술 : 유준범 교수 (jbyoo@konkuk.ac.kr, 02-450-3258)

KCSE 2020 프로그램

2 월 3 일 (월)

시간	행 사 내 용				
12:00–13:00	**KCSE 2020 등록**				
	튜토리얼 T1 좌장: 지은경(KAIST) 장소: 세미나실 1	**튜토리얼 T2** 좌장: 이찬근(중앙대) 장소: 세미나실 2	**튜토리얼 T3** 좌장: 한종대(상명대) 장소: 세미나실 3	**튜토리얼 T4** 좌장: 이정원(아주대) 장소: 그랜드홀 2	**워크숍: 신뢰적 지능형 CPS 연구단** 장소: 세미나실 4 (09:00-16:00)
13:00-14:30 (90 분)	ADOxx Meta-Modeling Platform 이문근 교수 (전북대)	Concolic 유닛 테스팅 김윤호 교수 (KAIST)	전산학 논문 작성법 (Technical Writing in English) 차성덕 교수 (고려대)	AI 시스템의 품질 특성 및 검증 방법 최영재 팀장 (STA 테스팅 컨설팅)	
14:30-14:40	**휴식**				
	튜토리얼 T5 좌장: 이찬근(중앙대) 장소: 세미나실 1	**튜토리얼 T6** 좌장: 김영철(홍익대) 장소: 세미나실 2	**튜토리얼 T7** 좌장: 이정원(아주대) 장소: 세미나실 3		
14:40-16:10 (90 분)	SW 안전 확보를 위한 기법: 도메인 적용 방안 및 사례 지은경 교수 (KAIST)	블록체인의 핵심 개념, 산업별 응용 사례 그리고 소프트웨어공학 연구 이슈 소개 김순태 교수 (전북대)	의미 기반 프로그램 수정 (Semantics-based Program Repair) 이주용 교수 (UNIST)		
16:10-16:20	**휴식**				
	개회식 장소: 그랜드홀 2	사회: 김정아 조직위원장(가톨릭관동대)			
16:20-16:40 (20 분)	개회사: 이병정 회장 (한국정보과학회 소프트웨어공학 소사이어티) 김정아 운영위원장 (정보처리학회 소프트웨어공학연구회)				
	기조강연 I 장소: 그랜드홀 2	사회: 김정아 조직위원장(가톨릭관동대)			
16:40-17:30	Smart S/W Development : AI for Software Engineering 김강태 상무 (삼성전자)				
	신진 연구자 초청 발표 I 좌장: 지은경(KAIST) 장소: 세미나실 1	**신진 연구자 초청 발표 II** 좌장: 홍장의(충북대) 장소: 세미나실 2	**신진 연구자 초청 발표 III** 좌장: 백종문(KAIST) 장소: 세미나실 3		
17:30-18:20	Prioritizing test environments for cost-effective software test in continuous integration 권정현 박사 (KT)	Automatic Patch Generation with Context-based Change Application 김진대 박사 (홍콩과기대 졸업)	Learning Heuristics for Fast and Precise Java Points-to Analysis 정세훈 박사 (고려대 졸업)		
18:20-19:10	**석 식**				

2 월 4 일 (화)

시 간	행 사 내 용			
	논문 발표 A			
	A1: SW 프로세스 좌장: 이선아(경상대) 장소: 세미나실 1	**A2: SW 모델링 및 설계** 좌장: 배경민(포항공대) 장소: 세미나실 2	**A3: 요구사항 및 보안** 좌장: 김순태(전북대) 장소: 세미나실 3	**A4: 인공지능 I (SE for AI)** 좌장: 류덕산(전북대) 장소: 그랜드홀 2
09:20-11:00 (100 분)	인공지능 의료기기 소프트웨어의 표준 준수를 위한 개발자 관점 품질관리 프로세스 정의 **[최우수 일반논문]** 김동엽, 박예슬(아주대), 이병정(서울시립대), 이정원(아주대) 시장의 리드 타임에 충족하는 속도와 품질 확보를 위한 코드분석 기반의 배포 프로세스 수립과 적용사례 [산업체 논문] 안선희, 김진태(소프트웨어공학엑스퍼트그룹) SW 개발자 커뮤니티를 통한 개발 문화 혁신 사례 [산업체 논문] 김상기, 유광엽, 주석원(SKT) 국제표준 개정에 따른 국방 소프트웨어 개발절차 개선방향 [산업체 논문] 박태현, 심승배, 김의순, 조성림, 홍수민, 윤웅직(한국국방연구원) OpenSource 와 Web(IDE) Plug-In 중심으로 ALM 시스템 구축 [산업체 논문] 전인복(Kt ds), 백종문(KAIST)	멀티 응용의 실시간 성능 분석을 위한 스케줄 기반 SDF (Synchronous Dataflow) 모델 변환 기법 [일반논문] 정도환, 김장률, 하순회(서울대) 데이터 플로우 모델 기반 임베디드 소프트웨어 플랫폼에서의 공유 정보 관리 기술 [일반논문] 강우석, 홍혜선, 정은진, 마리리스 올드자, 하순회(서울대) 지능형 군집 무인체계의 개발을 위한 모델링&시뮬레이션 방법론 [일반논문] 김희수(리얼타임비쥬얼), 성영화(국방과학연구소) 가상화 환경 기반의 무기체계 소프트웨어 규격화 품질향상 방안 **[우수 산업체논문]** 최민관, 국승학, 이태호(국방과학연구소) 군집개체 강화학습을 위한 시뮬레이션 환경 기술동향 [산업체 논문] 배정호, 김성호, 김용덕, 성영화(국방과학연구소)	국방 SPL 프레임워크 기반의 무기체계 항법 SW 플랫폼 요구사항 분석 [산업체 논문] 노성규, 박병수, 남성호, 성창기, 백승준, 박삼준(국방과학연구소) Triplet 추출 기반 범용 온톨로지 구축 설계 [단편논문] 김아령, 이상백, 이규철(충남대) 준 지도 학습을 활용한 네트워크 패킷에서의 이상검출 [단편논문] 어준선, 장기영, 지효상, 양성봉(연세대) 비밀번호 복구를 위한 사진 문맥 데이터 기반 스마트폰 사용자 인증 접근법 [단편논문] 송성한, 김순태, 김태영(전북대) PVFS 의 보안성 향상을 위한 Temporal Logic 기반의 허위 보고서 판단 방법 [단편논문] 안정섭, 조대호(성균관대) 무인이동체 소프트웨어 요구사항 추적 기준 연구 [단편논문] 장정훈(모아소프트)	커버러지 방법을 이용한 심층 신경망 구조 최적화 **[우수 일반논문]** 이민수, 이찬근(중앙대) ResNet 의 뉴런 활성 값과 Epoch 간 상관관계 분석을 통한 화이트 박스 방법의 모델 평가 [일반논문] 박재현, 최현재, 채흥석(부산대) 캡슐내시경 영상 데이터의 학습 성능 강화를 위한 색 공간 군집 기반 특성 편향도 메트릭 [일반논문] 임창남, 박예슬(아주대), 이광재(아주대 의과대학), 이정원(아주대) 개인지원 로봇의 머신러닝 기능을 위한 리스크 감소 프로세스 [단편논문] 이연재, 한혁수(상명대) 인공지능 센터-엣지 플랫폼 아키텍처와 사례 [후원 산업체 발표] 박병훈 대표(T3Q)
11:00-11:10	**휴식**			

2 월 4 일 (화)				
시 간	**행 사 내 용**			
	논문 발표 B			
	B1: SW 안전 좌장: 서영석(영남대) 장소: 세미나실 1	**B2: 스마트 시스템 I** 좌장: 이지현(전북대) 장소: 세미나실 2	**B3: 인공지능 II (AI for SE)** 좌장: 남재창(한동대) 장소: 세미나실 3	**B4: SW 분석** 좌장: 홍장의(충북대) 장소: 그랜드홀 2
11:10-12:30 (80 분)	소프트웨어 위험원 분석을 위한 상황 분류 기반의 조합을 통한 STPA 문맥표 최적화 [일반논문] 양현수, 권기현(경기대) 결함 트리와 마코프 모델을 이용한 안전 기능의 정량적 검증 **[최우수 일반논문]** Ngoc-Tung La, 김소연, 권기현(경기대) 기능 안전 표준 기반의 무기체계 소프트웨어 개발 및 관리 매뉴얼 문제점 분석 및 개선 방안 제시 **[우수 산업체 논문]** 김태현, 박다운, 백옥현(국방과학연구소) STPA 를 이용한 군집 운행 시스템의 안전성 분석 사례 연구 [단편논문] 김의섭, 유준범(건국대)	Effect-driven Dynamic Selection of Physical Media for Visual IoT Services using Reinforcement Learning **[초청논문 ICWS'19]** K. Baek, I-Y. Ko 데이터베이스 서버 그룹 관리를 위한 EMS [산업체 논문] 김효일(KT), 백종문(KAIST) 트리거-액션 기반 서비스 매쉬업의 신뢰도를 개선하기 위한 전제 상태 및 목표 상태 검증 방법 [단편논문] 김상훈, 고인영(KAIST) BERT 기반의 SNS 메시지 불안 분류기 및 사회적 불안 분포 시각화 시스템 [단편논문] 송지수, 최용석(한양대)	소프트웨어 결함 예측의 조선해양/해상운송 산업 적용 사례 연구 **[우수 일반논문]** 강종구(KAIST), 류덕산(전북대), 백종문(KAIST) 순환 인공 신경망을 활용한 코드 변경 추천 시스템의 학습 시간 단축 방법 연구 **[우수 일반논문]** 배병일, 강성원(KAIST), 이선아(경상대) 딥러닝 기반 버그 추적 기법의 OOV 단어 처리를 위한 형태 정보와 문맥 정보 통합 [단편논문] 김영경, 김미수, 이은석(성균관대) 이슈 보고서와 사용자 매뉴얼 간의 추적성 수립을 위한 머신러닝 기법들의 비교 [단편논문] 조희태, 박도제, 이선아(경상대)	Precise Learn-to-Rank Fault Localization using Dynamic and Static Features of Target Programs **[초청논문 TOSEM]** Y. Kim, S. Mun, S. Yoo, M. Kim 행위 모델 기반 RESTful 웹 어플리케이션 결함 위치 추정 **[우수 일반논문]** 장종인, 이낙원(KAIST), 류덕산(전북대), 백종문(KAIST) 조치 가능한 버그 예측을 위한 유사 패치 추천 **[우수 단편 논문]** 신지호, 남재창(한동대) Python 코드를 위한 정적 분석기 개발 [단편 논문] 홍제성, 박보경, 장우성, 이원영, 정세준, 김영철(홍익대)
12:30-13:40	**중식**			

	논문 발표 C			
	C1: 학부생 논문 I 좌장: 서영석(영남대) 장소: 세미나실 1	**C2: 학부생 논문 II** 좌장: 홍신(한동대) 장소: 세미나실 2	**C3: 블록체인** 좌장: 김순태(전북대) 장소: 세미나실 3	**C4: SW 테스팅 I** 좌장: 백종문(KAIST) 장소: 그랜드홀 2
13:40-15:40 (120 분)	CNN 모델 평가를 위한 이미지 데이터 증강 도구 개발 **[우수 학부생 논문]** 최영원, 이영우, 채흥석(부산대) 소프트웨어 소스 코드의 다중 카테고리 분류를 위한 딥러닝 기반 기법 김민하, 심규진, 이찬근(중앙대) YOLO 모델을 활용한 영상 내의 유해 정보 검출 및 필터링 시스템 개발 조성윤, 권용준, 김채록, 최지원, 김석호, 최나람, 김민석, 정순기(경북대) 텍스트 마이닝과 TF-IDF 유사도 가중치를 이용한 연관 아이템 발견 김민정, 김주창, 박성수, 신동훈(경기대), 정호일(대림대), 정경용(경기대) 문서 유형에 따른 분류를 이용한 마이닝 기반 토픽 추출 구병국, 최소영, 강지수, 백지원(경기대), 박찬홍(상지대), 정경용(경기대) AI Impact on Software Engineering [후원 산업체 발표] 김동영(한국 IBM)	블록체인 기반 CCTV 영상정보 무결성 지원 방안 강민구, 홍준기, 송성한, 김태영, 김순태(전북대) STEAM 교육과 게임을 도입한 EPL 기본 설계 **[우수 학부생 논문]** 이정희, 김민우, 조성휘, 김정아(가톨릭관동대) 도커시스템에서 사용 가능한 모니터링 도구 비교 분석 정채은, 박용범(단국대) 미들웨어 기반 빅데이터 시각화 시스템 아키텍처 설계 백재형, 송진범, 서상원, 남재창(한동대) AI 와 SE 를 활용한 수화 번역 프로젝트 이유렬, 김은진, 이혁진, 이선아(경상대) CCTV 를 이용한 실시간 폭행 및 난동행위 인식 시스템 개발 권용휘, 전승원, 배재빈, 김성윤, 김석호, 정순기(경북대)	이더리움 스마트 컨트랙트 상태 모니터링 시스템의 설계 및 구현 **[초청논문 JSES]** 홍준기, 김순태, 류덕산 실시간 인식 기반의 제품의 표시 정보 추출을 위한 모바일 어플리케이션 설계 및 구현 **[초청논문 JSES]** 민경식, 최지수, 이철훈, 정동주, 이병정 계약 관련 음성 녹취 정보의 신뢰성 있는 저장과 체계적 관리를 위한 블록체인 기반 시스템 구성 방식 [단편 논문] 김인근, 서중원, Alshiri Saad, 장민오, 이유정, 박수용(서강대) 그룹 리더 선출을 통한 비트코인 병렬 트랜잭션 처리 [단편 논문] 이도현, 송아람, 이선재, 박우람, 박수용(서강대) 블록체인 코드의 복잡도 측정을 위한 코드 가시화 [단편 논문] 안현식(홍익대), 박지훈, 김기두(TTA), 박보경, 조재형, Md Ibrahim Khalil(홍익대), 김동호(데이터메트릭스), 김영철(홍익대) 블록체인 기반 실시간 수하물 트레킹 시스템 [단편 논문] 이열국, 이동현, 박수용(서강대)	Concolic Testing for High Test Coverage and Reduced Human Effort in Automotive Industry **[초청논문 ICSE '19]** Y. Kim, D. Lee, J. Baek, M. Kim Javadoc 대상 테스트 요구사항 추출 기법의 정확도 평가 [단편 논문] 김지웅, 홍신(한동대) Enhancing Patch Validation in APR by Introducing Test Case Prioritization and Sampling [단편 논문] Y. Venugopal, Q-N. Phung, E. Lee(성균관대) 무인이동체 소프트웨어 시험 평가 기준 및 절차 연구 [단편 논문] 장정훈(모아소프트) DO-178C 기반의 항공 SW Safety 검증 프로세스 [후원 산업체 발표] 강유선(모아소프트)
15:40-16:00	**휴식**			
	기조강연 II 장소: 그랜드홀 2			사회: 이병정 대회장 (서울시립대)
16:00-16:50	인공지능과 인간의 공존, 그리고 포스트 휴머니즘 이중원 교수 (서울시립대, 한국철학회 차기회장)			
	기조강연 III 장소: 그랜드홀 2			사회: 김정아 대회장 (가톨릭관동대)
16:50-17:40	Build AI-Powered Applications through Data driven Innovation 최창남 부문장 (한국 오라클)			

17:40~18:00	**휴식**
18:00-21:00	**우수논문상, 공로상, 감사장 수여식** 사회: 김정아 조직위원장 (가톨릭관동대) **만찬** 사회: 한종대 교수 (상명대) 장소: 그랜드홀 1

2 월 5 일 (수)

시 간	행 사 내 용		
	논문 발표 D		
	D1: 아키텍처 및 유지보수 좌장: 이지현(전북대) 장소: 세미나실 1	**D2: 스마트 시스템 II** 좌장: 남재창(한동대) 장소: 세미나실 2	**D3: 인공지능 III (AI for SE)** 좌장: 류덕산(전북대) 장소: 세미나실 3
9:20-10:40 (80 분)	Automatic Detection and Update Suggestion for Outdated API Names in Documentation **[초청논문 TSE]** S. Lee, R. Wu, S.C. Cheung, S. Kang 무기체계 항법소프트웨어 플랫폼 피처모델링 [산업체 논문] 박병수, 백승준, 이인섭, 서강선, 노성규, 박삼준(국방과학연구소) 자동화된 소프트웨어 가변성 추출을 위한 정의/사용 정보 기반 제품군 공유 코드 정렬 [단편 논문] 김태영, 이지현(전북대)	세그멘테이션 기반 검출과 어텐션 기반 인식을 활용한 유통기한 OCR [산업체 논문] 문형진, 나병진(에스에스지닷컴) E-Learning 을 위한 Open CV 기반 집중도 측정 시스템 설계 [단편 논문] 임대근, 조재춘(상명대) PPMI 와 NPMI 를 이용한 자동화 된 감성 사전 구축 방법 [단편 논문] 류기곤(고려대) 온라인 교육을 위한 Word2vec 기반의 핵심 문장 추출 시스템 설계 [단편 논문] 유준영, 유재준, 조재춘(상명대) '아동 및 어린이' 키워드를 이용한 뉴스 기사 분석 [단편 논문] 최은지(청주대 문헌정보), 박지연(청주대), 김용환(청주대 문헌정보), 노기섭(청주대)	기계학습 기반 안전중심 소프트웨어의 학습 데이터 품질 평가 [일반논문] 김문현, 나자캇알리, 홍장의(충북대) Seq-GAN 알고리즘을 활용한 자동 버그 정정 기법 [일반논문] 양근석, 이철훈, 최현호, 이병정(서울시립대) HCAII : 고융합 AI 인프라스트럭쳐(infrastructure)에 대한 가성비 및 확장성 분석 [단편 논문] 전민규, 최규휘, 김지원, 성병학, 정재웅(아토리서치), 강양욱, 기양석(삼성전자) 국내외 인공지능 지식 구조 및 변동 분석: 한국, 미국, 중국의 논문 계량분석을 중심으로 [단편 논문] 박종현, 김문구(한국전자통신연구원)
10:40-10:50	**휴식**		

2 월 5 일 (수)			
시 간	**행 사 내 용**		
	논문 발표 E		
	E1: 학부생 논문 III 좌장: 이선아(경상대) 장소: 세미나실 1	**E2: SW 검증** 좌장: 배경민(포항공대) 장소: 세미나실 2	**E3: SW 테스팅 II** 좌장: 홍신(한동대) 장소: 세미나실 3
10:50-12:10 (80 분)	스테이트먼트 유형 정보를 활용한 옳은 패치 생성률 증대 기법 허진석, 정호현, 이은석(성균관대) IoT 초보 개발자를 위한 검색어 향상 정찬미, 남재창(한동대) 오픈소스 라이선스 충돌 여부 식별을 위한 TF-IDF 및 LDA 기법 성능 분석 남성국, 이동건, 서영석(영남대) 텍스트 마이닝을 사용한 Configuration 버그 리포트 예측 최정환, 최지원, 류덕산, 김순태(전북대) BERT 를 이용한 중복 버그 보고서 검색 모델 문석암, 강민구, 류덕산, 김순태(전북대)	Model checking embedded control software using OS-in-the-loop CEGAR **[초청논문 ASE'19]** D. Kim, Y. Choi IoT 운영체제 모델 기반 파밴드 어플리케이션 안전성 검증 [단편 논문] 양송, 신영술, 김동우, 김동영, 최윤자(경북대) 소프트웨어 결함 예측 모델을 위한 N-gram 로그 확률 메트릭 [단편 논문] 원은지, 남재창(한동대) A Model Projection Technique for Compositional Verification using Model Checking [단편 논문] 이동아, 유준범(건국대)	헬리콥터 자동비행조종컴퓨터 검증을 위한 시나리오 기반 시험 사례 [산업체 논문] 김재만(한국항공우주산업), 이선아(경상대) Automated mutant generation for function block diagram programs **[우수 단편 논문]** L. Liu, 지은경, 배두환(KAIST) 임베디드 소프트웨어에 대한 테스트케이스 생성을 지원하는 분산 Concolic 테스팅 도구의 설계와 구현 **[우수 단편 논문]** 최한솔, 임혜린, 김효림, 홍신(한동대)
12:10-12:40 (30 분)	**폐회식** 장소: 그랜드홀 2		사회: 김정아 조직위원장 (가톨릭관동대)

KCSE 2020 튜토리얼

튜토리얼 T1: ADOxx Meta-Modeling Platform

- 일시: 2월 3일(월) 13:00~14:30
- 장소: 세미나실 1
- 제목: ADOxx Meta-Modeling Platform
- 연사: 이문근 교수 (전북대학교 컴퓨터공학부)
- 튜토리얼 초록:

  소프트웨어공학에서 SW 제품 생산 공정 과정은 SW 품질을 결정하는 매우 중요한 과정이다. 그리고, 이 공정 과정 중, 모델링 Activity 는, SW 제품이 구조 및 기능적 요구사항에 맞게 설계되고, 설계된 결과물이 안전성과 보안성을 보장하는 지를 분석하고 검증할 수 있는 필수적인 단계이다. 특히, 이 단계에서의 모델링 도구 사용은 최종 SW 제품의 품질과 공정 과정의 생산성을 결정하는 매우 중요한 수단이다. 이 Tutorial 에서는 오스트리아 비엔나대학교의 세계적인 OMiLAB 모델링 연구소에서 개발한 ADOxx Meta-Modeling Platform 을 소개한다. ADOxx 는 이러한 모델링 도구를 개발하기 위한, Modeling Language, Modeling Techniques, 그리고 Modeling Mechanism & Algorithm 을 기반으로 한 Modeling Method 를 제공한다. 이 기법에 따라, 모델링 도구의 기본 기능을 구현할 수 있는, Modeler, Analyzer, Simulator, Evaluator 의 기본 Framework 과 Library 를 제공한다. 현재까지 OMiLAB 연구소에 등록된 Open 모델링 도구는 약 75 개에 이른다.
- 약력:
  - 2015 ~ 현재: 전북대학교, OMiLAB Korea 연구소 소장
  - 1996 ~ 현재: 전북대학교, 컴퓨터공학부, 교수
  - 미국, Philadelphia, CCCC, Computer Scientist
  - 미국, University of Pennsylvania (Upenn), Computer & Information Science, 석사/박사
  - 미국, Pennsylvania State University (PSU), Computer Science, 학사

튜토리얼 T2: Concolic 유닛 테스팅

- 일시: 2월 3일(월) 13:00~14:30
- 장소: 세미나실 2
- 제목: 오류 검출에 효과적인 Concolic 유닛 테스팅
- 연사: 김윤호 교수 (KAIST 전산학부)
- 튜토리얼 초록:

  본 튜토리얼에서는 SW 오류 검출에 효과적인 Concolic 유닛 테스팅 기법을 소개한다. SW 자동 테스트 생성 기법인 Concolic 테스팅 기법을 먼저 소개하고 Concolic 테스팅 기법을 활용해 C 프로그램의 유닛 테스트 드라이버/스텁/테스트 입력 값을 자동으로 생성하는 Concolic 유닛 테스팅 기법을 설명한다. 또한, 산업체 적용 사례를 통해 Concolic 유닛 테스팅 기법이 오류 검출 및 테스트 커버리지 향상에 얼마나 효과적인지 보여준다. 마지막으로 Concolic 유닛 테스팅을 쉽게 적용할 수 있는 CROWN2 도구를 소개하여 Concolic 유닛 테스팅이 실제 개발 현장에 적용될 수 있는 완성도 높은 기술임을 보여준다.
- 약력:
  - 2018 ~ 현재: KAIST 전산학부 연구조교수
  - 2017 ~ 2018: KAIST 정보전자연구소 연수연구원
  - 2017: KAIST 전산학부 박사
  - 2007: KAIST 전산학과 학사

튜토리얼 T3: 전산학 논문 작성법 (Technical Writing in English)

- 일시: 2월 3일(월) 13:00~14:30
- 장소: 세미나실 3
- 제목: 전산학 논문 작성법 (Technical Writing in English)
- 연사: 차성덕 교수 (고려대학교)
- 튜토리얼 초록:

  열심히 연구하여 좋은 결과를 내는 것도 필요하지만, 연구결과를 논문으로 정리하는 작업도 매우 중요합니다. 그리고 심사위원이나 다른 연구자들이 핵심적인 내용을 제대로 그리고 쉽게 이해할 수 있도록 논문을 쓰는 것은 매우 중요합니다. 영어 작문실력도 필요하겠지만 논리적인 사고, 체계적인 접근방법이 논문작성에 필요한 이유입니다. 본 튜토리얼은 석.박사 과정 지도학생의 논문 지도의 경험을 바탕으로 어떻게 하면 논문 작성을 잘 할 수 있는지 같이 생각해 보고자 합니다. 논문작성에서 도움이 되는 자료도 소개하겠습니다.
- 약력:
  - 2008 ~ 현재: 고려대학교 정보대학 정교수
  - 1994 ~ 2008: KAIST 조교수/부교수/정교수
  - 1991 ~ 1994: The Aerospace Corporation, El Segundo, CA 연구원

튜토리얼 T4: AI 시스템의 품질 특성 및 검증 방법

- 일시: 2월 3일(월) 13:00~14:30
- 장소: 그랜드홀 2
- 제목: AI 시스템의 품질 특성 및 검증 방법과 테스트 오라클이 없을 때 적용 가능한 블랙박스 테스팅
- 연사: 최영재 팀장 (STA테스팅컨설팅)
- 튜토리얼 초록:

  AI 시스템의 활용이 늘어나고 있는 상황에서 AI 시스템의 비결정성 및 확률성으로 인해 ISO 25010 에서 정의하는 품질 특성만으로 부족한 면이 있다. AI 의 품질 특성을 정의하고 있는 표준이 아직은 없는 상황에서 ISO 의 Software testing WG 의장이 제시하는 AI 시스템의 신뢰성을 높일 수 있는 9 가지 품질 특성과 해당 품질 특성을 검증하기 위해 사용할 수 있는 테스트 기법을 소개한다. AI 시스템이 가지는 특성으로 인해 테스트에 필요한 테스트 오라클을 정의하기 어렵고 테스트 케이스의 기대 결과를 예측하기가 힘들다. 이렇게 테스트 오라클이 없는 상황에서 적용할 수 있는 블랙박스 테스트 기법 3 가지, 백-투-백 테스팅, A/B 테스팅, 메타모픽 테스팅을 소개하고 구체적인 예제를 통해 살펴본다.
- 약력:
  - 2019 ~ 현재: ISTQB AI Testing Syllabus Task Force
  - 2018 ~ 현재: STA 테스팅컨설팅 컨설팅혁신팀 팀장
  - 2015 ~ 2017: KSTQB 책임연구원
  - 2013 ~ 2014: Cisco Systems QC 팀 팀장
  - 2007 ~ 2013: NDS QC 팀 팀장

튜토리얼 T5: SW 안전 확보를 위한 기법: 도메인 적용 방안 및 사례

- 일시: 2월 3일(월) 14:40~16:10
- 장소: 세미나실 1
- 제목: SW 안전 확보를 위한 기법: 도메인 적용 방안 및 사례
- 연사: 지은경 교수 (KAIST)
- 튜토리얼 초록:

  안전 필수 시스템에서 SW 비중이 높아지면서 SW 로 인한 위험이 발생하지 않도록 SW 안전성을 확보하기 위한 기법의 개발 및 적용이 중요해지고 있다. 본 튜토리얼에서는 SW 안전 확보를 위한 기법들 중 모델체킹, 테스팅, safety case 기법을, 도메인에 적용한 사례를 중심으로 소개한다. 원자력 분야 소프트웨어 대상 정형검증 및 테스팅 기법 개발 및 적용 사례, 의료 분야와 원자력 분야 소프트웨어 시스템 대상 safety case 작성 사례를 소개한다.
- 약력:
  - 2019 ~ 현재: KAIST 전산학부 연구부교수
  - 2011 ~ 2019: KAIST 전산학부 연구조교수
  - 2009 ~ 2011: University of Pennsylvania 박사후연구원
  - 2009: KAIST 전산학 박사 (소프트웨어공학 전공)
  - 2003 ~ 2004: ㈜이마린로직스 소프트웨어 엔지니어
  - 2001 ~ 2002: 몽골 울란바타르 대학 컴퓨터학과 전임강사
  - 2001: KAIST 전산학 석사 (소프트웨어공학 전공)
  - 1999: KAIST 전산학 학사

튜토리얼 T6: 블록체인의 개념, 응용 사례 및 연구 이슈 소개

- 일시: 2월 3일(월) 14:40~16:10
- 장소: 세미나실 2
- 제목: 블록체인의 핵심 개념, 산업별 응용 사례 그리고 소프트웨어공학 연구 이슈 소개
- 연사: 김순태 교수 (전북대학교)
- 튜토리얼 초록:

  최근 들어 블록체인은 4 차산업 혁명의 핵심 기술 중 하나로 빠짐없이 등장하고 있다. 하지만, 아직 많은 연구자들은 이에 대한 개념 정도만 이해하고 있고, 얼마나 사회적인 큰 파급효과가 있는지에, 그리고 소프트웨어공학 연구자로서 어떤 접근을 취해야할 지에 대하여도 막연한 상황이다. 본 튜토리얼을 통하여 블록체인의 핵심 개념을 이해하고, 이를 바탕으로 최근 블록체인의 이슈인 STO (Security Token Offering), FaceBook 의 Libra 등의 사회적 변화, 컨텐트 시장의 변화, 뿐만 아니라 국내 블록체인의 국내 시범 사업등의 산업별 적용 사례를 알아본다. 더불어 소프트웨어공학 전공자로써 블록체인 분야에 어떠한 연구 이슈 및 접근 방법이 있는지에 대해서도 파악해 보는 시간을 갖는다.
- 약력:
  - 2014 ~ 현재: 전북대학교 소프트웨어공학과 교수
  - 2011 ~ 2013: 강원대학교 컴퓨터공학과 교수
  - 2005 ~ 2010: 서강대학교 컴퓨터공학과 석사/박사
  - 2003 ~ 2005: 소프트웨어 크래프트 컨설팅 엔지니어/컨설턴트
  - 2003: 중앙대학교 컴퓨터공학과 학사
- 연구분야: 소프트웨어 공학(SW 아키텍처, 디자인 패턴, 소스코드 마이닝), 블록체인(스마트 컨트랙트)

튜토리얼 T7: 의미 기반 프로그램 수정 (Semantics-based Program Repair)

- 일시: 2월 3일(월) 14:40~16:10
- 장소: 세미나실 3
- 제목: 의미 기반 프로그램 수정 (Semantics-based Program Repair)
- 연사: 이주용 교수 (UNIST)
- 튜토리얼 초록:

  오늘날의 소프트웨어는 끊임없이 변화 혹은 진화하며 다양하고 빠르게 변화하는 현대 사회의 요구를 수용하고 있습니다. 하지만 이에 수반하여 소프트웨어의 오류도 계속적으로 발생하고 있으며, 수동으로 일일이 오류들을 수정하는 전통적인 개발 방법이 점차 한계에 봉착하고 있습니다. 이에 따라 최근 자동으로 프로그램 오류를 수정하는 연구가 활발히 진행되고 있습니다. 본 튜토리얼에서는 자동 프로그램 오류 수정 기술에 대해 소개하고자 합니다. 특히, 제가 개발해오고 있는 의미 기반 프로그램 수정 방법에 초점을 맞추어 설명할 예정입니다. 의미 기반 방법은 오류가 있는 프로그램에서 필요한 프로그램 행위 변경을 추론하고 이에 기반하여 패치를 합성하는 방법을 취합니다. 본 튜토리얼에서는 의미 기반 방법이 어떻게 발전되어 왔는지 설명할 예정입니다.
- 약력:
  - 2019 ~ 현재: UNIST 전기전자컴퓨터공학부 조교수
  - 2017 ~ 2019: Innopolis University 조교수
  - 2011 ~ 2016: 싱가포르 국립 대학 박사후 연구원
  - 2010 ~ 2011: 고려대학교 박사후 연구원 및 연구교수
  - 2008 ~ 2010: 캔자스 주립 대학교 박사후 연구원
  - 2007 ~ 2008: KAIST 박사후 연구원
  - 2007: Aarhus University (BRICS) 전산학 박사
- 연구분야: 소프트웨어공학, 프로그램 분석/합성/수정

KCSE 2020 기조강연

기조강연 I

- 일시: 2월 3일(월) 16:40~17:30
- 장소: 그랜드홀 2
- 제목: Smart S/W Development : AI for Software Engineering
- 연사: 김강태 상무 (삼성전자)

기조강연 II

- 일시: 2월 4일(화) 16:00~16:50
- 장소: 그랜드홀 2
- 제목: 인공지능과 인간의 공존, 그리고 포스트 휴머니즘
- 연사: 이중원 교수 (서울시립대, 한국철학회 차기회장)

기조강연 III

- 일시: 2월 4일(화) 16:50~17:40
- 장소: 그랜드홀 2
- 제목: Build AI-Powered Applications through Data driven Innovation
- 연사: 최창남 부문장 (한국 오라클)

KCSE 2020 신진 연구자 초청 발표

신진 연구자 초청 발표 I

◆ 일시: 2월 3일(월) 17:30~18:20

◆ 장소: 세미나실 1

◆ 제목: 지속적 통합 환경에서 비용효율적인 소프트웨어 테스트를 위한 테스트 환경 우선순위 부여 기법
Prioritizing test environments for cost-effective software test in continuous integration

◆ 연사: 권정현 박사 (KT, KAIST 졸업)

◆ 초록:

이 논문에서는 어플리케이션을 테스트할 때, 여러 다른 환경에서 테스트를 비용효율적으로 수행하는 것을 다루었다. 웹브라우저 종류나 플랫폼 등의 환경 속성에 따라 다른 종류의 테스트 실패가 발생할 수 있다. 따라서 어플리케이션이 개발됨에 따라 여러 환경에서 회귀 테스트가 수행되어야 한다. 지속적 통합 환경(Continuous Integration, CI)이 보편화 되면서, 코드가 중앙 코드 저장소에 업데이트가 되면, 다양한 테스트 환경에서의 회귀 테스트를 자동으로 수행하는 프로젝트가 늘어나고 있다. 테스트 결과는 개발자에게 전달되어 디버깅 등의 다음 개발 활동을 수행한다. 하지만 많은 환경에서 회귀 테스트가 수행됨에 따라 전체 테스트 시간과 비용이 많이 들고, 이로 인해 테스트 실패에 대한 피드백이 지연될 수 있는 문제가 있다. 이 연구에서는 다양한 테스트 환경에서 회귀테스팅을 수행할 때, 개발자에게 실패에 대한 피드백을 더 빨리 줄 수 있도록 4개의 기술과 2개의 하이브리드 방법을 제안한다. 제안 기술은 테스트 우선순위 기법에서도 자주 사용되는 최근 실패 정보, 자주 실패되는 정도를 기반으로 한다. 제안 기술은 특히 지속적 통합 환경에서 사용하기 적절하다. 대규모 CI 환경일수록 회귀 테스트 간의 시간 간격이 짧다. 개발자는 중앙 코드 저장소에 자주 커밋을 하고, 코드와 관련 있는 회귀 테스트가 독립적인 환경에서 실행된다. 기존의 코드 커버리지를 바탕으로 하는 비용 효율적인 회귀 테스트 기술은 이러한 과정에서 발생하는 코드 변경 속도를 따라가기가 어렵다. 제안 기술은 5개의 인기있는 오픈소스 웹 어플리케이션에 대해 실증적으로 검증되었다. 실험 결과로 제안 기술이 비용 효율적임을 보였다. 제안 기술은 일반적으로 두개의 베이스라인 기술 (우선순위를 적용하지 않은 방법, 무작위로 우선순위를 적용한 방법)보다 더 빠르게 실패를 감지하였다. 그리고, 제안된 우선순위 기술은 서로 비교를 통해 어떤 기술이 가장 비용효율적인지 실험하였다. 뿐만 아니라, 본 연구는 CI 개발환경에서 개발자가 관심있어 하는 테스트 환경에서의 테스트 결과를 우선적으로 알려주기 위한 방법도 다루었다.

신진 연구자 초청 발표 II

- 일시: 2월 3일(월) 17:30~18:20
- 장소: 세미나실 2

- 제목: 문맥기반 변경사항 적용기법을 이용한 자동 패치생성
  Automatic Patch Generation with Context-based Change Application

- 연사: 김진대 박사 (홍콩과기대 졸업)

- 세미나 초록:
  자동 패치생성 문제는 주어진 결함이 있는 코드와 이를 검증할 수 있는 테스트들에 대해서 코드를 수정하여 모든 테스트를 통과하는 패치를 자동으로 생성하는 문제이다. 이를 위해 자동 패치생성 기법들은 주로 저마다의 방법으로 코드를 변경하여 수많은 패치 후보들을 생성한다. 이 패치후보 생성은 어느 한 패치 후보가 모든 테스트를 통과할 때까지 반복되는데, 이런 방식을 생성과 검증(Generate-and-Validate) 방식이라고 부른다. 이 방식을 따르는 자동 패치생성 기법이 고려해야 할 점은 두 가지이다. 하나는 얼마나 다양한 패치 후보를 생성하여 보다 많은 결함에 대한 패치를 확보할 것인가이고, 다른 하나는 얼마나 효율적으로 테스트를 통과하는 패치를 생성할 것인가이다. 본 논문에서는 새로운 기법인 ConFix를 소개한다. ConFix는 우선 과거의 패치에서 코드 변경사항과 함께 이들의 추상문법트리 문맥(Abstract Syntax Tree Context)을 수집한다. 이후 결함이 있는 코드에 이 문맥이 일치하는 경우에만 변경사항을 적용하는 문맥기반 변경사항 적용기법(Context-based Change Application)을 사용하여 패치후보를 생성한다. 이 방식으로 ConFix는 수집된 변경사항을 이용해 매우 다양한 패치 후보를 생성할 수 있고, 변경사항을 문맥기반으로 적용하여 효율적으로 필요한 패치를 생성할 수 있다.

신진 연구자 초청 발표 III

- 일시: 2월 3일(월) 17:30~18:20
- 장소: 세미나실 3

- 제목: Learning Heuristics for Fast and Precise Java Points-to Analysis

- 연사: 정세훈 박사 (고려대학교 졸업)

- 세미나 초록:

  This talk demonstrates that data-driven techniques can significantly outperform hand-crafted approaches by experts in generating effective abstraction heuristics for Java pointer analysis. Heuristics for context-sensitivity and heap abstraction are essential ingredients of effective analysis. Existing heuristics generally have to balance trade-offs between scalability and precision, whereas the proposed approach is highly scalable while achieving acceptably low levels of reduction in precision: 1) The process of generating heuristics, based on machine learning approaches, is automated in that complex heuristics are derived as a combination of atomic features; 2) Resulting heuristics, when applied to widely used DaCapo benchmark programs, are highly efficient compared to the ones manually crafted by human experts; and 3) Precision of the learned heuristics is also superior to the hand-tuned rules when applied to context-sensitive analysis. When applied to the heap abstraction problem, precision level is almost equivalent in that additional alarms generated by data-driven heuristics never exceeded by more than 4%.

우수 국제학회/학술지 초청 논문발표

◆ **Precise Learn-to-Rank Fault Localization using Dynamic and Static Features of Target Programs**
- ACM Transactions on Software Engineering and Methodology (TOSEM), Vol.28, No.4, 2019
- Yunho Kim, Seokhyeon Kim, Shin Yoo, and Moonzoo Kim
- Session: B4 (SW 분석)

◆ **Automatic Detection and Update Suggestion for Outdated API Names in Documentation**
- IEEE Transactions on Software Engineering (TSE), 2019
- Seonah Lee, Rongxin Wu, S.C. Cheung, and Sungwon Kang
- Session: D1 (아키텍처 및 유지보수)

◆ **Concolic Testing for High Test Coverage and Reduced Human Effort in Automotive Industry**
- The 41st ACM/IEEE International Conference on Software Engineering (ICSE 2019)
- Yunho Kim, Dongju Lee, Junki Baek, and Moonzoo Kim
- Session: C4 (SW 테스팅 I)

◆ **Model checking embedded control software using OS-in-the-loop CEGAR**
- The 34th IEEE/ACM International Conference on Automated Software Engineering (ASE 2019)
- Dongwoo Kim and Yunja Choi
- Session: E2 (SW 검증)

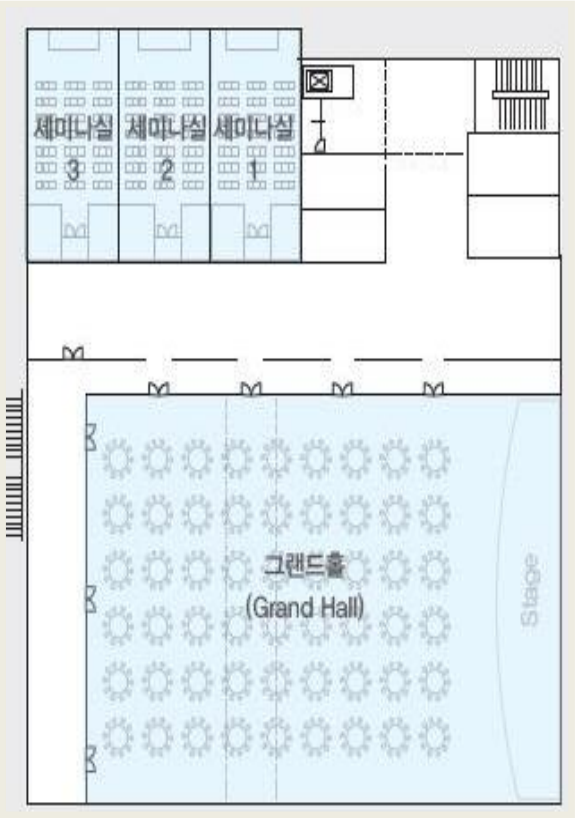
◆ **Effect-driven Dynamic Selection of Physical Media for Visual IoT Services using Reinforcement Learning**
- The 24th IEEE International Conference on Web Services (ICWS 2019)
- Kyeongdeok Baek and In-Young Ko
- Session: B2 (스마트 시스템 I)

◆ **이더리움 스마트 컨트랙트 상태 모니터링 시스템의 설계 및 구현**
- Journal of Software Engineering Society (JSES), Vol.28, No.2 2019
- 홍준기, 김순태, 류덕산
- Session: C3 (블록체인)

◆ **실시간 인식 기반의 제품의 표시 정보 추출을 위한 모바일 어플리케이션 설계 및 구현**
- Journal of Software Engineering Society (JSES), Vol.28, No.2 2019
- 민경식, 최지수, 이철훈, 정동주, 이병정
- Session: C3 (블록체인)

MEMO

KCSE 2020 행사장 및 식당 위치



1 F

B1

논문 목차

일반논문

단편논문

산업체 논문

학부생 논문

기계학습 기반 안전중심 소프트웨어의 학습 데이터 품질 평가

김문현 [1,O] 나자캇알리 [2] 홍장의 [2]
[1] 충북대학교 국제경영학과, [2] 충북대학교 컴퓨터과학과
moonhyun.dev@gmail.com jehong@chungbuk.ac.kr

Evaluating the Quality of Learning Data in Machine Learning-Based Safety Critical Software

Moon-Hyun Kim[1,O], Nazakat Ali[2], and Jang-Eui Hong[2]
[1]Dept of International Business, Chungbuk National University, South Korea
[2]Dept of Computer Science, Chungbuk National University, South Korea

요 약

최근 인공지능 기술의 확산으로 인하여 기계 학습과 딥 러닝 기반의 다양한 응용 소프트웨어 개발이 이루어지고 있으며, 임무 중심 소프트웨어, 안전 중심 소프트웨어 분야에서도 인공지능 기술이 적용되는 추세이다. 기계 학습 기반의 소프트웨어의 정확성 및 품질을 향상시키기 위해서는 어떤 데이터로 학습을 진행하는가가 매우 중요하다. 이는 학습 데이터의 품질에 따라 전체적인 소프트웨어의 학습 품질을 예측해 볼 수 있으며, 이에 따른 서비스 결과도 결정되기 때문이다. 학습 데이터의 중요성에도 불구하고, 기존의 연구들에서는 학습 데이터의 생성 방법에 대한 연구들이 진행되었고, 구성된 학습 데이터의 품질을 평가할 수 있는 기준이나 방법은 제시되지 않았다. 따라서 본 논문에서는 기계학습 기반 소프트웨어의 학습 데이터의 품질을 평가할 수 있는 기준을 제시한다. 제시 된 기준들은 학습 데이터를 사용하는 지도 학습 기반의 소프트웨어를 개발할 시 활용 할 수 있다. 이러한 기준들을 통해 기계 학습을 진행하기 전 소프트웨어의 품질이나 정확성을 예측하거나 예측한 결과에 따라 학습을 진행 할지, 데이터를 재구성할 지 판단할 수 있다. 이는 예측 정확성이 낮은 데이터로 학습 하는 것을 방지 해 시간, 비용적으로 낭비를 줄일 수 있으며, 트랜드 변화 주기가 빠른 현재 시장 환경에서 소프트웨어의 개발 시간을 단축시키는데 기여할 수 있다.

1. 서 론

최근 기계 학습과 딥 러닝 기반의 다양한 응용 소프트웨어 개발이 이루어 지고 있다. 이렇게 개발된 많은 소프트웨어가 사용자들에게 서비스를 제공하고 있는데, 이러한 소프트웨어 중 임무 중심, 안전 중심 소프트웨어의 경우 학습 알고리즘에 대해 의존적인 결과를 제공한다. 이러한 특성 때문에 학습 알고리즘의 성능이 매우 중요하여 학습 알고리즘의 정확성을 높이기 위한 여러 연구들이 진행되고 있다[1,15,20]. 소프트웨어가 학습을 하는 방식에는 지도 학습, 비 지도 학습, 강화 학습, 준지도 학습 등의 방법이 있으며 각 방식에 따라 학습 알고리즘 또한 상이하다[21]. 그 중 지도 학습 방법은 라벨링된 학습 데이터를 이용하여 분류나 회귀 분석을 목적으로 하는 소프트웨어를 개발하는데 사용되고 있다[2]. 따라서 지도 학습 기반 소프트웨어에서는 학습 알고리즘 이외에도 학습에 사용되는 데이터 집합이 필요한데, 특히 안전 중심 소프트웨어의 품질과 정확성을 보증하기 위해서는 학습 데이터의 품질이 매우 중요하다. 이는 학습 데이터의 품질을 통해 안전 중심 소프트웨어의 학습 품질을 통한 기능 안전성을 대비한다는 관점이다. 또한 좋은 품질의 데이터는 스스로 그 데이터가 가진 가치를 보장할 수 있기 때문에, 큰 데이터 집합을 사용하는 소프트웨어에서의 사전 조건이 될 만큼 필수적이다[18]. 기존의 여러 연구들에서 학습 데이터를 구성하는 방법으로 수집한 데이터 셋을 단순히 일정 비율로 나누는 방법이나 근접한 데이터의 셋들을 하나의 데이터

유형으로 나누는 방법 등 여러 방법들이 소개된 바 있다[5]. 일반적인 어플리케이션 데이터의 품질을 평가하기 위한 방법이 기존에 연구 되어 왔지만, 기계학습 기반 소프트웨어의 학습에 사용 될 데이터의 품질을 평가할 수 있는 기준이나 방법에 대한 연구는 찾아보기 어렵다. 따라서 본 논문에서는 기계학습 기반의 학습 데이터의 품질을 평가할 수 있는 기준을 다음과 같이 다섯 가지로 제시한다.

- 데이터 커버리지 : 학습하고자 하는 대상에 대한 데이터 유형의 다양성이 충분한가를 나타내주는 척도
- 데이터 분포성 : 학습 데이터가 정규분포를 따르는지 확인하는 척도
- 데이터 완전성 : 학습 데이터 집합에 학습하고자 하는 대상의 모든 속성이 포함되어 있는가를 나타내주는 척도
- 데이터 중복성 : 학습 데이터 집합에 중복되는 데이터가 얼마나 포함되어 있는가를 나타내주는 척도
- 데이터 추적성 : 학습 데이터 집합으로 지능 소프트웨어 시스템이 학습을 진행한 후 새로운 데이터 입력에 대해 예상 결과와 다른 결과를 보여주었을 때 어떤 학습 데이터가 학습의 성능을 낮추는지 확인(추적)할 수 있는 척도

본 논문에서 소개하는 기준들은 해당 데이터의 속성들로 하여 그 평가 값이 이미 정의되어 있는, 즉 라벨링 된 데이터[6]를 학습에 사용하는 지도 학습 기반의 소프트웨어 개발 시 활용할 수 있다. 이를 활용하여 기계 학습을 진행 하기 전 소프트웨어의 품질이나 정확성을 예측해 볼 수 있고, 예측한 결과에 따라 학습을 진행 할지 혹은 학습 데이터를 재 구성할지 판단할 수 있다. 이는 기대 정확성이 낮은 데이터로 학습 하는 것을 방지해 시간, 비용 측면으로 낭비를 줄일 수 있으며 트랜드 변화 주기가 빠른 현재 시장 환경에서 개발 시간을 단축 시키는데 기여할 수 있다[3].

본 논문은 총 5장으로 구성되어 있다. 먼저 1장에서는 학습 데이터에 대한 품질 평가 방법이 필요한 이유에 대해 설명하고 이에 대한 새로운 기준을 소개한다. 2장에서는 기존 연구의 분석을 통해 데이터 평가 방법이 어떠한 것들이 제시되었는지 또 한계는 무엇인지 대해서 살펴본다. 3장에서는 본 논문에서 제시하는 학습 데이터 품질 평가 방법들에 대해 기술한다. 이를 위해 각각의 기준들의 척도를 제시하고 적용 예를 통해 어떻게 데이터들이 평가 될 수 있는지를 설명한다. 4장에서는 데이터 품질 평가를 위한 프로세스를 제시한다. 마지막 5장에서는 본 연구의 결론과 향후 연구에 대해서 기술한다.

2. 관련 연구

데이터의 품질이 소프트웨어의 학습 성능에 얼마나 영향을 미치는 가에 대해서 Corinna Cortes [19] 등에 의해 연구 된 바 있다. 해당 연구에서는 학습 데이터 중 노이즈가 포함된 데이터의 양이 증가할 때마다 학습 성능에 얼마나 영향을 미치는 지 계산하였고 계산된 수치를 통해 학습 데이터의 품질에 대한 중요성을 설명하였다. 데이터의 품질을 평가하기 위한 방법들은 기존의 여러 연구들에서 논의된 바 있다. Leo Pipino [17] 등은 기업으로부터 생산되는 데이터의 품질 척도를 표 1과 같이 16개의 차원으로 나누고, 각각의 항목에 대한 정의와 그 의미를 설명하였다. 해당 연구에서는 위의 데이터 품질 측정 기준에 대해 데이터의 품질 평가를 주관적, 객관적인 평가로 나누어서 진행하였다. 이를 매트릭으로 구성하여 객관적인 품질 평가 지수, 주관적인 품질 평가 지수를 합해 데이터의 최종 품질 평가 지수가 산정되도록 하였다. 해당 매트릭을 통해 고객들에게 공개되는 기업의 데이터 품질을 평가 할 수 있다고 설명하였다.

[표 1] 데이터 품질 차원[17]

Dimensions	Definitions
Accessibility	the extent to which data is available, or easily and quickly retrievable
Appropriate Amount of Data	the extent to which the volume of data is appropriate for the task at hand
Believability	the extent to which data is regarded as true and credible
Completeness	the extent to which data is not missing and is of sufficient breadth and depth for the task at hand
Concise Representation	the extent to which data is compactly represented
Consistent Representation	the extent to which data is presented in the same format
Ease of Manipulation	the extent to which data is easy to manipulate and apply to different tasks
Free-of-Error	the extent to which data is correct and reliable
Interpretability	the extent to which data is in appropriate languages, symbols, and units, and the definitions are clear
Objectivity	the extent to which data is unbiased, unprejudiced, and impartial
Relevancy	the extent to which data is applicable and helpful for the task at hand
Reputation	the extent to which data is highly regarded in terms of its source or content
Security	the extent to which access to data is restricted appropriately to maintain its security
Timeliness	the extent to which the data is sufficiently up-to-date for the task at hand
Understandability	the extent to which data is easily comprehended
Value-Added	the extent to which data is beneficial and provides advantages from its use

GIRRES [8] 등은 Open Street Map을 활용하여 해당 어플리케이션으로부터 나오는 데이터 집합을 평가하는 연구를 진행하였다. 해당 연구에서는 특정 어플리케이션에서 나오는 데이터에 대한 품질 평가를 통해 애플리케이션의 성능측면에서 신뢰성이 얼마나 되는지 판단하였다. 이러한 연구 외에도 여러 연구들에서 데이터 품질을 평가할 수 있는 여러 척도들을 제시한다. 하지만 앞선 연구들처럼 그 활용의 범위가 매우 구체적이며 한정적이다. 또한 품질 측정 결과를 해당 시스템의 안정성 및 신뢰성을 판단하는 용도로만 사용 하기 때문에 본 연구의 목적인 학습 데이터의 품질 평가 척도로 사용되기는 적합하지 못하는 문제가 있다. 따라서 본 논문에서는 기존에 제시된 여러 척도들 이외에 소프트웨어의 학습 성능을 예측할 수 있는 학습 데이터의 품질 평가 척도를 새롭게 제시한다.

3. 학습 데이터 품질 평가 척도

이 장에서는 학습 데이터 품질을 평가할 수 있는 척도를 제시한다. 데이터 평가 척도를 설명하기 앞서 본 논문에서는 해당 척도들이 지도 학습 기반의 지능 소프트웨어에서만 적용 된다는 것과 생성된 학습 데이터 집합은 라벨링된 데이터를 사용해야 한다는 것을 사전 조건으로 선정하였다. 이는 데이터의 라벨을 통해서 본 논문에서 제시하는 척도들을 계산할 때 필요한 특성들을 식별 할 수 있기 때문이다.

데이터 품질 평가 척도로는 데이터 커버리지, 데이터 분포성, 데이터 중복성, 데이터 완전성, 데이터 추적성 5개를 제시하였다. 이 중 데이터 추적성은 사후 데이터 품질 평가 기준으로, 나머지 4개의 항목에 대해선 사전 데이터 품질 평가 기준으로 구분하였다. 이는 해당 척도들이 소프트웨어의 학습 전후 어느 단계에서 적용되는지에 따른 기준으로 나누어 진다. 해당 내용에 대해 추후 3장에서 나누어진 데이터 품질 평가 기준 별 프로세스를 통해 자세히 설명한다.

3.1 데이터 커버리지 (Data Coverage)

데이터 커버리지는 '학습하고자 하는 대상에 대한 데이터 유형의 다양성이 충분한가'를 나타내주는 척도이다. 해당 기준이 필요한 이유는 소프트웨어가 특정 대상에 대해 학습하고자 할 때, 여러 유형의 데이터를 학습함으로써 어떠한 데이터가 입력 되더라도 서비스가 중단되지 않고 정확한 결과를 출력하기 위함이다. 여기서 커버리지란 말은 본래 소프트웨어의 테스트를 논할 때 얼마나 테스트가 충분 한가를 나타내주는 지표로 사용된다[4].

본 논문에서는 데이터가 학습을 하기 위해 얼마나 충분한지를 나타내기 위해 그 의미를 달리하였다. 본 논문에서 제시하는 데이터 커버리지 척도를 산정하기 위해서는 데이터 유형이 사전에 정의되어야 한다. 일반적으로 데이터의 분류는 크게 정상 데이터(Valid data)와 비정상 데이터(Invalid data)로 분류할 수 있다. 본 특성 하에서는 이를 좀 더 세분화하여 정상 범주의 데이터를 원본 데이터(Original Data), 유사 데이터(Similar data), 오류를 발생 시킬 수 있는 데이터(Adversarial Data)로 구분한다. 이와 같은 데이터 유형의 분류는 기계학습 기반 소프트웨어의 응용 영역에 대하여 추가 또는 삭제 될 수 있다. 다만 학습 데이터의 유형이 사전 정의되어야 하고 이에 따라 수집한 학습 데이터들이 사전 정의된 유형들로 나뉘어 질 수 있어야 한다.

데이터 커버리지는 이렇게 세분화된 학습 데이터의 유형에 적어도 하나 이상의 데이터가 존재 해야 한다는 것이다. 각각의 유형에 적어도 하나 이상의 데이터가 존재한다면 구성된 데이터는 모든 데이터 유형을 커버한다고 할 수 있다. 학습 데이터 커버리지를 산출하는 척도는 식 1과 같이 정의한다.

$$\text{데이터커버리지} = \frac{\text{데이터가존재하는유형의수}}{\text{정의된학습데이터유형의수}} \times 100\ (\%)$$

[식 1] 데이터 커버리지 산출 척도

식 1에서 정의 한 것처럼 데이터 커버리지는 데이터가 하나라도 존재하는 유형의 수를 전체 유형의 수로 나눈 것의 백분율로 표현한다. 예를 들어 하나의 이미지 분류기에서 사전에 정의된 데이터 유형이 10가지이고 준비된 학습 데이터의 유형이 8가지 일 때, 학습 데이터 커버리지는 80%의 데이터 커버리지를 갖는다 표현할 수 있다.

3.2 데이터 분포성 (Data Distribution)

데이터 분포성은 학습 데이터가 정규분포를 따르는지 확인하는 척도이다. 자연의 데이터 대부분은 정규분포를 이루지만, 학습 데이터를 수집할 때 그 수가 충분하지 않거나 편향된 데이터를 수집하는 경우 정규 분포를 따르지 않게 된다. 학습 데이터도 정규분포를 형성해야 기계 학습 기반의 소프트웨어를 학습 시킬 때 좋은 성능을 기대할 수 있다. 따라서 소프트웨어의 학습을 진행하기 전에 데이터의 정규 분포를 확인하고 데이터를 다시 구성할지 학습을 진행할지 판단할 수 있다. 데이터의 분포성을 나타내는 척도를 원본 데이터로부터의 거리를 기준으로 산정하는 표준 편차로 정의하였다. 이는 일반적인 통계 분석에서 정의하는 방법과 동일하게 식 2와 같이 표현 되었다.

$$\text{데이터 분포성} = \sqrt[\pm]{\frac{\Sigma(\text{각학습데이터와원본데이터와의거리})^2}{(\text{학습데이터의총수})-1}}$$

[식 2] 데이터 분포성 산출 척도

원본 데이터(중앙값)를 중심으로 학습 데이터가 표준 정규 분포를 따를 때, 여러 측면의 데이터 유형에 대한 학습 효과가 나타난다. 따라서 데이터 분포성은 표준 편차가 1에 가까울 수록 좋다고 할 수 있으며, 유의 수준은 95%로 정의한다. 학습하고자 하는 데이터가 정규 분포를 따르지 않거나, 학습하고자 하는 대상이 자연상태에서 매우 희박한 확률로 나타나는 경우라면 언더 샘플링과 오버 샘플링 방식을 통해 데이터 집합을 다시 구성할 수 있다[9].

3.3 데이터 완전성 (Data Thoroughness)

데이터 완전성은 학습 데이터 집합에 학습하고자 하는 대상의 모든 속성이 포함되어 있는가를 나타내주는 척도이다. 이 척도를 산정하기 위해서는 먼저 사용자는 학습하고자 하는 대상을 선정한다. 단순 이진 분류기라면 두 개에 대한 학습을 진행하면 되지만, 더 높은 차원의 분류기라면 여러 차원의 학습 대상을 선정한다[10]. 나눈 기준들을 토대로 학습 데이터 집합에 해당 속성을 나타낼 수 있는 데이터들이 포함되어 있는가를 판단하여 척도를 계산한다. 데이터 완전성에 대한 산정은 식 3과 같이 정의 한다.

$$\text{데이터 완전성} = \frac{\text{학습 데이터에 포함된 속성의 수}}{\text{학습 대상의 속성수}} \times 100\ (\%)$$

[식 3] 데이터 완전성 산출 척도

예를 들어, 이미지를 통한 물체 인식의 경우, 하나의 물체를 표현하기 위한 다양한 구조적 형상을 기준으로 전체적인 속성의 수가 정해진다. 보다 상세히 사람의 경우를 살펴보면 두 발, 두 손, 가슴, 등, 골반(엉덩이), 머리, 눈, 코, 입, 귀와 같은 속성들을 식별할 수 있으며, 학습 데이터 전체로부터 이러한 속성이 누락 없이 모두 포함되어 있는가를 확인하는 것이 학습 데이터의 완전성 속성이다. 만약 사람의 속성으로 정의된 것들이 학습 데이터에서 누락된 경우 이를 사람의 일부로 판단할 수 없게 된다.

앞서 나열한 속성들을 나누는 방법은 사람이 식별할 수 있는 속성들에 한정되어 있다. 하지만 기계 학습 기반의 소프트웨어에서는 사람이 식별할 수 있는 특징 이외에도 대상의 고유한 특징을 추출하고 그 중 대상의 특성을 가장 잘 나타내주는 특징을 선택하는 방식으로 대상에 대한 인식이 이루어진다[11, 12]. 이러한 작업을 수행하는 알고리즘을 통해 사람이 인지하지 못하는 속성들을 추가적으로 식별하여 완전성을 산정하는 식에 반영할 수 있다.

3.4 데이터 중복성 (Data Redundancy)

데이터의 중복성은 학습 데이터 집합에 중복되는 데이터가 얼마나 포함되어 있는가를 나타내주는 척도이다. 학습에 있어서 데이터가 중복되는 데이터가 많이 포함되어 있다면 이미 학습한 데이터를 다시 반복 학습하는 의미 없는 과정을 거치게 된다. 따라서 학습 효율 측면에서 좋은 결과를 기대하기가 힘들다[13]. 데이터에서 완전히 똑같은 데이터는 포함될 확률이 적지만 거의 유사한 데이터가 포함될 수 있는 확률은 높다. 이를 방지하기 위해 먼저 학습 대상의 속성을 가장 잘 보여주는 혹은 반드시 학습해야하는 속성을 지닌 데이터들을 선별하여 데이터 속성이 같은 것끼리 집합을 구성한다.

여기서 같은 데이터 속성 집합 이라는 것은 데이터가 포함하는 벡터 정보가 기준 데이터와 일정한 차이($\pm\alpha$) 이내에 존재하는 데이터들의 모임을 의미한다. 이후 비교하고자 하는 데이터와 선별한 데이터 집합에 속한 데이터들을 비교하여 유사도를 측정한다. 데이터의 속성들은 다차원의 형태를 가지기 때문에 단순 데이터간의 거리를 통해 계산하는 방식을 사용하는데 제한이 있다. 따라서 본 연구에서는 다차원의 데이터 유사도를 측정하는 방법으로 유클리디언 거리계산, 민코프스키 거리, 코사인 유사도 측정 등의 방법을 사용한다. 유클리디언 거리는 n차원의 공간에서 두 점 간의 거리를 알아내는 공식이다. 이를 데이터에 적용할 경우 거리가 가깝다면 유사도가 높다고 판단할 수 있다[16]. 코사인 유사도의 경우 두 벡터 간의 코사인 각도를 이용하여 두 벡터의 유사도를 구한다[17]. 이 역시 기계 학습에서 데이터의 유사도를 구할 때 많이 사용되는 방식이다. 따라서 이러한 방식들을 소프트웨어의 응용 영역에 따라 적합한 계산법을 채용해 유사도를 산출한다[14]. 데이터간의 유사도를 구하는 계산법은 식 4에서 $Sim(d_j, d_{ji})$로 표현 되며 전체 수식은 아래와 같다.

$$\text{데이터 중복성} = \frac{\sum_{j=1}^{m}\sum_{i=10}^{n} sim\,(d_j\,,\,d_{ji}\,)}{n_1+n_2+..+n_m}$$

[식 4] 데이터 중복성 산출 척도

식 4에서 n은 동일 유형에 속하는 학습의 데이터 개수이고, m은 유형의 수이다. 데이터의 유사도 $Sim(d_j, d_{ji})$는 유형 j에 속하는 데이터 d_j를 기준으로 유형 내의 모든 다른 데이터와의 유사도를 산출한 후 이들을 합한 값이다. 이를 전체 학습 데이터의 개수로 나누면

데이터의 중복성을 나타내는 척도가 된다. 산출된 데이터 중복성 값이 특정 임계치 보다 높게 산정되면, 유사도가 가장 높게 나온 데이터부터 삭제하고 데이터를 재구성해야 한다.

3.5 데이터 추적성 (Data Traceability)

데이터 추적성은 학습 데이터 집합으로 지능 소프트웨어 시스템이 학습을 진행 한 후, 새로운 입력 데이터에 대하여 예상 결과와 다른 결과를 보여주었을 때 어떤 데이터가 학습의 성능을 낮추는지 확인(추적)할 수 있는 척도를 말한다. 즉 올바른 데이터가 입력되는 경우 그에 따라 올바른 결과를 내야하고, 올바르지 못한 데이터가 입력되는 경우, 올바르지 못하다는 결과를 제시해야 한다. 그런데, 올바르지 않은 데이터를 입력하였음에도 불구하고, 올바르다고 판단하는 경우(false negative)와 혹은 올바른 데이터를 입력하였음에도 불구하고 올바르지 않다고 판단하는 경우(false positive), 이 두 가지 경우에는 학습 데이터에 문제가 있음을 예상할 수 있다. 따라서 문제의 원인이 어디서 비롯 되었는지를 학습 데이터로부터 추적할 수 있어야 한다. 앞서 설명한 네 가지의 학습 데이터 품질 평가 척도와 달리 이 품질은 사후 데이터 품질 평가 기준의 척도로 삼는다. 즉 기계 학습 기반의 앞선 사전 데이터 품질 평가 기준을 만족하는 데이터로 소프트웨어를 학습시켰을 때 다음과 같은 두 가지 경우가 발생하는 경우에만 데이터 추적성을 평가해야 한다.

- 올바른 입력 데이터에 대한 잘못된 결과 (false positive) : 이 경우에는 올바른 데이터에 대한 학습의 부족으로 인한 결과이다. 따라서 유사도가 높은 데이터를 추가로 생성 혹은 수집하여 학습 데이터에 포함시켜야 한다.
- 올바르지 못한 입력 데이터에 대한 올바른 결과 (false negative) : 이 경우는 학습의 오류에 해당되며, 올바르지 못한 입력 데이터와 유사도가 높은 학습 데이터를 제거해야 한다.

학습 데이터에 대한 추적성은 식 5와 같이 표현할 수 있으며, 이는 추적성 존재 유무에 대하여 바이너리 값으로 평가된다.

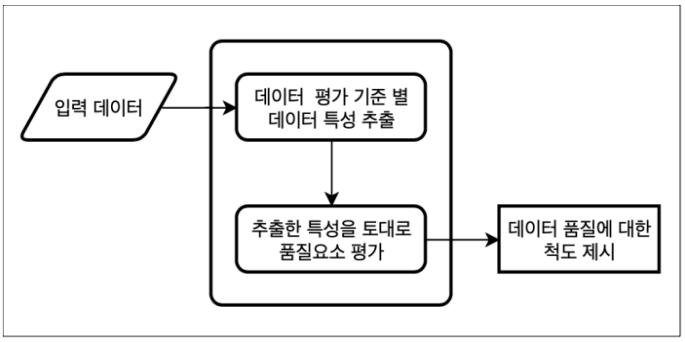
$$\text{데이터추적성} = \begin{cases} 1, & \text{if } \exists d \in D_L \; such that \; I_p \xrightarrow{\alpha} d \\ 0, & \text{if } \forall d \notin D_L \; such that \; I_p \xrightarrow{\alpha} d \end{cases}$$

[식 5] 데이터 추적성 산출 척도

식 5에 따르면 속성 p를 갖는 입력 데이터 I_p와 학습 데이터 집합 D_L의 원소 중 유사한 속성을 갖는 원소로 매핑(함수 α)된다면 추적성은 1의 값을, 그렇지 않은 경우는 0을 값으로 평가된다. 추적성의 값이 0으로 나타나는 경우는 학습 모델에 수정이 필요하게 된다.

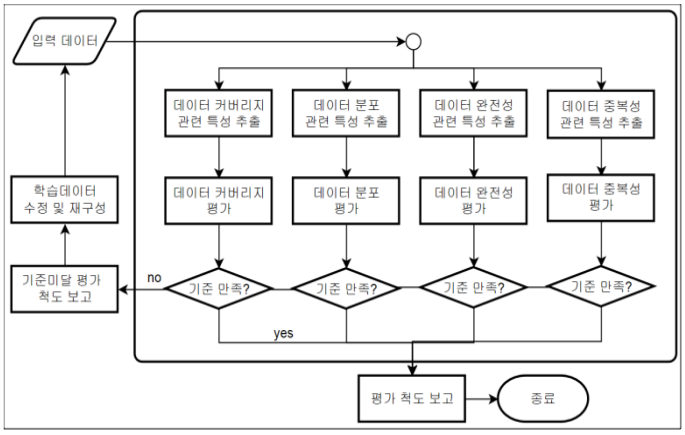
4. 학습 데이터 품질 평가 프로세스

이 장에서는 데이터 품질 평가 기준을 활용한 학습 데이터 품질의 평가 프로세스에 대해 기술한다. 학습 데이터 품질 평가 프로세스는 그림 1과 같다. 본 논문에서는 하나의 시스템내에서 해당 프로세스가 실행된다고 간주하고 그 과정에 대해서 기술한다.



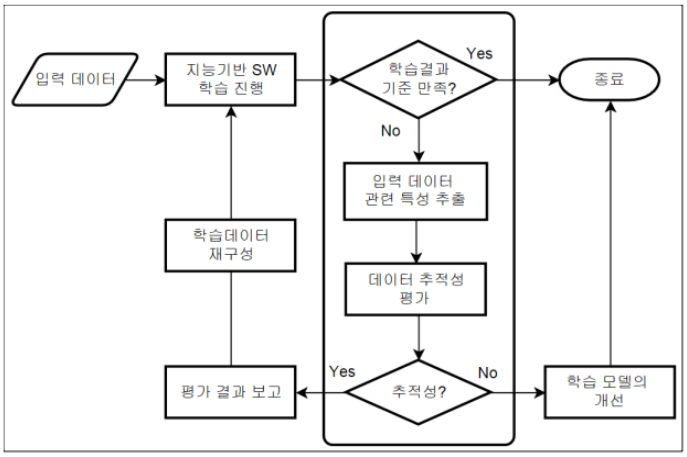
[그림 1] 학습 데이터 품질 평가 프로세스

그림 1과 같이 학습 데이터 품질 평가 시스템은 학습 데이터로 사용될 데이터들을 입력 데이터로 사용한다. 입력된 데이터들을 토대로 해당 시스템은 본 논문에서 제시하는 데이터 평가 기준 별 데이터 속성을 추출하는 과정과 추출한 속성을 토대로 품질 요소를 평가하는 두 과정으로 이루어진다. 각 품질 요소의 평과 결과들을 종합해 데이터의 품질을 평가할 수 있는 척도를 확인할 수 있다. 학습데이터에 대한 평가를 진행한 사용자는 학습 데이터를 그대로 사용할지 아니면 다시 구성할지 결정하는데 있어서 해당 결과를 토대로 결정할 수 있다.



[그림 2] 학습 데이터 품질 평가 세부 프로세스

그림 2는 그림 1의 학습 데이터 품질 평가의 세부 프로세스를 나타낸 것이다. 이는 앞서 제시한 5가지의 학습 데이터 품질 평가 척도 중 사전 데이터 품질 평가 척도인 4가지에 대한 프로세스를 나타낸다. 해당 데이터 품질 평가를 위해 시스템은 특성 추출 모델을 통해 평가 기준 별 특성을 추출한다. 추출한 특성들을 토대로 수치화 된 척도를 계산하고 사용자가 정의한 기준을 만족하는지 판별한다. 이후 결과를 종합하여 기준이 만족하는지, 아닌지 사용자에게 평가 척도를 보고하고 이를 사용자가 판단하여 데이터를 재구성할 것인지 학습을 진행할 것인지 결정한다.



[그림 3] 데이터 추적성 평가 프로세스

그림 3은 사후 학습 데이터 평가 기준인 데이터 추적성을 평가하기 위한 프로세스를 나타낸다. 앞서 사전 데이터 평가 척도를 통해 기준을 만족하는 경우 사용자는 학습 데이터를 통해 소프트웨어의 학습을 진행한다. 학습을 진행한 후 예상하는 결과가 나오지 않거나 소프트웨어의 성능이 기대 성능에 미치지 못한 경우 해당 추적 프로세스를 진행한다. 먼저 학습 데이터로부터 데이터 추적성 관련 특성을 추출한다. 올바르지 못한 결과를 도출하는 데이터를 찾고 해당 데이터의 데이터 추적성을 평가한다. 앞서 제시한 기준에서처럼 데이터 추적성의 바이너리 값이 0이 나오는 경우 소프트웨어의 학습 모델을 개선을 진행한다. 반대로 데이터 추적성 값이 1이 나오는 경우 해당 데이터를 학습 데이터 집합에서 삭제하고 다른 데이터를 추가하여 학습 데이터를 재구성 한다. 학습 데이터가 재구성 된다면, 데이터 집합의 성질이 달라지게 되는 경우가 발생할 수 있으며, 이 경우에는 사전 데이터 품질 평가를 재수행하게 된다.

5. 결론 및 향후 연구

본 논문에서는 기계학습 기반 소프트웨어의 학습 데이터의 품질을 평가할 수 있는 기준 다섯 가지를 제시하였다. 제시 된 기준들은 학습 데이터를 사용하는 지도 학습 기반의 소프트웨어를 개발할 시 활용 할 수 있다. 이러한 기준들을 통해 기계 학습을 진행 하기 전 소프트웨어의 품질이나 정확성을 예측해 볼 수 있고 예측한 결과에 따라 학습을 진행할지 데이터를 재구성할 지 판단할 수 있다. 이는 예측 정확성이 낮은 데이터로 학습하는 것을 방지 해 시간, 비용적으로 낭비를 줄일 수 있으며, 트랜드 변화 주기가 빠른 현재 시장 환경에서 소프트웨어의 개발 시간을 단축시키는데 기여할 수 있다. 또한 제시한 기준들을 활용하여 기계학습 기반의 소프트웨어에 사용되는 학습 데이터의 품질을 보장하고 궁극적으로 소프트웨어의 성능을 높이는데 기여할 수 있다. 특히 안전중심의 소프트웨어의 경우 어떠한 학습이 이루어졌는가가 의사 결정에 매우 중요한 영향을 미치기 때문에 학습 데이터의 사전 품질 점검은 매우 중요하다고 할 수 있다.

추후 연구 방향으로는 실제 데이터의 품질을 평가할 수 있는 학습 데이터 품질 평가 시스템을 개발하여 본 논문에서 제시한 기준들이 얼마나 효율적인 결과를 도출해내는지 확인하고자 한다. 또한 데이터 품질 측면에서 학습 데이터에 포함된 민감 정보 포함 여부 및 제거 방법에 대해서 연구가 추가적으로 진행 될 수 있을 것이다. 빅 데이터 시대에서 많은 정보들이 생겨나고 있지만 그 중 개인정보 등 민감한 정보를 포함하고 있는 데이터들도 다수 존재할 수 있기 때문에 데이터의 정보 보호 관점에서 이러한 민감 데이터를 확인하는 과정이 학습 또는 데이터 가시화 이전에 평가되어야 할 필요가 있다.

Acknowledgement

이 논문은 2019년도 과학기술정보통신부의 재원으로 한국연구재단 - 차세대정보컴퓨팅기술개발사업(NRF-2017 M3C4A7066479)의 지원을 받아 수행한 연구임.

참고 문헌

[1] LEVESON, Nancy G. “Software safety: Why, what, and how,” ACM Computing Surveys (CSUR), 18.2: pp.125-163, 1986.
[2] LISON, Pierre. An introduction to machine learning. Language Technology Group (LTG), 1, 35, 2015.
[3] BLACKBURN, Joseph D.; SCUDDER, Gary D.; VAN WASSENHOVE, Luk N. “Improving speed and productivity of software development: a global

survey of software developers," IEEE transactions on software engineering, 22.12: pp.875-885, 1996.
[4] MALAIYA, Yashwant K., et al. "The relationship between test coverage and reliability," Proceedings of 1994 IEEE International Symposium on Software Reliability Engineering. pp. 186-195, 1994.
[5] ZHANG, Zhongheng. "Introduction to machine learning: k-nearest neighbors," Annals of translational medicine, 4.11, 2016.
[6] XIAO, Tong, et al. "Learning from massive noisy labeled data for image classification," In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2691-2699 , 2015.
[7] PIPINO, Leo L.; LEE, Yang W.; WANG, Richard Y. "Data quality assessment," Communications of the ACM, 45.4: pp.211-218, 2002.
[8] GIRRES, Jean-François; TOUYA, Guillaume. "Quality assessment of the French OpenStreetMap dataset," Transactions in GIS, 14.4: pp.435-459, 2010.
[9] LUENGO, Julián, et al. "Addressing data complexity for imbalanced data sets: analysis of SMOTE-based oversampling and evolutionary undersampling," Soft Computing, 15.10: pp.1909-1936, 2011.
[10] LIU, Ting; MOORE, Andrew W.; GRAY, Alexander G. "Efficient Exact k-NN and Nonparametric Classification in High Dimensions," In: NIPS, pp. 265-272, 2003.
[11] KHALID, Samina; KHALIL, Tehmina; NASREEN, Shamila. "A survey of feature selection and feature extraction techniques in machine learning," IEEE 2014 Science and Information Conference, pp. 372-378, 2014.
[12] GUYON, Isabelle; ELISSEEFF, André. An introduction to feature extraction, Feature extraction. Springer, Berlin, Heidelberg, 2006.
[13] OHNO-MACHADO, Lucila; FRASER, Hamish S.; OHRN, A. "Improving machine learning performance by removing redundant cases in medical data sets," Proceedings of the AMIA Symposium. American Medical Informatics Association, p. 523, 1998.
[14] BORIAH, Shyam; CHANDOLA, Varun; KUMAR, Vipin. "Similarity measures for categorical data: A comparative evaluation," Proceedings of the 2008 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, pp. 243-254, 2008.
[15] Stuart Reid, Phd, FBCS, AI testing workshop, STA Consulting Inc, 2019.
[16] QIAN, Gang, et al. "Similarity between Euclidean and cosine angle distance for nearest neighbor queries," Proceedings of the 2004 ACM symposium on Applied computing. ACM, pp.1232-1237, 2004.
[17] NGUYEN, Hieu V.; BAI, Li. "Cosine similarity metric learning for face verification," Asian conference on computer vision. Springer, Berlin, Heidelberg, pp. 709-720, 2010.
[18] CAI, Li; ZHU, Yangyong. "The challenges of data quality and data quality assessment in the big data era," Data science journal, 14, 2015
[19] CORTES, Corinna; JACKEL, Lawrence D.; CHIANG, Wan-Ping. "Limits on learning machine accuracy imposed by data quality," Advances in Neural Information Processing Systems, pp. 239-246, 1995.
[20] KOTSIANTIS, Sotiris B.; ZAHARAKIS, Loannis D.; PINTELAS, Panayiotis E. "Machine learning: a review of classification and combining techniques," Artificial Intelligence Review, 26.3: pp.159-190, 2006.
[21] ADELI, Hojjat. Machine Learning-Neural Networks, Genetic Algorithms and Fuzzy ystems. *Kybernetes*, 1999.

행위 모델 기반 RESTful 웹 어플리케이션 결함 위치 추정

장종인 O1 이낙원 1 류덕산 2 백종문 1
1 한국과학기술원 전산학부 2 전북대학교 소프트웨어공학과
1{forestar0719, skrdnjs, jbaik}@kaist.ac.kr 2duksan.ryu@jbnu.ac.kr

Behavior Model-Based Fault Localization of RESTful Web Applications

Jong-In Jang O1 Nakwon Lee1 Duksan Ryu2 Jongmoon Baik1
1School of Computing, KAIST 2Department of Software Engineering, JBNU

요 약

웹 어플리케이션과 같은 복잡한 시스템에서 결함 위치 추정을 수행하는 기존 연구들은 시스템 내 구성 요소들 간의 간접적인 상호작용이 존재한다는 점과 시스템 자체가 동적으로 재구성될 수 있다는 점을 제대로 고려하지 못하고 있다. 이러한 한계점을 극복하기 위해 본 연구에서는 웹 어플리케이션의 동적인 특성과 간접 상호작용의 존재를 고려하여 보다 효과적인 결함 위치 추정 기법을 제안한다. 제안하는 기법은 RESTful 웹 어플리케이션의 실행 기록을 구성 요소들이 수행하는 행위들의 순열로 모델링하여 그 행위 모델 상에서 결함 위치 추정을 수행한다. 이 기법은 직간접적 상호작용 모두를 반영할 수 있는 행위 모델을 사용하고 행위 모델링 과정 자체도 시스템 실행 전에 모델을 미리 구축할 필요가 없어 동적 환경에 적합한 경량의 빠른 모델링을 가능케 하여 기존 기법들의 문제점을 해결한다. 기법의 평가를 위해 YouTube Data API v3를 기반으로 구축한 RESTful 웹 어플리케이션 대상의 사례 연구를 수행하여, 본 연구에서 제안하는 행위 모델 기반 RESTful 웹 어플리케이션 결함 위치 추정 기법이 대규모의 복잡한 웹 어플리케이션의 디버깅 노력을 효과적으로 감소시킬 수 있음을 보였다.

1. 서론

웹 어플리케이션은 전통적인 소프트웨어 시스템에 비해 복잡하고, 규모가 크며, 블랙박스 구성 요소를 가지고 있을 확률이 높기 때문에 기존의 결함 위치 추정 기법을 적용하기가 어렵다[1]–[3]. 복잡도나 규모 등의 웹 어플리케이션 결함 위치 추정을 어렵게 하는 요인 외에도 기존의 위치 추정 기법 적용을 방해하는 두가지 요인이 더 존재한다. 현재까지의 웹 어플리케이션을 추상화된 구조적 모델로 모델링한 다음 해당 모델 상에서 결함의 위치를 추정하는 연구들[1]–[7]에서 상대적으로 덜 다루어진 이들 요인은 다음과 같다. 첫번째, 웹 어플리케이션 시스템의 구성 요소들 간에 간접적인 상호작용이 존재한다는 것[7]–[9], 그리고 두번째, 웹 어플리케이션은 동적이고 유연하기 때문에 자주 재구성된다는 것이다[10]–[12].

웹 어플리케이션의 구조적 모델에서는 직접적인 연결이 없는 것으로 나타나는 구성 요소들 사이에서도 실제로는 공유 리소스(resource)의 수정 및 관측을 통해 서로 영향을 미치는 간접적 상호작용이 존재할 수 있다. 웹 어플리케이션 구성 요소들은 특정 오퍼레이션(operation)들을 수행함으로써 웹 상의 리소스를 관측하거나 수정한다. 이러한 오퍼레이션들은 웹 어플리케이션 구성 요소가 리소스에 어떻게 영향을 미치고, 또 그로부터 어떻게 영향을 받는지 분석할 수 있는 본질적인 매개체이며, 본 논문에서는 앞으로 이를 웹 어플리케이션의 "행위(behavior)"라 칭한다. 웹 어플리케이션 시스템과 같이 구성 요소들이 동일한 리소스나 환경을 공유하는 경우, 이러한 간접 상호작용의 중요성은 매우 크다[7]–[9].

웹 어플리케이션은 동적으로 재구성되는 특성을 가지기 때문에 시스템 작동 전에 미리 만들어진 분석 결과나 모델을 이용해서 결함 위치 추정을 수행하기 어렵다. 웹 어플리케이션은 실제로 배치되는 순간에 이르러서야 실재하는 웹 서비스들과 그들 사이의 연결로 실체화된다[11]. 따라서 사전에 생성된 시스템 모델에 크게 의존하지 않는 경량 모델링 및 분석 기술이 필요하다.

본 연구의 목표는 웹 어플리케이션의 동적인 특성과 간접 상호작용의 존재를 모두 고려하는 결함 위치 추정 기술의 개발이다. 제안하는 기법은 최근 웹 서비스 및 어플리케이션 개발 스타일의 대표격인 RESTful(REpresentational State Transfer-ful)[13] 웹 서비스로 구성된 웹 어플리케이션 을 타겟으로 한다. RESTful 웹 서비스 아키텍처 스타일은 웹 어플리케이션이 리소스들을 조작하는 행위에 대한 풍부한 정보를 제공한다. 이 정보를 통해 RESTful 웹 어플리케이션의 실행 기록은 행위의 순열로 모델링 되고 이를 본 논문에서는 "행위 모델(behavior model)"이라 한다.

행위 모델 상에서 실제로 결함 위치 추정을 수행하기 위해서 본 기법은 스펙트럼 기반 결함 위치 추정 기술[14] 을 사용한다. 결함 위치 추정의 세분성(granularity) 단위는 순서를 고려한 한 쌍(pair)의 행위로 선정하였다. 스펙트럼 분석에 사용되는 커버리지 정보와 실행 결과는 실행된 RESTful 웹 어플리케이션의 행위 모델과 해당 어플리케이션 실행의 성공 혹은 실패 여부가 된다. 제안하는 기법의 평가를 위해서 YouTube Data API v3[15]를 사용한 사례 연구를 수행했다.

* 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2017M3C4A7066212 & NRF-2019R1G1A1005047)

2. 행위 모델 기반 RESTful 웹 어플리케이션 결함 위치 추정

2.1. 행위 모델링

본 기법에서 RESTful 웹 어플리케이션의 실행 기록은 그 어플리케이션이 수행한 행위들의 순열로 표현된다. RESTful 웹 서비스 아키텍처 스타일에 따라 개발자 문서 등의 산출물(artifact)에 명시된 조작 대상이 되는 리소스와 그 리소스에 행해지는 등록, 수정, 삭제, 조회 등의 표준적인(standard) 오퍼레이션의 쌍들이 바로 그 행위들이다. 실행된 행위들은 웹 어플리케이션 실행 로그로부터 추출하거나 모니터링을 통해 파악할 수 있다.

표 1. YouTube Data API 기반 RESTful 웹 어플리케이션의 실행 기록 행위 모델

실행 기록 행위 모델	Pass/Fail
[17,21,0,17,10,21,10,32,32,0]	Pass
[27,17,0,32,27,1,2,18,18,28]	Pass
[28,17,1,21,0,28,0,20,10,1]	Pass
[0,2,5,15,7,16]	Fail
[0,1,2,19,19]	Fail
[17,1,20,18,27,18,28,21,19,27]	Pass
[28,21,30,27,10,17,10,17,1,30]	Pass
[1,0,0,21,10,28,0,0,19,30]	Pass
[21,10,0,20,20,28,0,28,21,28]	Pass
[28,21,31,1,30]	Fail

위의 표 1은 사례 연구에서 활용한 YouTube Data API 기반 RESTful 웹 어플리케이션의 실행 기록들을 행위 모델로 변환한 결과를 보여준다. 각 실행 기록들은 각 상황마다 주어진 리소스와 조건에 따라 임의의 가용한 리소스에 임의의 호출 가능한 오퍼레이션을 실패가 발생하기 전까지 순차적으로 최대 10번 실행한 결과이다. 따라서 이 실행 기록들은 최대 10개의 행위(0부터 32의 정수로 총 33개의 각 행위들을 명명)들의 순열로 나타나는 행위 모델로 표현된다.

3장의 사례 연구에서 RESTful 웹 어플리케이션에서 행위 정보를 수집하는 방법과 실행 기록 행위 모델링 및 실행 기록 확보 방법에 대해 더 자세히 다룬다.

2.2. 행위 모델 상의 스펙트럼 기반 결함 위치 추정

본 기법의 결함 위치 추정의 세분성 단위는 행위의 쌍이다. 특정한 두 개의 행위가 함께 실행되었을 때, 즉 특정 상호작용이 발생했을 때 결함이 발생할 수 있다고 보는 것이다. 한 쌍의 행위는 구성 요소들 간의 직접적 또는 간접적 상호작용을 의미한다. 행위의 쌍을 결함 위치 추정의 세분성 단위로 사용함으로써 개별 구성 요소들에는 결함이 없지만 그 구성 요소들 간의 상호작용에서만 비로소 발생하는 결함들도 찾아낼 수 있게 된다[16]–[18].

각 행위 쌍들의 커버리지 정보는 실행 기록으로부터 얻어진다. 하나의 행위 모델에 함께 존재하는(순서를 고려한) 두 행위가 있을 때, 그 두 행위의 순서쌍이 이 모델에 해당하는 실행이 커버한 행위 쌍으로 식별된다. 예를 들어, 표 1과 같은 10개의 순열을 획득했을 때 행위 쌍 (0,19)는 성공(pass)한 실행과 실패(fail)한 실행 모두에서 커버한 행위 쌍, (7,16)은 실패한 실행에서만 커버한 행위 쌍, 그리고 (2,3)은 어떤 실행에서도 커버하지 못한 행위 쌍이다.

이러한 스펙트럼의 누적을 통해 실패한 웹 어플리케이션 실행에 제일 많이 포함되었고, 성공한 실행에 제일 적게 포함된, 따라서 가장 의심스러운 행위 쌍들을 도출할 수 있다. 커버리지 정보로부터 의심도 점수를 계산하는 공식에는 여러가지가 있으며 본 기법에도 다양한 공식을 그대로 적용할 수 있으나, 본 논문의 사례 연구에서는 가장 널리 사용되는 의심도 점수 계산 공식 Tarantula[19]를 사용했다.

3. 사례 연구

3.1. 사례 연구 설정

사례 연구에 사용된 RESTful 웹 어플리케이션은 YouTube Data API v3(이하 YouTubeAPI)를 사용하는 웹 어플리케이션이다. 이 웹 어플리케이션은 YouTubeAPI가 제공하는 API들을 해당 API를 호출할 조건이 만족될 때마다 임의로 호출하는 어플리케이션이다.

RESTful 웹 어플리케이션이 실행 중에 http 에러가 반환되는 경우, 해당 웹 어플리케이션이 실패했다는 것을 감지하고 그 실행 기록과 행위 모델에 실해 실행 결과를 부여한다.

타겟 어플리케이션의 가용한 행위들의 목록은 YouTubeAPI에서 제공하는 리소스 목록과 그 리소스들에 대한 오퍼레이션의 목록으로부터 쉽게 얻을 수 있다. 이 리소스-오퍼레이션 목록은 YouTubeAPI의 개발자 안내서[15]를 참조하여 정의되었다. 본 사례 연구에서는 YouTubeAPI에서 가용한 33개 행위 전체를 구현했다.

3.2. 행위 모델 기반 결함 위치 추정

하나의 실행 기록은 행위를 10번 호출한 결과를 보여주며, 총 10개의 실행 기록을 수집했다. 수집된 실행 기록들의 행위 모델과 실행 결과는 표 1에 나타나 있다.

실행 기록 행위 모델로부터 얻을 수 있는 행위 순서쌍들에 대한 의심도 점수를 계산했다. 본 사례 연구는 33개의 행위들을 구현하여 수행되었으므로 존재할 수 있는 가능한 모든 행위 순서쌍의 수는 총 1,089개(동일 행위 반복 순서쌍 허용)이다. Tarantula 의심도 계산 공식을 통해 표 1의 총 10개의 실행 기록으로부터 알 수 있는 각 행위 순서쌍들의 의심도를 계산한 결과, 아래 표 2와 같이 20개의 행위 쌍들이 1.0의 제일 높은 의심도 점수를 가지는 것으로 나타났다.

표 2. 의심도 점수 상위 20개의 행위 순서쌍(행위 번호에 따른 내림차순 정렬)

행위 순서쌍	각 행위 순서쌍의 대상 리소스와 오퍼레이션	의심도 점수
(31,30)	(chSec-D),(chSec-Pu)	1.0
(31,1)	(chSec-D),(pList-G)	1.0
(28,31)	(chSec-G),(chSec-D)	1.0
(21,31)	(Actv-G),(chSec-D)	1.0
(19,19)	(pList-D),(pList-D)	1.0
(15,16)	(cmt-P),(cmt-Pu)	1.0
(15,7)	(cmt-P),(cmtThd-D)	1.0
(7,16)	(cmtThd-D),(cmt-Pu)	1.0
(5,16)	(cmtThd-P),(cmt-Pu)	1.0
(5,15)	(cmtThd-P),(cmt-P)	1.0
(5,7)	(cmtThd-P),(cmtThd-D)	1.0
(2,19)	(pListIt-G),(pList-D)	1.0
(2,16)	(pListIt-G),(cmt-Pu)	1.0
(2,15)	(pListIt-G),(cmt-P)	1.0
(2,7)	(pListIt-G),(cmtThd-D)	1.0
(2,5)	(pListIt-G),(cmtThd-P)	1.0
(0,16)	(channel-G),(cmt-Pu)	1.0
(0,15)	(channel-G),(cmt-P)	1.0
(0,7)	(channel-G),(cmtThd-D)	1.0
(0,5)	(channel-G),(cmtThd-P)	1.0

실제로 실패를 유발한 상호작용은 표 2에서 음영 처리로 강조되어 있다. 위 표에서 볼 수 있듯이 이 세 상호작용들은 모두 제일 높은 의심도 점수 1.0을 부여받았으므로 개발자는 전체 가능한 행위 순서쌍 1,089개의 불과 약 1.8%에 해당하는 20개의 가장 의심스러운 순서쌍만 검사하여도 실제 결함 상호작용 세 개를 모두 찾아낼 수 있는 것이다. 또한 무작위로 행위 순서쌍을 하나씩 선택하여 검사하는 경우, 실패를 유발한 세 개의 행위 순서쌍을 모두 찾을 때까지 검사해야 하는 순서쌍 수의 기댓값은 817.5개이다. 이는 본 논문에서 제안하는 행위 모델 기반 결함 위치 추정 기법을 적용했을 때 검사해야 하는 행위 순서쌍의 수인 20개의 약 41배에 가까운 개수로 본 기법의 결함 위치 추정 효과성을 반증한다.

4. 결론 및 향후 연구 방향

본 연구는 웹 어플리케이션 내의 간접 상호작용의 존재와 웹 어플리케이션의 동적인 특성 때문에 기존의 결함 위치 추정 기술들이 보이는 한계점을 극복하기 위해 행위 모델 기반 결함 위치 추정 기술을 개발하였다. 이 기술은 웹 어플리케이션을 어플리케이션에서 실행된 행위를 기준으로 모델링하고 그 행위 모델 상에서 스펙트럼 기반 결함 위치 추정을 수행하는 기법을 포함한다. 또한 YouTube Data API v3를 기반으로 구현한 RESTful 웹 어플리케이션을 대상으로 진행한 사례 연구를 통해 제안한 기법의 유용성을 보였다.

후속 연구로 널리 사용되고 있는 RESTful 웹 어플리케이션들의 실제 결함들을 본 기법으로 위치 추정하는 실험과 RESTful 웹 어플리케이션 결함 위치 추정에 활용될 수 있는 다른 기법들과의 위치 추정 효과성 및 간접 상호작용 위치 추정 능력과 잦은 동적 재구성 대응 능력 등의 성능 비교 분석 실험을 수행하여 제안하는 기법의 유용성을 더욱 엄밀하게 검증하고자 한다.

5. 참고문헌

[1] M. Kim, R. Sumbaly, S. Shah, M. Kim, R. Sumbaly, and S. Shah, "Root cause detection in a service-oriented architecture," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 1, p. 93, Jun. 2013.

[2] Q. He *et al.*, "Localizing Runtime Anomalies in Service-Oriented Systems," *IEEE Trans. Serv. Comput.*, vol. 10, no. 1, pp. 94–106, Jan. 2017.

[3] E. Kiciman and A. Fox, "Detecting Application-Level Failures in Component-Based Internet Services," *IEEE Trans. Neural Networks*, vol. 16, no. 5, pp. 1027–1041, Sep. 2005.

[4] X. Huang, S. Zou, W. Wang, and S. Cheng, "Fault management for Internet Services: Modeling and Algorithms," in *2006 IEEE International Conference on Communications*, 2006, pp. 854–859.

[5] S. Kandula *et al.*, "Detailed diagnosis in enterprise networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, p. 243, Aug. 2009.

[6] A. J. Oliner, A. V. Kulkarni, and A. Aiken, "Using correlated surprise to infer shared influence," in *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, 2010, pp. 191–200.

[7] D. Keil and D. Goldin, "Indirect Interaction in Environments for Multi-agent Systems," Springer, Berlin, Heidelberg, 2006, pp. 68–87.

[8] D. Goldin and D. Keil, "Toward domain-independent formalization of indirect interaction," in *13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 393–394.

[9] D. Keil and D. Goldin, "Modeling indirect interaction in open computational systems," in *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*, pp. 371–376.

[10] S. Artzi, J. Dolby, F. Tip, and M. Pistoia, "Fault Localization for Dynamic Web Applications," *IEEE Trans. Softw. Eng.*, vol. 38, no. 2, pp. 314–335, Mar. 2012.

[11] K. Bennett, P. Layzell, D. Budgen, P. Brereton, L. Macaulay, and M. Munro, "Service-based software: the future for flexible software," in *Proceedings Seventh Asia-Pacific Software Engeering Conference. APSEC 2000*, pp. 214–221.

[12] E. Di Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl, "A journey to highly dynamic, self-adaptive service-based applications," *Autom. Softw. Eng.*, vol. 15, no. 3–4, pp. 313–341, Dec. 2008.

[13] R. Fielding and R. Taylor, "Architectural styles and the design of network-based software architectures," *Dr. Diss. Univ. Calif.*, vol. 7, 2000.

[14] R. Abreu, P. Zoeteweij, R. Golsteijn, and A. J. C. van Gemund, "A practical evaluation of spectrum-based fault localization," *J. Syst. Softw.*, vol. 82, no. 11, pp. 1780–1792, Nov. 2009.

[15] YouTube, "YouTube Data API (v3), Getting Started." [Online]. Available: https://developers.google.com/youtube/v3/getting-started.

[16] M. Burger and A. Zeller, "Replaying and isolating failing multi-object interactions," in *Proceedings of the 2008 international workshop on dynamic analysis held in conjunction with the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2008) - WODA '08*, 2008, p. 71.

[17] J. Zhang, F. Ma, and Z. Zhang, "Faulty Interaction Identification via Constraint Solving and Optimization," Springer, Berlin, Heidelberg, 2012, pp. 186–199.

[18] E. Kıcıman and L. Subramanian, "A root cause localization model for large scale systems," in *Proc. Proceedings of USENIX Hot Topics On Dependability, Hot-Dep*, 2005.

[19] V. Dallmeier, C. Lindig, and A. Zeller, "Lightweight Defect Localization for Java," Springer, Berlin, Heidelberg, 2005, pp. 528–550.

멀티 응용의 실시간 성능 분석을 위한 스케줄 기반 SDF (Synchronous Dataflow) 모델 변환 기법

정도환 ○, 김장률, 하순회*
서울대학교 컴퓨터공학부
{dowhancool, urmydata, sha}@ snu.ac.kr

Scheduling-based SDF Model Transformation Technique for Real-Time Performance Analysis of Multiple Applications

Dowhan Jeong ○, Jangryul Kim, Soonhoi Ha*
Department of Computer Science and Engineering, Seoul National University

요 약

임베디드 시스템에서 실제 수행 전에 주어진 제약조건을 충족하는지 확인하는 정적 분석이 시스템 안정성 측면에서 매우 중요하다. 이 때문에 정적 분석이 가능한 SDF(Synchronous Dataflow) 모델을 이용하여 임베디드 시스템을 설계하는 방법론이 제안되었다. 하지만 최악 응답 시간을 정적으로 분석하기 위해서는 SDF 모델을 정적 분석 도구가 요구하는 태스크 그래프 모델로 변환할 필요가 있다. 그러나 SDF 모델을 단순 변환하면 태스크 그래프의 크기가 기하급수적으로 커지고 분석 시간이 따라서 급증하게 된다. 한편 임베디드 시스템의 효율성을 위해서는 SDF 그래프를 멀티 프로세서에 어떻게 매핑 및 스케줄링 할지 결정하는 설계 공간 탐색도 중요하다. 만일 그래프 변환 시 설계 공간 탐색을 고려하지 않고 그래프를 변환하면 비효율적인 매핑 및 스케줄링으로 유도할 수 있다. 이에 따라서 본 연구에서는 그래프 변환 시 그래프의 크기는 줄이나 설계 공간의 크기를 줄이지 않으면서 임베디드 시스템 설계 시 성능 분석은 빠르게 하는 새로운 기법을 제시한다. 실험을 통해서 제안하는 방법이 효과적으로 그래프 크기를 줄이며 성능 분석 시간을 획기적으로 감소시킴을 확인하였다.

1. 서 론

최근 계산량과 자원 요구량이 큰 딥 러닝 기반 응용을 수행하기 위해서 멀티 코어 CPU와 GPU 뿐만 아니라 뉴럴 프로세서라고 불리는 딥 러닝 가속기를 탑재한 이종 멀티 프로세서 임베디드 시스템이 등장하고 있다. 그러나 이러한 시스템 위에서 멀티 프로세서의 장점을 활용하여 병렬성을 갖는 소프트웨어를 개발하는 것은 어려운 문제이다. 더욱이, 임베디드 시스템에서는 프로세서 개수, 메모리 크기, 배터리 용량 등의 자원 제한 조건과 응용의 종료 시한(Deadline), 대역폭(Throughput) 등의 실시간 제한 조건을 갖고 있다. 예를 들어 자율 주행 자동차에서 수행되는 여러 딥 러닝 응용들은 제한된 자원을 공유하며 동시에 수행되고 일반적으로 종료 시한 조건이 주어진다. 만일 하나의 응용이라도 주어진 종료 시한 내에 수행 완료되지 못하면 치명적인 결과가 발생한다. 이와 같이 정확성이 보장된 임베디드 소프트웨어를 개발하는 것은 매우 중요하지만 어려운 문제이다. 이 때문에 임베디드 시스템의 실제 수행 전에 주어진 제약조건을 충족하는지를 확인하는 정적 분석(Static Analysis)이 중요한 주제이다. 그러므로 임베디드 시스템 설계 시 병렬성을 표현하기 용이하고, 정적 분석이 가능한 동기식 데이터플로우(Synchronous Dataflow, SDF) [1] 모델이 제안되었다.

반면 여러 응용이 수행될 때 최악 응답 시간(Worst Case Response Time)을 정적으로 예측하는 성능 분석 기법들이 연구되어 왔다 [2, 3]. 이러한 성능 분석 기법들은 SDF 그래프 중 특수한 경우인 HSDF(Homogeneous Synchronous Dataflow) 그래프 또는 DAG(Directed Acyclic Graph)를 입력 태스크 그래프로 받는다 [4]. 그러므로 최악 응답 시간을 분석하기 위해서는 SDF 그래프로 명세된 응용을 HSDF 그래프 혹은 DAG로 변환해야 한다. 모든 SDF 그래프는 HSDF 그래프로 변환이 가능하지만, 단순한 변환은 그래프의 크기와 성능 분석 시 소요되는 시간을 크게 증가시킨다 [1, 2].

한편, SDF 그래프로 명세 된 응용은 멀티 프로세서로의 매핑 및 스케줄링에 따라서 성능이 매우 달라진다. 예를 들어서 모든 응용이 하나의 프로세서에

※ 본 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 중견연구사업임(NRF-2019R1A2B5B02069406).
* 교신 저자(Corresponding author)

매핑되어 수행되는 경우와 여러 프로세서에 적절히 분배되어 수행되는 경우는 성능 차이가 극명하다. 그러므로 시스템 설계 시 매핑 및 스케줄링을 어떻게 구성할지 결정하는 설계 공간 탐색도 중요한 주제이다. 문제는 SDF 그래프 변환 시 그래프 크기를 감소하면 설계 공간도 같이 감소될 수 있다는 점이다. 설계 공간의 감소는 분석 시간이 감소한다는 점에서는 좋으나 최적의 해답을 찾는 문제에서는 비효율적인 매핑 및 스케줄링으로 유도할 수 있는 단점이 있다. 따라서 설계 공간의 크기를 감소시키지 않으면서 그래프의 크기를 줄이는 방법이 필요하다.

본 연구에서는 SDF 그래프로 명세된 응용들이 임베디드 시스템 위에서 실시간 제한 조건을 만족하는지 여부를 성능 분석기를 사용하여 검증하기 위해서 HSDF 그래프로 변환할 때 그래프의 크기가 기하급수적으로 커진다는 점에 착안하여, 매핑 및 스케줄을 기반으로 그래프 변환 시 그래프의 크기를 줄이는 기법을 제시한다. 이는 기존 연구들[5, 6, 7]과 달리 설계 공간 탐색을 고려하여 모두 매핑 및 스케줄을 기반으로 SDF의 그래프의 크기를 줄인다.

2. 배경 지식

2.1 SDF 그래프

임베디드 시스템 설계를 위해 병렬 수행을 가능케 하고 주어진 제약 조건을 만족하는지를 검증할 수 있도록 응용 프로그램을 데이터플로우(Dataflow) 형태의 정형적 모델로 표현하는 소프트웨어 설계 방법론이 제안되었다. 정형적 모델 중 하나인 SDF 모델은 응용을 프로세서에 매핑 가능한 단위인 태스크로 나누고, 태스크와 태스크 사이의 통신을 채널로 명세하는 분석 가능한 모델이다. SDF 모델은 계산량이 많은 과학계산 응용과 음성이나 화상을 처리하는 디지털 신호처리 응용에 적합한 모델로 알려져 있다 [1]. SDF 모델은 교착 상태나 버퍼 오버플로우와 같은 프로그램 오류를 사전에 탐지할 수 있고 태스크의 매핑과 스케줄링을 정적으로 결정함으로써 응용의 수행 성능을 사전에 예측할 수 있다는 장점이 있다. 그림 1은 SDF 그래프의 예시이다.

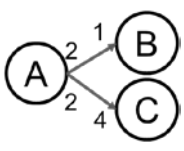


그림 1. SDF 그래프 예시

SDF 그래프의 각 노드는 태스크를 의미한다. 간선은 연결된 태스크들 간의 선입선출(FIFO) 통신 채널로 출발지에 해당하는 노드가 데이터 샘플을 생성하고 도착지에 해당하는 노드가 데이터 샘플을 소모한다. 간선 위의 숫자는 노드가 한번 수행될 때 생성되고 소모되는 데이터 샘플의 수를 나타낸다. 각 노드는 모든 입력 채널에 충분한 샘플이 있을 때 수행 가능하다. 예를 들어서 그림 1에서 노드 Ⓐ에서는 데이터 샘플 2개를 생성하고 노드 Ⓑ에서는 데이터 샘플 1개를 소모한다. 다시 말해서 노드 Ⓐ가 1번 수행되면 노드 Ⓑ가 2번 수행될 수 있다. 노드 Ⓐ가 2번 수행된 이후 노드 Ⓑ와 Ⓒ는 병렬적으로 수행될 수 있다.

각 노드 별 수행 횟수는 데이터 샘플 생성/소모 비율의 역수를 통해 구할 수 있는데, 그림1의 예에서 샘플 생성/소모 비율의 역수가 각각 Ⓐ:Ⓑ가 1:2, Ⓐ:Ⓒ가 2:1 이므로 세 노드의 데이터 샘플 생성/소모 비율의 역수는 Ⓐ:Ⓑ:Ⓒ가 2:4:1이 된다. 이 값을 노드의 반복 횟수(Repetition Count)라고 한다. 응용의 한 주기 수행은 모든 노드가 각각의 반복 횟수만큼 수행됨을 의미한다. 또한, SDF 그래프에서 각 노드는 프로세서로의 매핑 및 스케줄링의 단위이다. 각 노드의 매핑이 정해진 경우 노드들은 매핑된 프로세서 위에서 데이터 샘플로 인한 의존성(Dependency)을 지키면서 수행되며, 같은 프로세서에 매핑된 우선 순위가 높은 다른 노드에게 밀릴 수 있다.

2.2 SDF 그래프를 HSDF 그래프로 변환

SDF 그래프에서 모든 간선에서의 데이터 샘플 생성/소모 비율이 1:1인 그래프를 HSDF(Homogeneous Synchronous Dataflow) 그래프라고 한다. 모든 간선에서의 샘플 생성/소모 비율이 동일하므로 HSDF 그래프에서 모든 노드의 반복 횟수는 1이다.

모든 SDF 그래프는 동치인 HSDF 그래프로 변환할 수 있다. 가장 쉬운 변환 방법은 각 노드를 자신의 반복 횟수만큼 복제하는 방법이다. 그림 2는 그림 1의 SDF 그래프에서 노드 Ⓐ, Ⓑ, Ⓒ를 각각의 반복 횟수인 2, 4, 1만큼 복제하여 만든 HSDF 그래프이다.

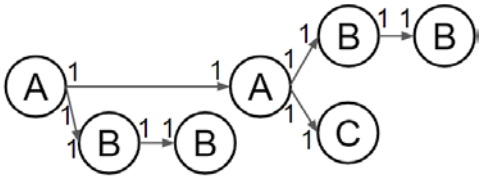


그림 2. 그림1과 동치인 HSDF 그래프

그러나 이러한 단순 변환 방법은 그래프의 크기를 기하급수적으로 증가시킨다는 단점이 있다. 특히 이는 데이터 샘플 생성/소모 비율에 크게 영향을 받는데, 만일 그림 1에서 노드 A와 B의 데이터 생성/소모 비율이 2:1 대신 2:3이라면 노드 Ⓐ, Ⓑ, Ⓒ의 반복 횟수는 (6, 4, 3)이 되어 총 노드 개수가 13개인 HSDF 그래프가 생성된다.

2.3 성능 분석

임베디드 시스템에서는 같은 자원을 공유하며 실행되는 여러 응용이 실시간 제약 조건을 만족하면서 수행 가능한지 여부가 중요하다. 이를 위해서 실제 수행 전에 각 응용의 최악 응답 시간을 분석하여 종료 시한(Deadline) 내에 모든 응용이 종료될 수 있는지를 판단한다. 이는 최악 응답 시간을 예측해서 종료 시한과 비교함으로써 판단한다. 다만, 여러 응용이 동시에 수행되는 경우 여러 응용 간의 간섭에 의한 영향을 고려해야 하므로 정확한 최악 응답 시간을 구하는 것은 매우 어려운 문제이므로 최악응답시간의 상한값을 최소화 시키려는 노력이 경주되고 있다.

실시간 시스템 분야에서 최악 응답 시간을 구하는 여러 연구들이 진행되어 왔으며, 그 예로는 HPA [2], STBA [3], SymTA/S [4] 등이 있다. 그러나 이러한 성능 분석 기법들은 응용을 독립적인 태스크, HSDF 그래프 또는 그 부분 집합인 DAG로 입력 받으며, 입력 그래프의 크기가 커짐에 따라 최악 응답 시간 계산 시 소요되는 시간이 비례하여 커진다. 따라서 빠른 성능 분석을 위해서는 SDF 그래프 변환 시 노드 개수를 줄이는 것이 중요하다. 또한, 정확성 측면에서는 원 SDF 그래프와 동일한 최악 응답 시간을 가지게 하는 것도 중요하다.

2.4 설계 공간 탐색

앞서 서술한 바와 같이 임베디드 시스템에서는 주어진 제약조건 충족 여부가 중요하며 이는 수행 전에 미리 최악 응답 시간 분석을 통해서 확인해야할 필요가 있다. 더 나아가서 최적의 매핑 및 스케줄링을 구해서 제한된 자원을 여러 응용이 공유하는 임베디드 시스템에서 효율적으로 자원을 사용할 필요가 있다. 이러한 과정을 설계 공간 탐색이라고 하며 이 문제는 멀티 프로세서 위에서 최적의 매핑 및 스케줄링을 찾는 문제이므로 NP-hard 문제이다 [8]. 그림 3은 하드웨어 플랫폼과 독립적으로 명세된 태스크 그래프 기반의 응용 명세를 프로세서에 매핑을 함으로써 설계 공간을 탐색하는 방법의 절차를 나타낸다.

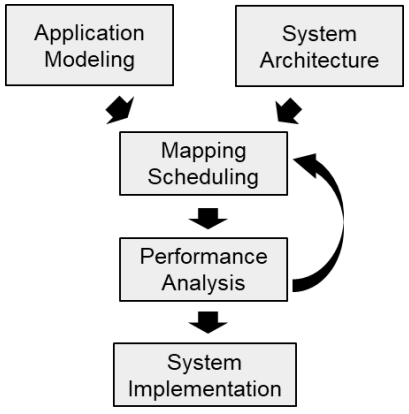


그림 3. 설계 공간 탐색 흐름도

임베디드 시스템의 설계 공간 탐색은 최적의 시스템 구성을 찾아가는 과정으로써, 시스템을 구성하는 하드웨어 요소 (System Architecture)와 응용의 모델링(Application Modeling)이 주어졌을 때 응용의 매핑 및 스케줄링을 결정한 후, 주어진 제약 조건을 만족하는지 성능을 분석(Performance Analysis)하고 제약 조건을 만족하는 경우에만 시스템을 구현한다. 이 때 제약 조건이 만족되지 않은 경우 새로운 매핑 및 스케줄링을 선택하여 다시 성능을 평가하여, 만족되는 조건을 찾을 때까지 반복한다. 제약조건을 높게 설정하여 보다 효율적인 매핑 및 스케줄링을 탐색할 수도 있다. 설계 공간 탐색 과정에서 매핑과 성능 분석 과정은 반복적으로 발생하므로 분석 시간을 줄이는 것이 빠른 시스템 설계 측면에서 중요하다.

3. 관련 연구

SDF 모델을 HSDF로 변환하는 기법은 2.2절에서 설명한 바와 같다. 즉 한 주기 동안 각 노드가 수행되는 횟수만큼 노드를 복사하는 방법으로 데이터 샘플의 사이즈에 따라서 기하급수적으로 노드 개수를 증가시킬 수 있다. 이 문제를 해결하기 위해 SDF 그래프를 변환하는 여러가지 기법들이 제안되었다.

[7]에서는 SDF 그래프에 적용할 수 있는 네 가지의 변환 방법을 제시하였다. 첫 번째는 사용자에 의해서 임의로 인접한 태스크들을 클러스터링 하는 방법으로 간단하지만 데드락(Deadlock)을 초래할 수 있음에 유의하여야 한다. 두 번째는 특정 프로세서에 매핑된 노드들을 클러스터링하는 방법이다. 다만 이는 스케줄을 고려하지 않고 묶는 방법으로 본 논문에서 제시하는 방법과 차이가 있다. 세 번째는 내부 상태가 없는 선형의 HSDF 그래프는 하나의 노드로 변환하는 방법이며, 네 번째는 내부 상태가 있는 SDF 그래프의 경우에는 펼쳐서 변환하는 방법으로 이 경우에는 노드의 개수가 오히려 증가할 수 있다. 그러나 본 논문에서 제시하는 방법에서는 내부 상태가 있어도 노드 개수를 줄일 수 있다.

[5]는 주어진 SDF 그래프를 HSDF 그래프로 변환할 때, 원 SDF 그래프의 대역폭(Throughput)과 지연 시간(Latency)에 대해서 보수적인 값을 가지도록 하는 HSDF 그래프를 만드는 방법을 제시하였다. 해당 방법은 2단계로 구성되어 있다. 첫 번째 과정은 주어진 SDF 그래프를 보다 작은 SDF 그래프로 변환하는 과정으로, 동일한 데이터 샘플 생성 비율을 가지는 노드들을 의존성을 고려하여 합친다. 이는 보다 작은 크기를 가지는 SDF 그래프로 변환시키나 합치는 과정에서 대역폭과 지연 시간이 원 그래프보다 보수적이게 된다. 두 번째 과정은 우선 한 주기 내에서 데이터 샘플이 소모되는 시간을 표현한다. 그리고 SDF 그래프를 Max-Plus 행렬로 변경하는 알고리즘을

수행하는데 이 때 데이터 샘플이 소모되는 시간 순서대로 알고리즘을 진행하여 HSDF를 만든다. 해당 방법은 원 SDF 그래프보다 보수적인 대역폭 및 지연 시간을 가지는 그래프로 변환하고, 원 SDF 그래프와 다른 수행 형태를 보일 수 있다. 또한, 실험 결과 노드 개수가 증가하는 경우도 있었다.

[6]은 노드의 가능한 가장 늦은 종료 시간(Latest Finish Time)에서 가능한 가장 빠른 시작 시간(Earliest Start Time)과 수행 시간을 뺀 변수인 슬랙(Slack)을 사용하여 그래프의 노드 개수를 줄이는 방법을 제시하였다. 양수의 슬랙 값을 가지는 노드가 있는 경우 교착 상태가 발생하지 않도록 반복적으로 노드들을 합치는 방법을 제시하였다. 양수의 슬랙 값이 있다는 것은 임계 경로(Critical Path)의 실행 시간이 응용의 종단간(End-to-End) 지연 시간 제한 조건을 초과 할 수 없음을 의미한다. 즉, 주어진 지연 시간 조건을 충족시키는 범위 내에서 최대한 노드를 합치는 방법이다.

앞의 모든 방법은 매핑 및 스케줄링이 결정되기 전에 그래프의 노드 개수를 줄임으로써 설계 공간의 범위를 줄였으며 이는 설계 공간 탐색 시 비효율적인 결과를 초래할 수 있다. 특히 [5], [6]은 노드의 매핑을 고려하지 않고, 수행 시간도 일정함을 가정한다. 다시 말해서, 이종 멀티 프로세서 환경에서의 설계 공간 탐색에 적합하지 않다. 그러나 본 논문에서 제시하는 방법은 매핑 및 스케줄링이 결정된 후에 클러스터링을 하여 원래의 그래프와 같은 설계 공간을 가진다. 설계 공간 탐색 시에는 매핑 및 스케줄링을 변경하면서 그래프 감소 및 성능 분석을 하여 효율적인 매핑 및 스케줄링을 찾을 수 있어서 설계 공간 탐색 측면에서 유용하다.

4. 문제 정의 및 풀이

4.1 문제 정의

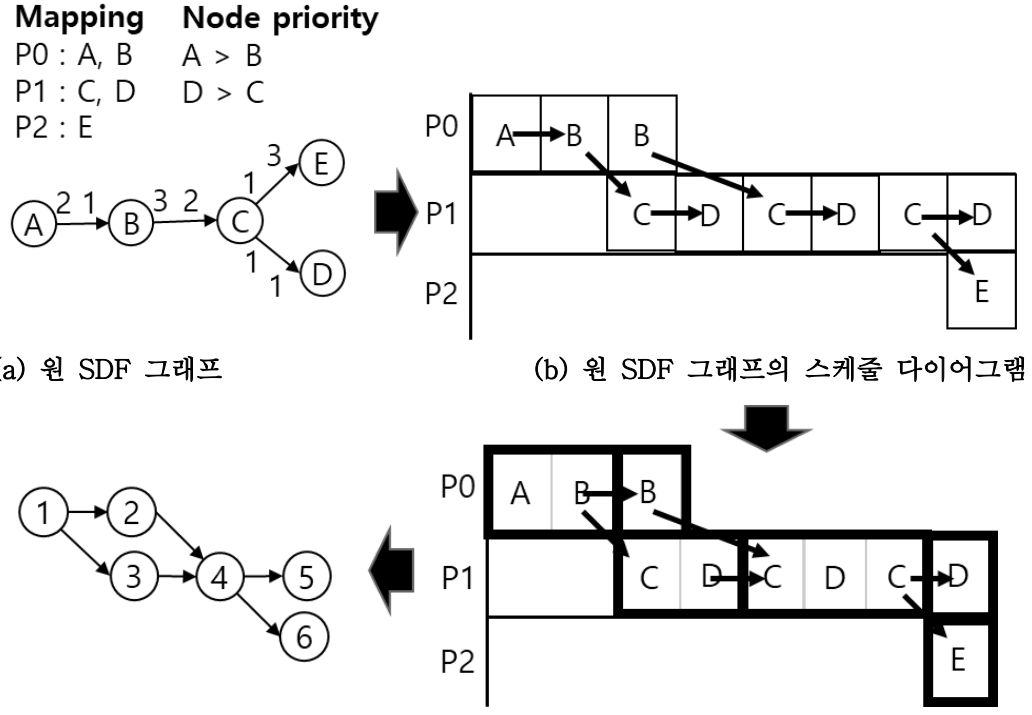
본 논문에서는 SDF 그래프로 표현된 응용과 이기종의 선점형 프로세서가 주어지고 노드의 매핑 및 우선순위가 정해졌을 때 그래프의 크기를 줄이는 문제를 풀이한다. 한 응용의 모든 노드는 다른 응용의 모든 노드보다 우선순위가 항상 높거나 낮게 주어지며, 모든 노드의 우선순위는 서로 다름을 가정하였다. 따라서 노드의 우선순위에 따라 응용의 우선순위를 매길 수 있다. SDF 그래프의 각 노드는 매핑 및 스케줄링의 단위로써 선점형 프로세서에 매핑될 수 있다. 노드의 매핑과 우선순위가 정해지면 하나의 스케줄이 정해지게 된다. 각 노드 별 각 프로세서에서의 수행 시간은 설계 공간 탐색 전에 주어졌다고 가정하였다. 또한, SDF 그래프는 일정한 주기를 가지고 주기적으로 실행되며 종료 시한이 응용의 주기를 넘지 않는 암묵적 종료 시한(Implicit Deadline) 조건을 가정하였다.

4.2 문제 풀이

주어진 그래프를 변환하는 과정은 다음과 같다. 먼저 주어진 SDF 그래프의 노드 매핑, 수행 시간, 우선순위를 기반으로 단일 응용의 스케줄을 생성한다. 이는 시간에 따른 노드의 실행을 시뮬레이션해서 얻을 수 있다. 클러스터링은 시뮬레이션을 통해 얻은 스케줄 다이어그램을 기반으로 수행된다.

클러스터링은 스케줄 순서에 영향을 주지 않으면서 여러 노드들을 하나의 큰 노드인 클러스터로 만드는 것을 의미한다. 클러스터링의 목표는 설계 공간 탐색 시 분석에 사용되는 노드 개수를 최대한 줄이기 위해, 가능한 많은 노드들을 합쳐 하나의 클러스터로 만드는 것이다. 클러스터링되어 합쳐지는 노드들은 같은 프로세서에 매핑되어 있어야 한다. 클러스터된 노드 또한 매핑 단위이기 때문에 하나의 클러스터가 여러 프로세서에 매핑된 노드들을 동시에 포함할 수 없기 때문이다. 또한 시작 노드와 끝 노드를 제외한 중간 노드는 다른 클러스터에 포함된 노드에 의존성을 갖지 않아야 한다. 이는 중간 노드가 다른 클러스터에 입력/출력 의존성을 갖는 경우, 다른 클러스터가 데이터를 생성한 직후에 클러스터 내 중간 노드가 수행되거나 중간 노드가 데이터를 생성한 직후에 다른 클러스터가 수행되는 불가능한 스케줄이 발생하기 때문이다. 따라서 클러스터를 생성하는 중요한 기준은 노드의 의존성 존재 여부이다. 클러스터는 시작 노드와 끝 노드만이 다른 클러스터에 의존성을 가질 수 있으며 클러스터의 의존성은 시작 노드의 입력 의존성과 끝 노드의 출력 의존성에 의해 정해진다. 각 클러스터의 수행 시간은 합쳐진 노드들의 수행 시간의 합으로 설정되고, 클러스터의 매핑은 합쳐진 노드들의 매핑과 동일하다.

마지막으로 클러스터링 된 스케줄 그래프로부터 새로운 DAG를 생성한다. 각 프로세서별로 하나의 클러스터를 하나의 노드로 대응시키고, 클러스터의 의존성을 노드의 간선으로 대응시킨다. 같은 프로세서에 매핑된 클러스터들 간에는 시간 순서에 따른 의존성을 추가함으로써 그래프의 실행 순서를 유지시킨다. 이를 통해 각 프로세서별로 모든 클러스터의 실행 순서가 정해져서 그래프의 실행 순서의 모호성이 없어진다. 새로 생성된 그래프는 원 그래프를 대신하여 성능 분석기의 입력으로 사용될 수 있다. 그림 4는 매핑과 노드 우선순위가 정해진 SDF 그래프의 스케줄 다이어그램을 기반으로 클러스터링을 수행하고, 클러스터링 된 스케줄 다이어그램으로부터 새로운 그래프를 생성하는 예제이다. 그림 4의 (d)의 노드 ①~⑥은 클러스터링 알고리즘을 통해 생성된 새로운 노드이다.



(a) 원 SDF 그래프 (b) 원 SDF 그래프의 스케줄 다이어그램

(d) 클러스터링된 그래프(DAG) (c) 클러스터링 적용 후 스케줄 다이어그램

그림 4. 그래프 변환 예시

클러스터링 알고리즘은 다음과 같다. 생성된 스케줄 다이어그램에서, 각 프로세서별로 연속되어 나타나는 노드들을 모두 묶어 하나의 클러스터로 만드는 초기 작업을 수행한다. 초기 클러스터링 작업에서는 노드의 의존성을 무시하며 단지 연속성만을 기준으로 클러스터링을 진행한다. 예시로 그림 4 (b)에서 초기 클러스터링을 수행하면 각 프로세서 별로 1개씩 총 3개의 초기 클러스터가 생성된다. 그 후 각 초기 클러스터에 대해 재귀적인 분할 작업을 수행한다. 알고리즘 1은 이러한 분할 작업을 수행하는 의사 코드이다.

알고리즘1: 클러스터 분할 알고리즘 의사 코드

```
1  atomize (C(n_1, n_2, ..., n_N))
2  |  for i = 1; i ≤ N; i = i + 1 do
3  |  |  if i ≠ 1 and hasDependency(n_k, n_i)
4  |  |  where n_k ∉ (n_1, n_2, ..., n_N) then
5  |  |  |  atomize(C(n_1, ..., n_{i-1}));
6  |  |  |  atomize(C(n_i, ..., n_N));
7  |  |  |  return;
8  |  |  else if i ≠ N and hasDependency(n_i, n_k)
9  |  |  where n_k ∉ (n_1, n_2, ..., n_N) then
10 |  |  |  atomize(C(n_1, ..., n_i));
11 |  |  |  atomize(C(n_{i+1}, ..., n_N));
12 |  |  |  return;
13 |  end
```

$C(n_1, n_2, \cdots, n_N)$ 은 n_1 부터 n_N 까지 N 개의 노드가 합쳐진 클러스터를 의미하고, $\text{hasDependency}(n_{k(i)}, n_{i(k)})$ $where$ $n_k \notin (n_1, n_2, \ldots, n_N)$ 함수는 노드 $n_{k(i)}$에서 노드 $n_{i(k)}$로 향하는 의존성 존재 여부를 반환하는 함수이다. 이 때 노드 n_k는 클러스터 $C(n_1, n_2, \cdots, n_N)$ 에 속하지 않은 노드이다. 즉 $\text{hasDependency}(n_{k(i)}, n_{i(k)})$ 함수는 검사하고자 하는 클러스터에 속한 노드 n_i로부터 클러스터 외부의 다른 노드에 의존성이 존재하는지 여부를 확인하는 함수이다. 주어진 알고리즘은 해당 클러스터에 속한 노드를 앞에서부터 순차적으로 검사하여 타겟 노드 n_i 가 클러스터 외부로 향하는/외부로부터 들어오는 의존성을 갖는지 확인한 후 해당 노드를 기준으로 클러스터를 분할한다. 만일 노드n_i가 외부로부터 들어오는 의존성을 갖는다면(라인 3-7), 기존 클러스터는 노드 n_{i-1}노드를 끝 노드로 가지며, 해당 노드 n_i는 새로운 클러스터의 시작 노드로 결정된다. 반대로 노드n_i가 외부로 향하는 의존성을 갖는다면(라인 8-12), 해당 노드는 기존 클러스터의 끝 노드로 결정되고, 다음 노드인 n_{i+1}노드는 새로운 클러스터의 시작 노드로 결정된다. 이를 통해 클러스터는 시작 노드와 끝 노드를 제외한 중간 노드가 클러스터 외부로의 의존성을 갖지 않게 된다. 분할되어 생성된 두 클러스터는 다시 알고리즘 1의 분할 과정을 거치며, 더 이상 분할되지 않는 클러스터는 최종 클러스터로 결정된다.

4.3 시간 복잡도

입력 그래프의 노드 수를 $\mathrm{N_g}$, 간선의 수를 E 라고 하자. 해당 클러스터링 알고리즘의 시간 복잡도는 다음 같이 계산할 수 있다. 라인 3, 8의 $\mathrm{hasDependency}(n_{k(i)}, n_{i(k)})\ where\ n_k \notin (\mathrm{n_1}, \mathrm{n_2}, \dots, \mathrm{n_N})$ 함수는 n_i 와 연결된 노드들이 다른 클러스터에 속해 있는지 여부를 검사함으로써 수행될 수 있는데 이 때의 검사 시간은 최대 그래프의 간선의 개수인 E이다. 또한 atomize 함수는 노드 개수가 N 개인 클러스터에 대해 노드의 입력/출력 의존성을 검사하며 분할 여부를 결정하므로 최대 N번의 검사를 수행한다. 따라서 크기 N의 클러스터를 검사하는 데에 소요되는 최악 수행 시간은 $O(N \cdot E)$이다.

하나의 클러스터는 크기 $N-l,\ l\ (1 \le l \le N-1)$ 인 두 클러스터로 분할될 수 있으며 따라서 atomize함수의 최악 수행 시간을 a_N 이라고 할 때 식 1의 관계식이 성립한다.

$$a_N = a_{N-l} + a_l + N \cdot E \quad (1)$$

크기 N 인 클러스터가 $\mathrm{n}\ (0 \le \mathrm{n} \le \mathrm{N}-1)$ 번의 분할 작업 후 $\mathrm{n}+1$개만큼의 더 이상 나눌 수 없는, 각각의 크기가 $l_1, l_2, \dots, l_{n+1}$ 클러스터로 분할된다고 가정하자. 앞의 점화식은 식 (2)로 계산된다.

$$a_N \le \sum_{k=1}^{n+1}(a_{l_k}) + n \cdot N \cdot E$$
$$< (N+1) \cdot \max(a_{l1}, a_{l_2}, \dots, a_{l_{n+1}}) + N^2 \cdot E \quad (2)$$

이 때 $\max(a_{l1}, a_{l_2}, \dots, a_{l_{n+1}})$ 은 상수값이므로, 크기 N 인 클러스터의 분할 시간 a_N은 $O(N^2 \cdot E)$의 시간 복잡도를 갖는다.

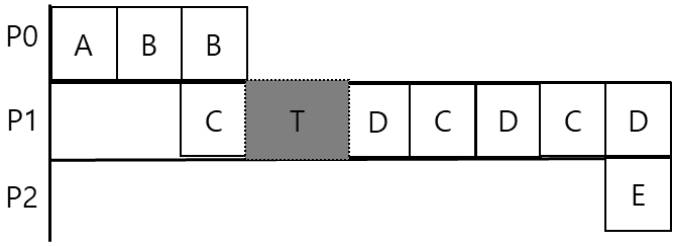
클러스터링은 프로세서별로 수행되므로 전체 수행 시간은 각각의 프로세서에서의 클러스터링 수행 시간의 합으로 계산된다. m 개의 프로세서에 $\mathrm{N_g}$ 개의 노드가 각각 $p_1, p_2, \dots, p_m$ 개씩 할당되었다고 하자. 각각의 프로세서에서 초기 클러스터링을 포함한 클러스터링 최악 수행 시간의 합은 식 (3)과 같다.

$$\sum_{k=1}^{m}({p_k}^2) \cdot E \le (\sum_{k=1}^{m} p_k)^2 \cdot E = {N_g}^2 \cdot E \quad (3)$$

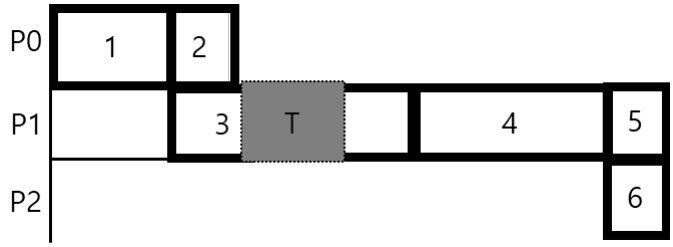
따라서 전체 그래프를 클러스터링하는 해당 알고리즘의 시간 복잡도는 $O({N_g}^2 \cdot E)$ 이다. 이를 통해 해당 알고리즘이 다항식 시간(Polynomial Time) 안에 수행이 완료됨을 알 수 있다.

4.4 보존성(Conservativity)

선점형 프로세서에서 원 그래프와 클러스터링 된 그래프는 성능 분석 관점에서 동일하다. 선점형 프로세서에서는 성능 분석 시 타겟 응용보다 우선순위가 낮은 응용에 의해서는 간섭이 발생하지 않는다. 또한 타겟 응용보다 우선순위가 높은 응용에 의한 간섭이 발생하는 경우 원 그래프에서의 간섭은 항상 새로운 그래프에서도 발생한다. 예로, 그림 5는 그림 4 예시에서 우선순위가 더 높은 노드 Ⓣ가 노드 Ⓒ와 Ⓓ 사이를 간섭하는 경우를 나타낸다. 원 그래프에서 노드 Ⓒ와 Ⓓ 사이에 발생하는 간섭은 새로운 그래프에서 노드 ③의 실행 도중 노드 Ⓣ가 해당 프로세서를 선점하여 발생하는 간섭과 동일하다. 원 그래프에서 발생하는 모든 간섭이 클러스터 된 그래프에서도 모두 발생하므로 성능 분석 단계에서 원 그래프 대신 새로운 그래프의 차이가 없어, 원 그래프 대신 새로운 그래프를 성능 분석에 사용할 수 있다.



(a) 원 그래프에서의 간섭 예시



(b) 새로운 그래프에서의 동일한 간섭

그림 5. 우선순위가 높은 노드에 의한 간섭

5. 실험

임의의 SDF 그래프를 생성하여 HSDF 그래프로 변환할 때 클러스터링을 적용하기 위해 SDF3 [9] 툴을 사용하여 랜덤 SDF 그래프를 생성하였다. SDF3는 그래프의 노드 개수와 각 노드의 반복 횟수의 합을 설정하여 교착 상태, 버퍼 오버플로우가 발생하지 않도록 보장된 임의의 그래프를 생성하는 기능을 제공한다. 그림 6은 노드의 개수를 5개, 반복 횟수의 합을 30으로 설정하였을 때 생성된 SDF 그래프의 예시이다. 간선의 이름 옆에 표시된 괄호 안의 숫자는 채널에 할당된 초기 데이터 샘플의 수를 의미한다. 이를 동치인 HSDF 그래프로 단순 변환하면 노드 반복 횟수의 합만큼인 30개의 노드가 생성된다. 데이터 생성/소모 비율은 주어진 노드 개수와 반복 횟수의 합을 기반으로 적당한 값이 할당된다. 모든 실험은 옥타코어(Intel i9-9900KF 5.0 GHz), 64GB RAM의 우분투 18.04 운영체제 머신에서 수행하였다.

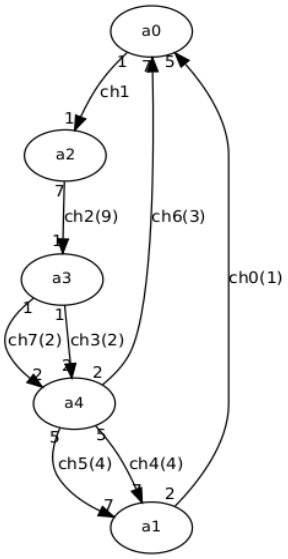


그림 6. 랜덤 생성된 SDF 그래프 예시

5.1. 실험1: 제시하는 알고리즘에 의한 노드 감소율

해당 실험에서는 임의의 SDF 그래프에 대해 클러스터링을 적용했을 때 노드 개수 감소율을 측정하였다. SDF 그래프에서 노드 개수를 5, 10, 50, 100개로 변화시키고, 노드 반복 횟수의 합을 노드 개수의 1, 2, 5, 10, 30, 50배로 설정하여 실험하였다. 각 노드의 실행 시간은 1~10 사이의 임의의 정수로 설정하였고, 이종의 프로세서 개수는 2개에서 4개까지 변화시키며 측정하였다. 각 노드는 주어진 프로세서 내에서 랜덤 매핑하였다. 그래프 형태, 노드의 매핑 등에 따라 클러스터링의 효과가 달라지므로 100회 반복 수행하여 그 평균값을 구하였다. 그 결과는 표 1과 같다.

표 1. 제시하는 알고리즘에 의한 노드 개수 감소율

SDF 노드 수	프로세서 개수	Rep값에 따른 노드 개수 감소율					
		1	2	5	10	30	50
5	2	19%	40%	48%	49%	65%	61%
	3	8%	35%	47%	49%	64%	61%
	4	4%	33%	46%	49%	64%	61%
10	2	12%	40%	55%	59%	60%	57%
	3	8%	38%	54%	58%	59%	57%
	4	5%	37%	53%	58%	60%	57%
50	2	7%	45%	68%	79%	80%	84%
	3	5%	44%	67%	79%	80%	84%
	4	4%	44%	68%	79%	80%	84%
100	2	6%	46%	73%	78%	88%	91%
	3	4%	44%	73%	79%	88%	91%
	4	3%	44%	72%	78%	88%	91%

표 1에서 Rep 값은 노드 반복 횟수의 합을 노드 개수의 Rep배 만큼의 값으로 설정하였음을 의미한다. 다시 말해서 해당 설정에서 노드 반복 횟수의 합은 노드 수와 Rep값이 곱으로 계산할 수 있다. 2.2절에서 단순 확장된 HSDF그래프에서는 노드 개수가 SDF 그래프의 노드 반복 횟수의 합만큼 생성됨을 이야기하였다. 따라서 단순 확장된 HSDF 그래프 대비 클러스터링 후 생성된 그래프의 노드 개수가 얼마나 감소하였는지를 통해 클러스터링의 효과를 계산할 수 있다.

제시하는 클러스터링 알고리즘은 SDF 그래프에서 Rep값이 1인 경우, 즉 그래프 자체가 HSDF 그래프인 경우에도 일부 클러스터링을 수행하였다. 또한, 대체로 동일한 SDF 그래프 노드 개수에 대해서 노드 반복 횟수의 합이 클수록 클러스터링이 더 효과적으로 수행되었다. 이는 노드 반복 횟수가 클수록 스케줄 상에서 하나의 노드가 연속적으로 여러 번 나타날 가능성이 높기 때문이다.

또한, HSDF가 아닌 SDF 그래프에서 동일한 평균 반복 횟수의 합에 대해 SDF 그래프 노드 개수가 많을수록 클러스터링이 효과적이었는데, 이는 노드 개수가 많을수록 클러스터링 될 수 있는 노드들이 연속적으로 나타나는 경우가 많아지기 때문이다. 반면 프로세서 개수가 많을수록 클러스터링의 효과가 조금 감소하였다. 이는 같은 수의 노드가 여러 프로세서에 매핑되는 경우 클러스터링이 될 수 있는 가능성이 적어지기 때문이다. 그러나 그 영향이 크지 않으며, 더욱이 노드 반복 횟수의 합이 커짐에 따라 프로세서 개수에 의한 영향이 거의 나타나지 않는다. 클러스터링에 의한 노드 개수 감소율은 그래프 크기가 가장 큰 경우인 노드 수 100개, 노드 반복 횟수의 합이 5000(Rep=50)일 때 최대치로써 노드 개수가 약 91%까지 감소하였다.

5.2. 실험2: 입력 그래프의 크기 및 개수에 따른 최악 성능 분석기 수행 시간

해당 실험에서는 가장 빠른 수행 시간과 정확한 최악 응답 시간 분석을 보이며 오픈 소스로 공개된 HPA [2]를 사용하여 성능 분석 시 소요되는 시간 감소를 확인하였다. 노드 클러스터링 전과 후의 그래프를 HPA의 입력으로 사용하였을 때 소요되는 시간을 측정 및 비교하였다. HPA는 DAG로 나타내어진 여러 응용을 입력으로 받아 그래프 내외의 간섭을 고려하여 각 그래프 별 최악 응답 시간을 출력하는 분석기이다. HPA의 입력으로 노드 개수가 10개, 50개이고 노드 반복 횟수의 합이 노드 개수의 5배인 각각의 그래프를 2개, 5개, 10개씩 사용하여 클러스터링을 적용하기 전과 후의 수행 시간을 비교하였다. 클러스터링 하지 않은 경우는 단순 변환한 DAG 그래프를 사용하였다. 클러스터링을 적용한 경우의 성능 분석 시간은 HPA 수행 시간과 클러스터링 수행 시간의 합으로

계산하였다. 프로세서의 개수, 노드 별 수행 시간, 매핑 방식은 실험 1과 동일하며 60회 반복 실험하였다. 실험 결과는 표 2와 같다.

표 2. 입력 그래프의 크기 및 개수에 따른 HPA 수행 시간

SDF 노드 수	그래프 개수	프로세서 개수	HPA 수행 시간 (클러스터링X) (단위: msec)	HPA 수행 시간 + 클러스터링 시간 (단위: msec)
10	2	2	21	20(28/72%)
		3	21	21(16/84%)
		4	41	15(19/81%)
	5	2	243	76(11/89%)
		3	251	70(12/88%)
		4	255	74(12/88%)
	10	2	1,973	490(10/90%)
		3	1,910	510(10/90%)
		4	1,959	526(11/89%)
50	2	2	925	105(58/42%)
		3	1,023	114(59/41%)
		4	1,191	117(60/40%)
	5	2	23,228	1,319(79/21%)
		3	32,284	1,495(81/19%)
		4	26,555	1,494(79/21%)
	10	2	255,090	14,861(89/11%)
		3	320,310	18,498(90/10%)
		4	335,431	22,466(92/8%)

HPA 수행 시간 + 클러스터링 시간 항목에서 괄호 안의 퍼센트(%)는 각각 HPA 수행 시간과 클러스터링에 소요된 시간의 비율을 의미한다. 노드 개수가 적을 때에는 클러스터링 작업을 수행하는 시간 비율이 HPA 수행 시간보다 높지만, 노드 감소에 의한 HPA 수행 시간 감소 효과가 더 크므로 클러스터링하지 않은 그래프를 입력으로 사용했을 때보다 전체 성능 분석 시간은 더 짧아졌다. 또한 노드 개수가 증가할수록 이러한 클러스터링 오버헤드가 감소하였다.

SDF 그래프 노드 개수가 증가하거나 입력 그래프의 수가 증가함에 따라 HPA의 수행 시간이 큰 폭으로 증가하였고, 일반적으로 경우 프로세서 개수가 증가함에 따라 HPA의 수행 시간이 증가하였다. 이에 따라 클러스터링에 의해 노드 개수의 감소가 전체 성능 분석 시간을 효과적으로 감소시키며, 노드 개수가 많아질수록, 입력 그래프의 수가 증가할수록 더 큰 폭으로 감소하였다.

6. 결론 및 향후 연구

본 연구에서는 SDF 그래프를 DAG로 변환할 때 SDF 그래프의 매핑 및 스케줄링 정보를 기반으로 노드 개수를 효과적으로 줄이는 클러스터링 알고리즘을 제안하였다. 제안하는 방법은 기존 연구들과 다르게 노드 개수를 줄이는 것뿐만 아니라 SDF 그래프의 매핑 및 스케줄링 문제의 설계 공간 범위를 해치지 않으면서 성능 분석 시간을 줄임으로써 설계 공간 탐색 측면에서 유리하다. 실험을 통해 클러스터링 알고리즘이 노드의 개수를 효율적으로 줄였으며, 타겟 성능 분석기인 HPA에서의 수행 시간이 크게 줄어들었음을 확인하였다. 특히 SDF 그래프의 크기가 클수록 노드 개수를 더욱 효과적으로 줄임을 확인하였다.

향후 연구로 제안한 클러스터링 알고리즘을 기반으로 이종 멀티프로세서 시스템에서 SDF 그래프로 명세된 여러 응용에 대해서 제약조건을 고려한 최적의 매핑 및 스케줄 탐색을 수행할 예정이다. 본 논문에서 제시한 방법으로 매핑과 우선순위가 정해지면 하나의 스케줄이 나오고 빠른 성능 분석이 가능하므로 이를 기반으로 매핑과 우선순위를 변경하면서 유전 알고리즘과 같은 메타 휴리스틱 알고리즘으로 효율적인 설계 공간 탐색을 진행할 예정이다.

7. 참고문헌

[1] E. A. Lee et al. “Static scheduling of synchronous data flow programs for digital signal processing.” IEEE Trans. Comput. 36, 24-35, 1, 1987. DOI:10.1109/TC.1987.5009446

[2] Choi, J., Oh, H. & Ha, S. “A hybrid performance analysis technique for distributed real-time embedded systems.” Real-Time Syst 54, 562-604 (2018) doi:10.1007/s11241-018-9307-x

[3] Jinwoo Kim, Hyunok Oh, Junchul Choi, Hyojin Ha and Soonhoi Ha, "A novel analytical method for worst case response time estimation of distributed embedded systems," 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, pp. 1-10, 2013.

[4] Rafik Henia et al. “System level performance analysis-the SymTA/S approach.” In IEE Proceedings-Computers and Digital Techniques 152.2, pp. 148-166, 2005.

[5] M. Geilen, "Reduction techniques for Synchronous Dataflow graphs," 2009 46th ACM/IEEE Design Automation Conference, San Francisco, CA, pp. 911-916. 2009, doi: 10.1145/1629911.1630146

[6] Hazem Ismail Ali et al. “Reducing the Complexity of Dataflow Graphs Using Slack-Based Merging.” In:TO-DAES22.2 pp.24. 2017. doi: 10.1145/2956232
[7] J. L. Pino and E. A. Lee, "Hierarchical static scheduling of dataflow graphs onto multiple processors," 1995 International Conference on Acoustics, Speech, and Signal Processing, Detroit, MI, USA, vol.4, pp. 2643-2646, 1995, doi: 10.1109/ICASSP.1995.480104
[8] Michael R. Garey et al. “Computers and Intractability; A Guide to the Theory of NP-Completeness.” New York, NY, USA: W. H. Freeman & Co., 1990.ISBN: 0716710455
[9] Stuijk, Sander & Geilen, M. & Basten, T. (2006). “SDF3: SDF for free.” Proceedings - International Conference on Application of Concurrency to System Design, ACSD. 23. 276 - 278. 2006. 10.1109/ACSD.

데이터 플로우 모델 기반 임베디드 소프트웨어 플랫폼에서의 공유 정보 관리 기술

강우석O, 홍혜선, 정은진, 마리리스 올드자, 하순회*
서울대학교 컴퓨터 공학부
wecracy@snu.ac.kr, hshong@snu.ac.kr, chjej202@snu.ac.kr, mari_liis@snu.ac.kr, sha@snu.ac.kr

Shared information management technique in an embedded software platform based on the dataflow model

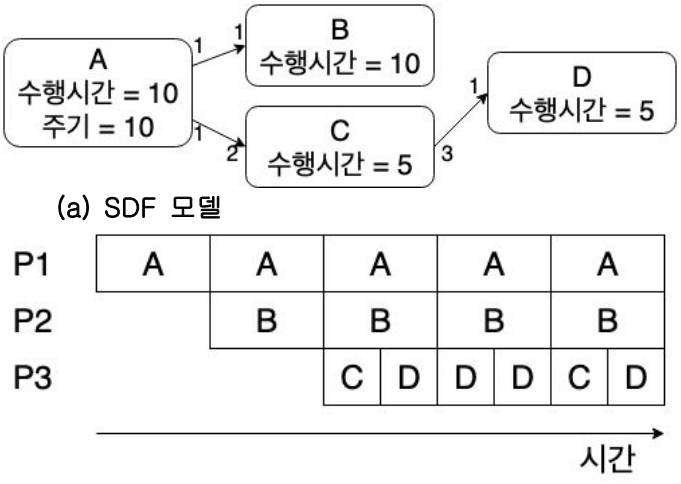
Kang WoosukO, Hong Hyesun, Jeong Eunjin, Mari-liis oldja, Ha Soonhoi*
Department of Computer Science and Engineering, Seoul National University

요 약

임베디드 소프트웨어의 개발을 위한 HOPES 프레임워크는 정형적인 데이터플로우 모델 명세로부터 정적 분석 및 코드 생성을 지원한다. 기존의 데이터플로우 모델은 연산 단위인 태스크 간의 통신을 채널로만 표현하고, 태스크가 컴파일 타임에 디바이스에 매핑되어, 구동 중에 공유 정보를 관리하는 데 제약이 있다. 따라서 본 연구에서는 멀티캐스트를 이용하여 태스크 간 통신하는 방법을 데이터플로우 모델에 추가하고, 구동 중에 동적으로 디바이스가 추가/삭제되더라도 전체 시스템에 영향을 주지 않도록 지원한다. 또한 공유 정보를 사용할 때 중앙에서 관리하는 것 외에도 분산적으로 관리할 수 있도록 한다. 본 연구에서는 스트리밍 서비스와 로봇들이 협업하는 시나리오를 사례로 보여, 제안하는 연구의 타당성을 검증하였다.

1. 서 론

임베디드 소프트웨어 개발은 일반 소프트웨어를 개발할 때보다 제한된 하드웨어 자원과 실시간성 보장 등 추가적인 요구조건으로 인해 높은 전문성을 필요로 한다. 따라서 임베디드 소프트웨어를 보다 쉽게 개발하기 위한 다양한 설계 방법론이 연구되었다. 그중 오픈소스로 된 HOPES 방법론[1]은 정형적인 모델 명세를 통해 정적 분석 및 성능 예측이 가능하며, 모델을 하드웨어에 매핑함으로써 실제 코드까지 자동 생성할 수 있다.



(a) SDF 모델

(b) (a)에 대한 매핑/스케줄링

그림 1. SDF 모델 및 매핑/스케줄링 예시

HOPES 방법론은 정형적인 데이터플로우 모델 중의 하나인 SDF (Synchronous Data Flow)[2] 모델로 응용을 명세하는 것을 기반으로 하고 있다. SDF 모델은 연산 단위인 태스크와, 태스크 사이의 데이터 전송을 나타내는 FIFO (First-In First-Out) 채널로 이루어져 있다. 채널에는 태스크가 수행되기 위해 필요한 데이터 수와 태스크가 수행된 후 생성되는 데이터 수를 [그림 1 (a)]에서처럼 표시하는데 이를 데이터 샘플 비(sample rate)라고 한다. 표시된 데이터 샘플 비는 고정되어, 태스크들의 수행 횟수 비율을 예측하고, 태스크를 프로세서에 정적으로 매핑하고 어떻게 스케줄링할 지를 결정할 수 있다. [그림 1 (a)]에서는 "A" 태스크는 주기적으로 수행되는 태스크이고, 나머지 태스크는 앞서 수행된 태스크의 결과를 입력으로 받아 수행되는 태스크이다. 여기서 “B” 태스크는 데이터 샘플 비에 따라 “A” 태스크가 한 번 수행된 후 한 차례 수행 가능하다. 반면, "C" 태스크는 “A” 태스크가 두 차례 수행이 되어야 한 번 수행 가능하다. “D” 태스크는 “C” 태스크가 수행된 이후 세 번 수행할 수 있다. 3개의 프로세서로 구성된 아키텍처에서 [그림 1 (a)]에 표현된 모델을 프로세서에 매핑 시 [그림 1 (b)]처럼 스케줄링이 가능하다. 이처럼 작성된 정형적인 태스크 그래프 모델은 분산 병렬 응용 프로그램을

명시적으로 명세할 수 있다. 또한, 개발자가 설계할 때 데이터 샘플 비를 통해 태스크 간 교착상태(deadlock)를 확인하고 정적으로 결정된 매핑과 스케줄링을 통해 성능과 자원 요구량을 예측할 수 있다. 또한, 병렬성을 잘 드러내는 모델로부터 실제 코드를 자동 생성할 수 있어서 하드웨어가 바뀌더라도 매번 새롭게 코드를 구현하지 않고 매핑을 바꾸어 코드를 재생성함으로써, 코드 개발의 효율성을 높일 수 있다. 그러나 SDF 모델은 제한적인 의미(semantics)로 인해 다양한 임베디드 소프트웨어 응용을 표현하는 데 한계가 있다. 이를 극복하기 위해 SDF 모델을 확장한 CSDF (Cyclo-Static DataFlow)[3], SADF (Scenario-Aware DataFlow)[4] 모델 등 여러 가지의 모델들이 제안되었다.

여러 확장된 모델 중 하나로 서버-클라이언트 모델 형태로 공유 자원을 표현하기 위해 라이브러리 태스크[5]가 제안되었다. 이 모델에서는 라이브러리 태스크 내에 함수를 정의하고, 각 태스크에서 함수 호출로 라이브러리 태스크에 접근할 수 있도록 하여 공유 자원을 관리할 수 있도록 하였다. 이때 공유 정보를 관리하는 라이브러리 태스크는 컴파일 이전 이미 특정 디바이스에 매핑되어 있으므로, 만약 라이브러리 태스크를 매핑한 디바이스에 문제가 생겨 공유 자원에 접근하지 못할 경우 전체 시스템에 영향을 미칠 수 있다. 예를 들어 여러 가지 센서와 액츄에이터로 구성된 로봇의 행위와 공유 정보를 [그림 2]처럼 표현할 수 있다[6]. 이때 공유 정보를 '로봇 1'에 매핑하였는데, '로봇 1'이 배터리가 떨어져서 동작할 수 없게 되면, '로봇 1'과 함께 정보를 공유하는 '로봇 n' 역시 공유 정보에 접근할 수 없게 된다. 또한, 기존의 모델은 채널을 통해서만 태스크 간에 정보를 주고받을 수 있으므로, 만약 여러 태스크에 동일한 정보를 전달하려고 할 때 여러 개의 채널을 이용하여 전달(broadcast)해야 한다. 채널을 이용할 경우 태스크의 개수와 태스크 간 연결 정보가 고정된다. 따라서 이미 응용이 구동 중일 때, 새로운 태스크가 추가될 수 없어, 로봇이 수행되는 도중 새로운 로봇을 투입하여 기존의 로봇들과 함께 임무를 수행할 수 없다. 이런 특성은 기존 모델들의 확장성을 떨어뜨리는 요인으로 작용한다.

본 연구에서는 SDF 모델의 정형성을 깨뜨리지 않는 범위에서 멀티캐스트를 이용하여 태스크 간 통신하는 방법을 추가하여 SDF 모델을 확장하는 기법을 제안한다. 멀티캐스트는 특정 집단에 속한 모든 태스크에 동일한 정보를 전달(broadcast)하는 것으로 채널을 이용하여 태스크 간 연결을 하지 않아도 된다는 장점이 있다. 또한, 멀티캐스트의 특성 덕분에 구동 중에 동적으로 디바이스가 추가되거나 삭제되더라도 전체 시스템에 영향을 주지 않게 된다. 그래서 기존의 라이브러리 태스크를 이용하여 중앙에서 공유 정보를

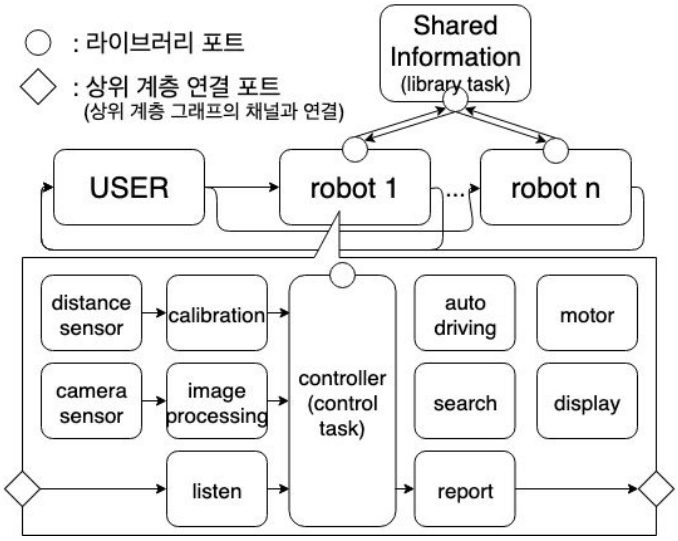


그림 2 여러 로봇의 움직임을 표현하는 태스크 그래프 예시

관리하는 방법 외에도 멀티캐스트 통신을 이용하여 공유 정보를 분산적으로 관리할 수도 있도록 지원한다. 그뿐만 아니라, 본 연구에서는 확장된 모델로부터 코드를 자동 생성하여 기존 방법론의 장점을 취하였다.

본 논문은 다음과 같이 구성되어 있다. 다음 장에서는 데이터플로우 모델 기반 임베디드 소프트웨어 개발 도구들과 함께 본 논문의 연구 배경인 HOPES 프레임워크를 소개한다. 3장에서는 기존의 공유 자원을 명세하는 라이브러리 태스크의 단점을 극복하고자, 멀티캐스트를 이용하여 확장한 태스크 간 통신 방법을 소개하고 확장된 모델에 따른 자동 코드 생성에 관해 설명한다. 4장에서는 라이브러리 태스크 모델과 본 논문에서 확장한 모델로 공유 정보를 관리하는 기법에 대해 설명한다. 제안하는 기법을 검증하기 위해, 5장에서는 스트리밍 서비스를 추가로 구독하는 예제와 두 대 이상의 군집 로봇들이 서로 협업을 하는 예제 시나리오에 대해 실험을 진행하고 6장에 결론을 맺는다.

2. 연구 배경 및 관련 연구

HOPES 프레임워크는 [그림 3]처럼 정형적인 데이터플로우 모델로부터 실행 가능한 코드를 자동 생성해주는 프레임워크이다. 본 프레임워크는 소프트웨어의 재사용성을 높이기 위해 알고리즘 정보(확장된 데이터플로우 모델)와 아키텍처 정보를 구분하여 명세한다. 태스크 그래프 모델은 태스크 그래프와 태스크 코드로 이루어져 있으며, 아키텍처 정보는 디바이스의 연산 요소 (processing element), 네트워크와 같은 하드웨어 정보와 운영체제 정보가 포함된다.

HOPES 프레임워크는 정형적인 데이터플로우 모델 중의 하나인 SDF 모델과 SDF 모델의 단점을 극복하기 위해 개발된 확장된 모델 및 크기 제한이 없는 FIFO

채널을 통해 정보 전달을 명세하는 KPN (Kahn Process Network) 모델을 이용하여 응용을 명세한다. 또한, 태스크 그래프를 계층적으로 표현할 수 있다. KPN 모델로 구성된 태스크 그래프의 경우, 내부에 유한상태기로 명세한 제어 태스크[7]나 SDF 혹은 KPN 모델을 가질 수 있다. 제어 태스크는 하나의 태스크나 하위 태스크 그래프를 가진 태스크를 제어하여 실행이 동적으로 바뀔 수 있다. 지원하는 확장된 SDF 모델의 경우, 응용 내 반복성을 명시적으로 드러낼 수 있는 루프 태스크[8]와 시스템 수준이 아닌, 응용 수준에서 동적 행태를 지원하는 MTM 태스크[7]를 지원한다. 그래서 이러한 확장된 모델들을 이용하여 신호 처리와 같은 순차적인 연산과 여러 로봇의 협업 등도 표현할 수 있다. 또한, 이 모델은 태스크 수준/시간적(temporal) 병렬성을 명시적으로 드러낼 수 있고, 병렬 프로그래밍을 작성할 때 발생할 수 있는 오버플로우나 교착 상태도 탐지할 수 있다.

사용자가 명세하는 태스크 코드는 'TASK_INIT', 'TASK_GO', 'TASK_WRAPUP' 세 부분으로 나누어지는데 먼저 'TASK_INIT' 부분은 처음 한 번 실행되는 코드로 각종 초기화 작업을 이 부분에 명세한다. 'TASK_GO' 부분은 실질적인 태스크 연산이 명시된 부분이며 매 구동마다 수행되는 부분이다. 이

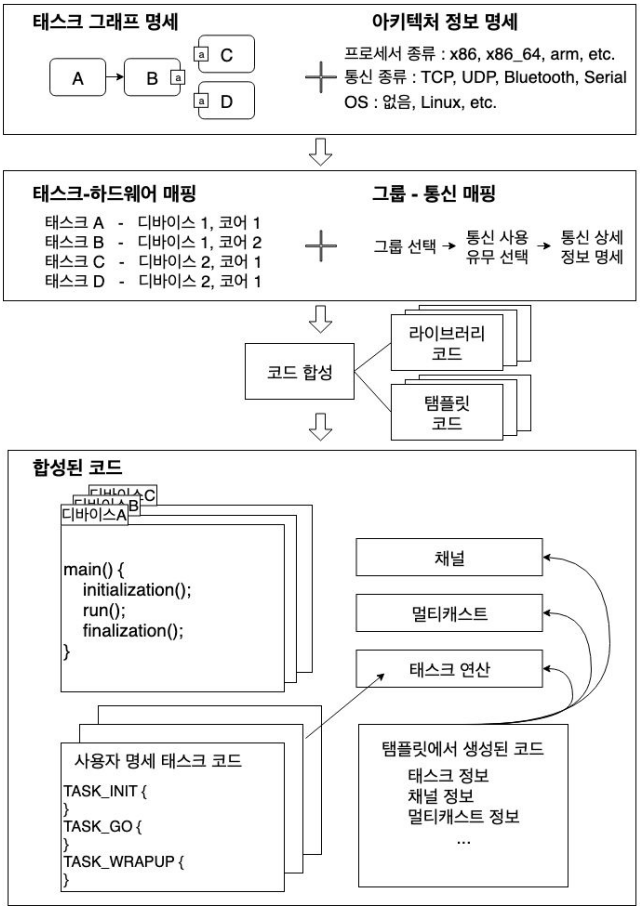


그림 3. HOPES 방법론의 소프트웨어 개발 흐름도

부분의 연산 시에 채널의 데이터를 읽어오고, 연산이 끝날 때 연결된 채널에 데이터를 보낸다. 각 태스크는 주기, 데이터, 또는 컨트롤 신호로 구동되는데 앞서 말한 바와 같이 [그림 1]에서 태스크 A는 일정 주기마다 수행되는 태스크이고 나머지 태스크는 데이터가 준비되었을 때 구동되는 태스크이다. 반면 [그림 2]에서 "auto driving", "search", "motor", "display" 태스크의 경우 제어 태스크인 "controller"에서 오는 제어 신호로 구동되는 태스크이다. 'TASK_WRAPUP'은 프로그램 종료 시 한 번 수행되는 코드로 랩업(wrap-up) 코드가 명세 된다.

태스크 모델과 아키텍처 정보를 이용하여 각 태스크를 수동 혹은 자동으로 아키텍처에 매핑한다. 이를 바탕으로 HOPES 프레임워크에서는 실제 코드를 원하는 디바이스에서 구동하기 전에 응용의 성능이나 자원 요구량을 분석할 수 있어 사전에 전체 시스템을 미리 검증할 수 있다. 또한, HOPES 프레임워크에서는 최악 성능 분석이나 처리량(throughput) 분석 등과 같은 정적 분석 도구를 제공하는데, 이를 통해 사용자는 디바이스에 최적화된 매핑 및 스케줄링을 확인할 수 있다.

HOPES 프레임워크에서는 태스크 정보, 아키텍처 정보, 매핑 정보를 종합하여 코드를 자동 생성할 수 있다. 코드 생성기는 사용자 명세에 독립적인 코드를 생성하는 것과 종속적인 코드를 생성하는 두 단계를 거친다. 우선 태스크와 채널에 대한 자료구조를 템플릿에 맞춰 자동으로 합성하고, 필요한 라이브러리만을 이용할 수 있도록 빌드 관련 설정 파일을 생성한다. 라이브러리의 형태로 관리되는 독립적인 코드는 아키텍처 정보를 이용하여 디바이스 종속적인 코드로 재정의된다. 라이브러리 코드는 비교적 안정적인 코드를 제공할 수 있기에 실제 구현 과정에서는 사용자 명세 부분만을 디버깅할 수 있도록 하여 임베디드 소프트웨어 개발이 쉬워지도록 만든다.

본 논문에서 사용하는 프레임워크 외에 다른 모델 기반 설계 프레임워크들을 소개하면 다음과 같다.

KPN 모델 기반으로 응용을 명세하는 도구로 Daedalus [9]와 DAL [10]이 있다. DAL의 경우, KPN 모델과 FSM(Finite-State Machine)을 사용하여 시스템 수준의 동적 행태를 명세할 수 있으며, 여러 플랫폼에서의 코드 생성을 지원한다. 그리고 Daedalus의 경우, 멀티미디어 응용에 대해 개발하는 데에 활용된다. 하지만 KPN 모델을 사용하는 한계상, 표현에도 한계가 있고, 성능 예측이 어렵다는 단점이 있다.

SDF 모델을 기반으로 둔 도구들로 StreamIt [11], Daedalus RT [12], 그리고 PREESM [13]이 있으며, 각각 개별적으로 확장된 SDF를 사용하여, 응용을 표현하고 있다. 이러한 도구들은 스트림 처리 응용이나 신호 처리 응용을 개발하는 데에 활용된다. 이러한

도구들은 SDF 모델의 특징상 표현할 수 있는 응용에 한계가 있다.

모델 기반으로 응용을 명세할 수 있는 상용 프레임워크로는 Mathworks 사의 Simulink 모델을 이용한 Simulink와 dSPACE 사의 TargetLink가 있다. 해당 모델을 이용하여 응용을 명세하고, 시뮬레이션도 함께 수행할 수 있다. 또 다른 상용 도구로 LabVIEW가 있으며, 자체적인 블록 다이어그램으로 모델로 응용을 명세한다. 이러한 도구에서는 정형적 모델로 분석이나 검증하기보다 시뮬레이션에 활용하며, 분산된 기기가 존재하는 환경을 대상으로 하기보다 단일 시스템 명세에 적합하다.

본 논문에서 활용하는 모델 기반 소프트웨어 설계 프레임워크와 같이 복합 모델을 사용하는 연구로 Ptolemy II 그룹에서 진행하였다 [14, 15, 16]. 해당 도구는 다양한 종류의 정형적 계산 모델을 활용하여 응용을 명세하도록 하는 것과 시뮬레이션을 지원한다. 하지만 코드를 생성하는 부분에서는 SDF 위주로 제한적이며, 분산된 기기를 대상으로 하는 코드는 생성하지 않는다.

다른 모델 기반 설계 프레임워크의 경우, 응용을 표현하거나 코드를 생성하는 대상 시스템 등에 한계가 있지만, 본 논문의 기법을 적용한 소프트웨어 개발 프레임워크에는 응용의 다양한 행태를 명세하고 코드를 생성할 수 있으며, 분산 시스템 환경도 고려하였기에 제안하는 기법을 해당 프레임워크에 적용하여 확장하는 연구를 진행하였다.

3. 제안하는 확장된 모델

기존의 태스크 그래프 모델에서는 태스크 간에 채널을 이용하여 일대일 통신만을 지원하였으나, 제안하는 모델은 멀티캐스트(multicast) 개념을 사용하여 여러 개의 태스크가 여러 개의 태스크에 정보를 한 번에 전달할 수 있다. 특히 멀티캐스트는 특정 그룹의 태스크끼리만 한 번에 동일한 정보를 주고받을 수 있다는 점에서 브로드캐스트(broadcast)와 다르다. 본 논문에서는 멀티캐스트를 이용하여 동적인 환경에서 분산정보를 표현할 수 있도록 기존의 SDF 모델을 확장하고, 태스크가 한 개 이상의 디바이스에 매핑이 가능하여 응용이 구동하고 있는 중에 디바이스 추가가 가능하도록 제안한다.

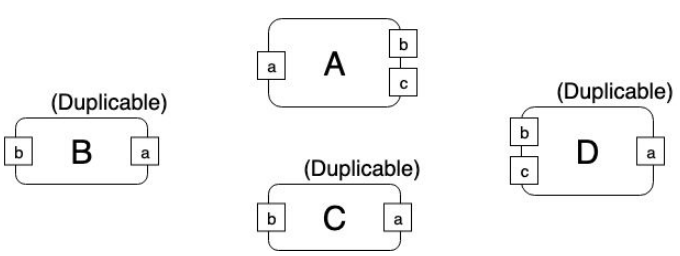


그림 5 멀티캐스트 예시

3.1. 확장된 모델의 구조, 동작 및 특징

기존의 태스크 그래프 모델은 앞서 설명하였듯이 독립적으로 실행 가능한 태스크와 태스크 사이의 데이터 전송을 나타내는 채널로 구성된다. 본 연구에서는 확장된 기능을 표현하기 위해 태스크에 추가적인 속성을 부여하고 멀티캐스트 포트와 멀티캐스트 그룹 개념을 새로 제안하였다. 먼저 태스크에 부여된 새로운 속성은 식 (1)과 같다.

태스크 = {태스크 이름, 채널 포트 목록,
멀티캐스트 포트 목록, 복사 가능 여부} (1)

멀티캐스트 포트 목록은 각 태스크가 지닌 멀티캐스트 포트의 집합이고 복사 가능 여부는 해당 태스크가 복사되어 한 개 이상의 디바이스에 매핑이 가능한지에 대한 속성이다. [그림 5]에서 보면 "B", "C", "D" 태스크는 태스크 상단에 표기한 것처럼 복사가 가능하여 여러 디바이스에 매핑이 가능함을 개념적으로 표현한다. 만약 기존의 모델을 이용할 경우 [그림 4]와 같이 직접 태스크를 복사되기 원하는 개수만큼 표현해야 한다. [그림 4]에 비해 [그림 5]가 '복사 가능 여부' 항목을 통해 중복적으로 모델을 설계하지 않아도 되어 빠르게 설계할 수 있음을 알 수 있다. 이때 태스크가 복사 가능하다면, 그 태스크는 반드시 해당 계층의 다른 태스크와는 별도의 디바이스에 매핑되어야 한다.

멀티캐스트 포트 = {이름, 방향, (2)
멀티캐스트 그룹 이름}

멀티캐스트 그룹 = {이름, 최대 버퍼 크기} (3)

멀티캐스트 포트는 식 (2)에서 정의한 바와 같은데, "이름"은 멀티캐스트 포트의 이름을 뜻하며, "방향"은 해당 멀티캐스트 포트가 데이터를 내보내는 곳인지, 아니면 데이터를 받아들이는 쪽인지 구분하는 것으로 {입력, 출력} 중 하나의 값을 취하게 된다. [그림 5]에서는 입력 포트는 태스크의 왼쪽에, 출력 포트는 태스크의 오른쪽에 표기하여 구분한다. "멀티캐스트

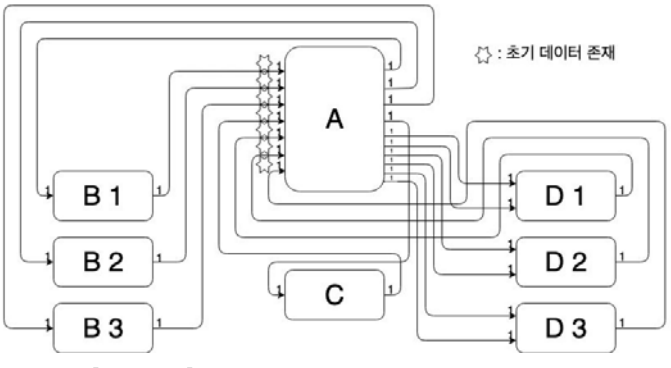


그림 4 [그림 5]의 채널을 이용한 태스크 그래프 명세

그룹 이름"은 멀티캐스트 포트가 속한 그룹을 표현하는 것이다.

여기서 그룹은 동일한 데이터를 함께 공유하는 태스크들의 집합으로 식 (3)처럼 정의된다. "이름"은 그룹을 구별하는 구별자로 앞서 언급한 멀티캐스트 포트 속성 중 “멀티캐스트 그룹 이름”의 요소가 된다. "최대 버퍼 크기"는 사전에 그 그룹에서 생산, 소비될 데이터의 최대 크기를 명시하도록 하여 최대 크기보다 큰 데이터를 보낼 수 없도록 한다. 임베디드 시스템은 제한된 자원 때문에 안정적인 운용을 위해서 메모리 동적 할당을 지양해야 하므로 멀티캐스트에 사용되는 데이터의 최대 크기도 명시되어야 한다.

구체적인 동작 예시를 들면, [그림 5]에서 "A", “D” 태스크는 3개의 멀티캐스트 포트를 가지고 있고, “B”, “C” 태스크는 각 2개의 멀티캐스트 포트를 가지고 있다. 본 예제에서는 멀티캐스트 포트의 이름과 멀티캐스트 그룹 이름을 동일하다고 가정하였다. "A" 태스크는 멀티캐스트 포트 b를 통해 그룹 b에 가입되어 있는 "B", "C", "D" 태스크에 정보를 전달하고 멀티캐스트 포트 c를 통해 그룹 c에 가입되어 있는 "D" 태스크에 정보를 전달한다. 다른 태스크의 경우에도 각 태스크가 가지고 있는 출력 포트를 통해 각 그룹에 해당하는 모든 입력 포트에 데이터를 전달할 수 있다. 만약 기존의 모델로 같은 응용을 표현하려고 하였다면, [그림 4]처럼 채널로 직접 연결해야만 한다. 이때 태스크 간 데이터 교류가 정적으로 고정되어, 확장성에 제한이 생긴다. 즉, [그림 5]와 같이 명세한 경우 "B", "C", "D" 태스크가 매핑된 각각의 디바이스를 임의로 늘리거나 줄일 수 있지만 [그림 4]와 같이 명세된 경우 디바이스 수가 고정된다. 게다가 [그림 5]에서는 완전히 새로운 태스크를 가진 디바이스도 그룹에만 속한다면 정보를 주고받을 수 있지만 [그림 4]에서 추가된 디바이스는 다른 디바이스로부터 정보를 주고받을 수 없다.

한편 일대일 통신을 사용하는 채널은 FIFO 큐로 이루어져 있어서, 데이터 샘플 비를 이용하여 사전에

```
TASK_INIT {
   ...
   UFMulticastPort_Initialize(...);
   UFMulticastPort_GetGroupSize(...);
   ...
}
TASK_GO {
   ....
   UFMulticastPort_ReadFromBuffer(...);
   ....
   UFMulticastPort_WriteToBuffer(...);
   ....
}
TASK_WRAPUP {
   ...
   UFMulticastPort_Finalize(...);
   ...
}
```

그림 6 멀티캐스트 API 예시

성능 예측 및 분석을 할 수 있다. 그러나 다대다 통신을 사용하는 멀티캐스트의 경우 누가 언제 데이터를 보냈는지 특정할 수 없기 때문에 버퍼를 사용한다. 이는 태스크 그래프 내에 순환 고리가 있으면 주고받는 데이터의 양이 무한히 증가하는 것을 방지한다. FIFO 큐로 만들어진 채널은 데이터가 반드시 태스크에 전달되는 것을 뜻하지만, 멀티캐스트는 버퍼를 사용하기 때문에 데이터 전송 신뢰성을 보장할 수 없다. 또한, 기존의 채널을 사용할 때에는 데이터 샘플의 크기, 데이터 샘플 비, 채널의 크기를 명세하여 채널의 크기가 태스크의 수행에 미치게 된다. 예를 들어 채널이 충분하게 크지 않아 태스크가 채널에 추가로 데이터 샘플을 전달하지 못할 때 태스크 수행이 멈추고 채널로 전달이 가능할 때까지 기다린다. 이에 반해, 멀티캐스트는 별도의 데이터 샘플 비를 명세하지 않고 버퍼를 사용하기 때문에 멀티캐스트를 사용함이 태스크를 수행하는 데 영향을 미치지 않는다.

3.2. 코드 생성

태스크 그래프 모델로부터 실제 코드를 자동 생성하기 위해, 태스크 그래프, 태스크 코드, 아키텍처 정보, 태스크의 매핑 정보뿐만 아니라, 멀티캐스트 그룹별 통신 정보를 입력으로 받는다.

먼저 태스크 그래프 모델의 경우, 앞서 설명한 멀티캐스트 그룹을 먼저 정의하고, 태스크에 멀티캐스트 포트 등록 시 그룹을 지정한다. 태스크 코드의 경우, [그림 6]과 같이 별도의 어플리케이션 프로그래밍 인터페이스(Application Programming Interface, API)를 이용하여 'TASK_INIT'과 TASK_WRAPUP'에서 각각 멀티캐스트 포트에 대한 초기화와 종료화를, 'TASK_GO'에서 실제 멀티캐스트 포트를 이용하여 데이터를 송, 수신하도록 명세 한다.

태스크 모델에 대한 명세가 끝난 이후에는 [그림 3]에서처럼 아키텍처 정보를 명세해야 한다. 아키텍처 정보는 프로세서의 종류, 공유메모리의 사이즈, 디바이스가 지원하는 통신 종류 및 각 디바이스의 통신이 서버, 클라이언트, 멀티캐스트 등 어떤 역할을 할 지 등으로 구성된다. 확장된 기능을 사용하기 위해서는 디바이스에 지원 가능한 통신 종류를 명세할 때 통신의 역할로 멀티캐스트를 지정해야한다. 아키텍처 정보와 태스크 모델을 토대로 매핑 단계에서 태스크를 각각의 연산 요소에 매핑하고, 그룹별 통신 방법을 함께 매핑해준다. 현재 멀티캐스트 통신은 사용자 데이터그램 프로토콜(User Datagram Protocol, UDP)만 지원하고 있으며, 그룹별 통신을 매핑할 때 그룹 통신에 사용될 멀티캐스트 대역의 IP 정보와 네트워크 포트 정보 등을 함께 명세한다.

사용자 명세가 끝나면 기존의 코드 합성에 더하여 본 연구에서 확장한 모델을 위해 [그림 7]과 같이 멀티캐스트에 대한 자료구조를 템플릿에 맞춰 추가로

합성한다. 사용자가 명세한 태스크 내부 코드 중 멀티캐스트 관련 API를 명세한 부분에서 동작 시에 UDP를 이용하도록 코드가 자동 생성된다. 즉, UDP 관련 라이브러리 함수를 이용할 수 있도록 앞서 명세한 UDP의 IP, 포트 정보 관련 자료 구조를 생성하고 통신 방법 관련 자료구조에 UDP를 이용할 것으로 명시하여 코드가 자동 생성된다. 확장된 기능을 지원하기 위해 라이브러리 코드도 구성하였는데 라이브러리 코드에는 UDP 소켓을 관리하는 부분, 멀티캐스트를 종합 관리하는 부분, 통신 방법에 맞추어 멀티캐스트 정보를 보내거나 받는 부분 등이 포함된다.

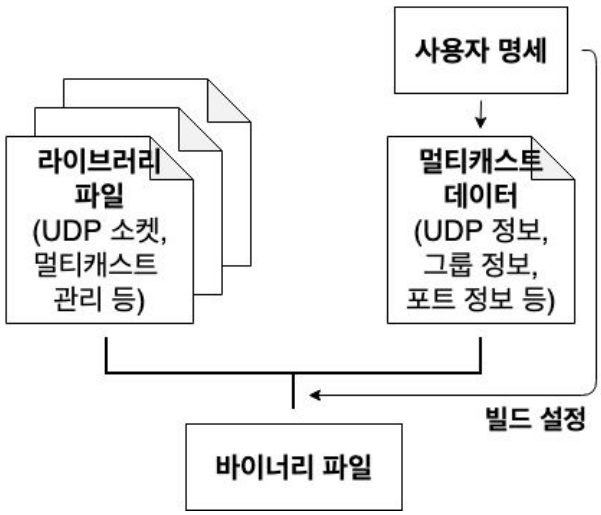


그림 7 멀티캐스트 코드 생성

4. 공유 정보 관리 기술

정보를 공유하는 방식으로는 크게 두 가지로 중앙 집중형 방식과 분산형 방식을 이용할 수 있다. 중앙 집중형 방식은 공유되는 정보를 특정 디바이스가 관리하고 있어 새로운 정보가 해당 디바이스에 들어올 때마다 정보를 갱신하고 요청이 발생할 때마다 정보를 보내주는 방식이다. 반면, 분산형 방식은 각자가 정보를 저장하고 새로운 정보를 갱신할 때마다 공유 정보를 사용하는 전체 디바이스에 보내주어서 각자의 정보를 갱신하도록 하는 방식이다. 기존의 SDF 모델은 라이브러리 태스크를 이용하여 공유 정보를 관리하도록 함으로써 중앙 집중형 공유 정보 관리 방식만을 지원하였다. 하지만 본 연구에서 확장한 멀티캐스트 기술을 이용하면 공유 정보를 특정 그룹의 태스크가 분산 관리하는 응용을 명세할 수 있다.

먼저 중앙 집중형 방식을 살펴보면 라이브러리 태스크는 다른 태스크처럼 매핑이 가능하므로, 라이브러리 태스크를 여러 디바이스 중 하나의 디바이스에 매핑하여 공유 정보를 중앙에서 쉽게 관리할 수 있도록 한다[5]. 이때 라이브러리 태스크가 매핑된 디바이스에서는 다른 디바이스로부터 오는 라이브러리 호출을 순차적으로 처리하여, 한 번에 하나의 요청만 공유 자원에 접근하도록 하므로 동기화 문제를 해결할 수 있다. 그러나 많은 태스크 혹은 디바이스에서 동시에 라이브러리 함수를 호출할 경우, 라이브러리가 매핑된 한 디바이스에서 이를 모두 처리해야하기에 병목이 발생할 수 있다. 또한, 라이브러리 태스크가 매핑된 디바이스에 문제가 생기면 라이브러리 태스크를 호출하는 모든 디바이스가 공유 정보에 접근할 수 없게 된다.

멀티캐스트를 통해 같은 그룹에 속한 디바이스가 데이터를 각자 저장하고 통신을 통해 동기화를 맞추는 분산 정보 관리 기술을 이용하게 되면, 중앙 집중형 관리에서의 단점을 보완하게 된다. 정보 처리가 하나의 디바이스에 의존하지 않고 그룹에 속한 모든 디바이스가 각자 정보를 처리하기 때문이다. 따라서 디바이스가 안정적이지 않으면 중앙 집중형 공유 정보 관리 기술 대신 분산 정보 관리 기술을 통해 데이터의 영구적 손실을 막을 수 있다. 분산 정보 관리 시에 동기화를 하는 방법으로는 정보에 시간을 함께 적어 데이터를 읽을 때 최신 정보인지 확인하는 방법을 사용할 수 있다[17].

멀티캐스트를 이용한 분산 정보 관리 기술에도 한계점이 있는데, 먼저 멀티캐스트는 아무런 제약 없이 사용한다면 기존 SDF 태스크 모델의 정적 분석을 해칠 수 있다. SDF 모델에서 각 디바이스에 매핑된 여러 태스크가 스케줄링 가능하여 성능을 분석할 수 있으면 정형성 조건을 만족한다. 만약 멀티캐스트가 디바이스 내의 여러 태스크 간 통신에 사용되면 누가, 언제 데이터를 보낼지가 스케줄링에 따라 달라져 정적 분석이 어렵다. 이는 라이브러리 태스크가 명시적으로 태스크와 채널을 표현하여 [18]에서 제시한 바와 같이 정적 분석이 가능한 것과는 대조된다. 다만 만약 확장한 분산 정보 관리 기술이 디바이스 간에만 사용되는 것으로 한정한다면 디바이스 내부의 스케줄링 가능성을 해치지 않아 정형성을 해친다고 볼 수 없다.

다음으로 멀티캐스트를 통한 태스크 간의 정보 전달은 데이터를 수신하는 모든 포트가 제대로 수신했는지 확인하지 않으므로 정보가 확실히 전달되었는지 보장하지 않는다. 게다가 정보를 수신하는 채널이 FIFO 큐가 아닌 버퍼이기 때문에 정보가 전달되었더라도 그 정보를 읽기 전 다른 태스크가 같은 그룹에 정보를 전달하게 되면 이전 정보를 잃게 된다. 따라서 데이터를 확실히 전달해야 하는 경우 멀티캐스트는 적합하지 않다.

두 가지 공유 정보 관리 기술은 각자 장단점을 가지고 있으므로 정보의 특성에 따라서 상황에 맞는 공유 방식을 이용해야 한다. 앞서 말한 바와 같이, 같은 디바이스 내에서 다른 태스크들이 정보를 공유하거나, 확실한 정보 전달이 필요한 경우에는 라이브러리

태스크를 이용한 중앙 집중형 방식을, 안정적인 디바이스를 확보할 수 없거나 디바이스의 추가/삭제가 필요한 경우 멀티캐스트를 이용한 분산형 공유 정보 관리 기술을 이용하는 것이 좋다.

5. 예제 검증

제안하는 기법의 타당성을 확인하기 위해 두 가지의 실제 예제 시나리오를 이용하여 검증하였다.

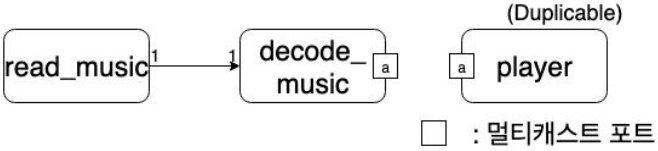


그림 8. 멀티캐스트를 이용한 스트리밍 서비스 표현

5.1. 스트리밍 서비스

스트리밍 서비스는 서버에서 제공하는 방송, 혹은 오디오 데이터를 클라이언트가 받아 정보를 처리하는 서비스로 클라이언트가 언제 생성되고 소멸하는지에 대한 정보가 사전에 정의되지 않는다. 따라서 단말의 추가/삭제가 동적으로 자유로워야 하는데 이런 상황은 기존의 데이터플로우 모델로는 명세하기 어려웠다. 따라서 본 연구에서는 새로 확장된 기술을 이용하여 MP3 스트리밍 서비스 실험을 하였다.

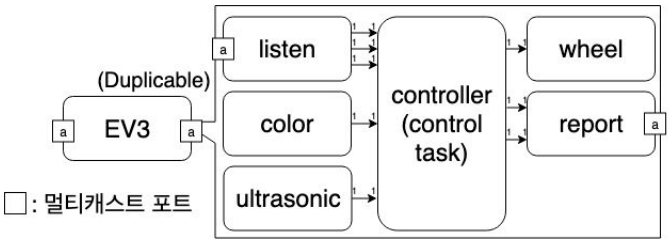
MP3 스트리밍 실험은 서비스를 제공하는 서버와 이를 재생하는 클라이언트를 3대의 라즈베리파이(Raspberry Pi 3 Model B+, 라즈베리파이용 OS “Raspbian” 탑재)로 구현하였다. [그림 8]과 같이 서버는 MP3 파일을 로드하고 재생할 수 있도록 디코딩하여 클라이언트인 "player" 태스크에 멀티캐스트 포트를 통해 전달하는데, 이때 "player" 태스크는 클라이언트의 추가/삭제를 위해 복사 가능한 태스크이다. 실험을 위해 하나의 클라이언트가 음악을 재생하고 있는 도중에 클라이언트가 추가/삭제되는 시나리오를 만들어 동작 여부를 확인하였다. 실험 결과 클라이언트가 추가된 시점부터 음원이 재생되는 것을 확인할 수 있었다. 태스크 간 데이터 전송을 기존의 채널에서 멀티캐스트 포트로 변경하고, 태스크 코드 내 API를 변경하는 것만으로 여러 개의 클라이언트를 생성할 수 있다는 점에서 효율적이다.

5.2. 로봇 서비스

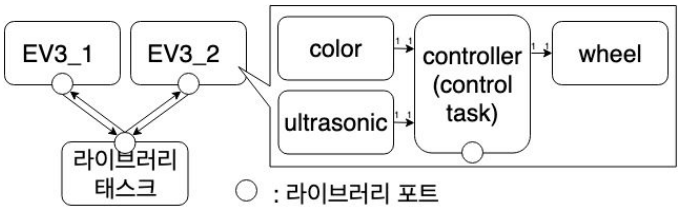
군집 로봇들이 하나의 공통된 미션을 수행하기 위해 임무 도중에 로봇이 추가/삭제될 수 있어야 안정적으로 미션을 수행할 수 있다. 또한, 군집 로봇의 경우 분산 정보를 관리함으로써 협업 시 안정적으로 미션을 수행하도록 할 수 있다. 이를 시험하기 위해 두 대의 EV3 로봇 (TI Sitara AM1808 - ARM926EJ-S core, 데비안 리눅스 기반의 "ev3dev" 탑재) 이 색깔 정보를 공유하면서 색종이 찾기를 수행하는 예제 시나리오를 가정하였다.

[그림 9 (a)]는 협업 예제 시나리오에 대해, 멀티캐스트를 이용하여 명세한 태스크 그래프이다. 로봇은 컬러 센서와 초음파 센서로부터 오는 정보를 병렬적으로 처리할 수 있도록 이들 구성 요소를 각각의 태스크로 표현하고, 외부로부터 받은 정보(센서, 통신)에 따라 동적으로 다르게 행동하기 위해 외부로부터 받은 정보는 "controller"라는 제어 태스크로 전달되어 처리된다. 제어 태스크 내에서는 유한 상태기(finite state machine, FSM)로 명세 되어, 상황에 맞게 “wheel” 태스크로 로봇의 움직임(액츄에이터)을 제어한다. 이때 다른 로봇에게 찾은 색종이 정보를 공유하기 위해 "listen"과 "report" 태스크는 멀티캐스트 포트를 이용하여 다른 로봇과 통신한다.

이에 반해 [그림 9 (b)]는 라이브러리 태스크를 이용하여 두 로봇 간의 색깔 정보를 공유하였다. 라이브러리 태스크는 채널로 연결된 태스크 간 통신만 가능하므로, 하나의 로봇이 먼저 수행하다가 추후 로봇이 추가될 수 없으며, 라이브러리 태스크가 매핑된 로봇이 멈추게 되면 로봇들이 더는 색종이 정보를 공유할 수 없게 된다. 하지만 멀티캐스트를 활용할 경우 로봇들이 정보를 분산 관리하기 때문에 이러한 단점이 사라지는 것을 확인할 수 있었다.



(a) 멀티캐스트 통신을 이용하여 명세한 로봇 태스크 그래프 (분산형 방식)



(b) 라이브러리를 이용하여 명세한 로봇 태스크 그래프 (중앙 집중형 방식)

그림 9. 로봇 협업 시나리오의 표현

5.3. 실험 요약

스트리밍 서비스는 데이터플로우 모델에서

디바이스의 동적 추가/삭제 가능성을 검증해주는 실험으로 IoT(Internet of Things)와 같이 디바이스 추가, 삭제가 수시로 일어날 수 있는 상황이나, 기존 응용들의 구동이 멈추지 않아야 하는 상황에서 새로 디바이스가 추가되어야 할 때 확장된 모델이 이용될 수 있음을 보여주었다. 로봇 예제 시나리오에서는 여러 디바이스가 협력하여 일을 처리할 수 있도록 하는 분산 정보 관리 기술을 검증하였다. 로봇을 포함한 임베디드 기기의 특성상 고장, 배터리 소진 등의 문제로 안정성을 보장할 수 없는데, 기존 데이터플로우 모델은 새로운 디바이스를 추가할 수 있는 모델을 명세할 수 없었으나 제안한 기술로의 확장을 통해 이러한 문제를 극복하여 비교적 안정적으로 서비스를 제공할 수 있음을 확인하였다.

표 1 자동생성된 코드 라인 수 비교

	총 라인 수	사용자 명세	템플릿 코드	라이브러리 코드
스트리밍 서버	29,454	15.3%	3.5%	81.2%
스트리밍 클라이언트	24,447	0.2%	1.9%	97.9%
로봇 (1대 기준)	26,614	3.3%	4.0%	92.6%

또한, 해당 실험은 HOPES 프레임워크를 이용하여 [표 1]에서 볼 수 있듯이 개발 효율성을 높였다. 스트리밍 서비스의 경우 실제 사용자가 작성한 코드는 스트리밍 서비스의 서버에서 4,519줄에 불과하지만 생성된 코드가 템플릿 코드에서 생성된 코드가 1,033줄, 라이브러리 코드에서 23,902줄이어서 전체의 15.3% 불과하다. 클라이언트 역시 사용자가 명세한 것은 52줄이지만 자동으로 생성된 코드는 24,395줄로 사용자가 명세한 부분이 0.2%로 매우 적음을 알 수 있다. 로봇 내 모듈 코드 역시 약 800여 라인의 태스크 코드로부터 태스크 간 병렬 프로그래밍과 로봇 간의 통신 코드가 모두 고려된 2만 라인이 넘는 복잡한 코드가 자동으로 생성되었다. 각각의 실험에서 자동으로 생성된 코드 대신 통신에 필요한 코드나 오류 처리를 위한 코드, 멀티쓰레딩을 위한 코드 등을 직접 개발했다면 상당히 많은 코드를 오랜 시간에 걸쳐 특정 응용을 생성하는데 개발해야 했기에, 해당 방법을 이용하여 개발의 효율성을 높였다고 볼 수 있다.

6. 결론 및 향후 연구

본 논문에서는 정형적인 SDF 모델에 멀티캐스트를 이용한 태스크 간 통신 방법을 확장하여 분산 정보 관리 기술을 제안하였다. 멀티캐스트 통신은 통신 도중 데이터 손실 가능성이 있기 때문에 디바이스 간에만 사용하는 것으로 제한함으로써, 디바이스 내의 정적 분석은 여전히 가능하다. 또한, 분산 정보 관리 기술을 통해 특정 디바이스의 의존성 없이 분산 시스템의 안정적인 서비스가 가능해지고 새로운 디바이스의 추가나 삭제가 쉬운 이점이 있다.

추후에는 HOPES 프레임워크 내에서 사용되고 있는 SDF 확장 모델들에도 제안하는 기술을 확장하여 다양한 예제에서 적용해볼 예정이다. 또한, 다양한 디바이스의 멀티캐스트 통신을 지원하기 위해 블루투스를 활용[19]하는 등 다양한 방법의 구현을 지원하고자 한다.

[사사(감사의 글)] 본 연구는 국방생체모방 자율로봇 특화연구센터를 통한 방위사업청과 국방과학연구소 연구비 지원으로 수행되었습니다 (UD190018ID).

참고문헌

[1] S. Ha and H. Jung, “Hopes: programming platform approach for embedded systems design,” Handbook of Hardware/Software Codesign, pp. 951-981, 2017.

[2] E. A. Lee and D. G. Messerschmitt, “Synchronous data flow,” Proceedings of the IEEE, vol. 75, no. 9, pp. 1235-1245, 1987.

[3]. P. Wauters, M. Engels, R. Lauwereins, and J. A. Peper-straete, “Cyclo-dynamic dataflow,” Proceedings of 4th Euromicro Workshop on Parallel and Distributed Processing. IEEE, pp. 319-326. 1996.

[4] S. Stuijk, M. Geilen, B. Theelen, and T. Basten, “Scenario-aware dataflow: Modeling, analysis and implementation of dynamic applications,” 2011 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation. (SAMOS) IEEE, pp.404-411, 2011.

[5] H.-w. Park, H. Jung, H. Oh, S. Ha, "Library support in an actor-based parallel programming platform," IEEE Transactions on Industrial Informatics, vol.7, no. 2, pp. 340-353, 2011.

[6] 홍혜선, 정한웅, 하순회, “로봇 협업 운용을 위한 프로그래밍 모델 및 코드 생성 기술,” 한국컴퓨터종합학술대회, pp. 558-560, 2015.

[7] H. Jung, C. Lee, S.-H. Kang, S. Kim, H. Oh, and S. Ha, “Dynamic behavior specification and dynamic mapping for real-time embedded systems: Hopes approach,” ACM Transactions on Embedded Computing Systems (TECS), vol. 13, no. 4s, p. 135, 2014.

[8] H. Hong, H. Oh, and S. Ha, “Hierarchical dataflow modeling of iterative applications,” Proceedings of the 54th Annual Design Automation Conference

(DAC) 2017. ACM, p. 39, 2017.
[9] H. Nikolov, M. Thompson, T. Stefanov, A. Pimentel, S. Polstra, R. Bose, C. Zissulescu, and E. Deprettere, “Daedalus: toward composable multimedia mp-soc design,” Proceedings of the 45th annual Design Automation Conference. (DAC) ACM, pp. 574–579, 2008.
[10] L. Schor, I. Bacivarov, D. Rai, H. Yang, S.-H. Kang, and L. Thiele, “Scenario-based design flow for mapping streaming applications onto on-chip many-core systems,” Proceedings of the 2012 international conference on Compilers, architectures and synthesis for embedded systems. (CASES) ACM, pp. 71–80, 2012.
[11] Thies, William, Michal Karczmarek, and Saman Amarasinghe. "StreamIt: A language for streaming applications." International Conference on Compiler Construction. (CC) Springer, Berlin, Heidelberg, pp. 179–196, 2002.
[12] M. A. Bamakhrama, J. T. Zhai, H. Nikolov, and T. Stefanov, “A methodology for automated design of hard-real-time embedded streaming systems,” Proceedings of the Conference on Design, Automation and Test (DATE) in Europe. EDA Consortium, pp. 941–946, 2012.
[13] M. Pelcat, J. Piat, M. Wipliez, S. Aridhi, and J.-F.Nezan, “An open framework for rapid prototyping of signal processing applications,” EURASIP journal on embedded systems, vol. 2009, no. 1, 2009.
[14] Bhattacharyya, Shuvra S., Praveen K. Murthy, and Edward A. Lee. Software synthesis from dataflow graphs. Vol. 360. Springer Science & Business Media, 2012.
[15] Tsay, Jeffrey J. A Code Generation Framework for Ptolemy II. Electronics Research Laboratory, College of Engineering, University of California, 2000.
[16] S. Tripakis, D. Bui, M. Geilen, B. Rodiers, and E. A. Lee, “Compositionality in synchronous data flow: Modular code generation from hierarchical sdf graphs,” ACM Transactions on Embedded Computing Systems (TECS), vol. 12, no. 3, p. 83, 2013.
[17] J. Kim, M. Kim, M.-O. Stehr, H. Oh, and S. Ha, “A parallel and distributed meta-heuristic framework based on partially ordered knowledge sharing,” Journal of Parallel and Distributed Computing, vol. 72, no. 4, pp. 564–578,2012.
[18] H. Jung, H. Oh, and S. Ha, “Multiprocessor scheduling of an sdf graph with library tasks considering the worst case contention delay,” Proceedings of the 14th ACM/IEEE Symposium on Embedded Systems for Real-Time Multimedia. (ESTIMedia) ACM, pp. 84–93, 2016.
[19] C. Cordeiro, S. Abhyankar, and D. P. Agrawal, “A dynamic slot assignment scheme for slave-to-slave and multicast-like communication in bluetooth personal area networks,” GLOBECOM’03. IEEE Global Telecommunications Conference (GLOBECOM), vol. 7., pp. 4127–4132, 2003.

커버러지 방법을 이용한 심층 신경망 구조 최적화

이민수 ○ 이찬근
중앙대학교 소프트웨어학부
als950901@naver.com, cglee@cau.ac.kr

Optimization of Deep Neural Network Structure Using Coverage Methods

Min-soo Lee○ Chan-gun Lee
School of Computer Science and Engineering, Chung-Ang University

요 약

최근 심층 신경망의 발전과 더불어 심층 신경망을 시험 검증하기 위한 많은 연구가 진행되고 있다. 이 중 가장 활발한 연구가 진행되고 있는 것이 커버리지 방법이다. 본 연구는 현재 연구 진행된 커버리지 방법들을 이용하여 학습한 데이터의 최적의 심층 신경망 구조를 찾는 것을 목표로 한다. 같은 학습 데이터를 학습한 다양한 구조의 서울시 관악구 기온 예측 시스템의 커버리지를 계산하고 그것을 비교 분석한다.

1. 서 론

최근 심층 신경망의 급속한 발전과 더불어 심층 신경망(deep neural network)에 대한 시험 검증의 중요성이 부각되고 있고, 심층 신경망을 검증하기 위한 많은 연구가 계속하여 진행되고 있다. 이러한 연구들은 대부분 일반적인 소프트웨어 시스템에 적용해오던 검증 방법을 심층 신경망에 맞게 변형하여 적용하는 것에서 시작하였으며, 이 중 가장 활발한 연구가 진행되고 있는 것이 커버리지(coverage) 방법[1][2][3]이다.

일반적인 소프트웨어 시스템에서의 테스트 커버리지 방법은 테스트 케이스(test case)를 이용해 활용되지 않는 코드의 일부분을 찾는 것이다. 이를 심층 신경망에 알맞게 변형한 방법은 뉴런 커버리지 (neuron coverage)이다. 심층 신경망은 일반적인 소프트웨어 시스템과는 다르게 코드가 아닌 층(layer)으로 구성되어 있으며, 각 층은 뉴런으로 구성되어 있다. 따라서 뉴런 커버리지는 활용되지 않는 뉴런을 찾는 것이다.

본 논문은 기존 정확도와 비용을 이용해 모델을 선택하는 방법과 달리 뉴런 커버리지를 이용하여 신경망을 구성하고 있는 각 층을 분석하여 신경망 구조를 최적화하는 방법을 제시한다.

본 논문의 구성은 다음과 같다. 2절에서는 본 논문에서 사용한 심층 신경망 기반 시스템, 뉴런 커버리지 방법과 측정 알고리즘을 소개한다. 3절에서는 신경망 구조 최적화에 대한 실험 방법과 분석 결과에 대해 소개한다. 마지막으로 4절에서 본 논문의 결론과 향후 연구 방향에 대해 소개하며 마무리한다.

2. 배경 지식

본 절은 본 논문에서 사용한 심층 신경망 시스템, 뉴런 커버리지 방법을 소개한다.

먼저 본 논문에서 사용하는 최적화 대상 심층 신경망 시스템은 기온 예측 시스템이다. 이 시스템은 서울특별시 관악구의 기온, 습도, 증기압, 이슬점온도, 현지기압, 해면기압, 지면온도, 5cm, 10cm, 20cm, 30cm 지층온도를 바탕으로 다음 날 같은 시간 대의 온도를 예측하는 시스템이다. 2017년 1월부터 2019년 12월까지 총 11090개의 데이터 중 6654개를 학습 데이터로 사용하고, 나머지를 테스트 데이터로 사용하였다. 평가 모델은 RMSE(Root Mean Squared Error)를 이용했다.

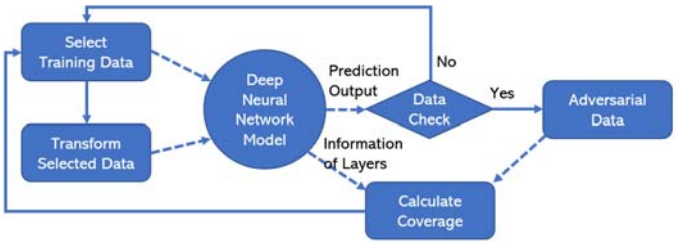


그림 1 커버리지 측정 알고리즘

커버리지를 측정하는 알고리즘은 그림 1과 같다. 알고리즘의 진행은 실선, 필요한 정보는 점선으로 표시했다. 그림 1과 같이 뉴런 커버리지 계산은 학습된 신경망에 입력을 가해 발생되는 각 층의 뉴런의 출력을 대상으로 뉴런의 활성화 여부를 판단하여 활성화된 뉴런을 기반으로 진행된다. 하지만 이미 학습된 데이터나 그와 비슷한 테스트 데이터는 신경망이 민감하게 반응하지 못한다. 따라서 우선 현재 보유 중인 데이터에 조금의 변형을 가한 후, 변형된 데이터를 입력으로 사용하여 뉴런 커버리지를 계산해야 한다.

변형된 데이터를 입력으로 가해 얻어지는 각 층의

표 1 장단기 메모리 층 커버리지와 활성화 횟수의 평균과 분산도

	CC	GC	SPC	SNC	평균(CC/GC)	분산도(CC/GC)
Lstm_1_dense_1	1	1	0.62	0.68	17 / 36.83	180.5 / 47.72
Lstm_2_dense_1	1	1	0.68	0.78	22.17 / 14.25	165.14 / 37.08
Lstm_3_dense_1	0.167	0.833	0.59	0.62	3.42 / 21.92	92.58 / 127.08

표 2 완전 연결 층 피쳐 별 커버리지

	2	4	6	8	10	12	14	16	18	20
TNC	1	1	1	1	1	1	1	1	1	1
KMNC	1	1	1	1	1	1	1	1	1	1
BNC	0.5	1	1	1	0.5	0.5	0.714	0.625	0.588	0.15

뉴런의 출력을 기반으로 해당 뉴런의 활성화 여부를 판단하고, 각 층의 전체 뉴런과 활성화된 뉴런의 비율로 커버리지를 측정한다. 이를 선별된 모든 데이터 셋을 대상으로 반복하여 측정한다.

하지만 이와 같은 커버리지 방법은 학습 알고리즘을 기준으로 다르게 측정되어야 한다. 따라서 본 논문에서 사용한 커버리지는 장단기 메모리 층을 대상으로 하는 Cell(CC), Gate(GC), Sequence Positive(SPC), Sequence Negative(SNC) 커버리지[3], 그리고 완전 연결 층을 대상으로 하는 Threshold (TNC)[1], K-Multisection(KMNC), Boundary(BNC)[2] 뉴런 커버리지이다.

3. 실험 방법 및 결과

우리는 실험을 위해 기온 예측 시스템의 신경망의 장단기 메모리 층을 1, 2, 3개로 다르게 하여 3개, 완전 연결 층의 피쳐를 2, 4, 6, 8, 10, 12, 14, 16, 18, 20으로 다르게 10개의 신경망을 학습시켰으며, 이 신경망들의 각 RMSE 수치는 0.119에서 0.129사이로 비슷하여 비교하기 힘들다. 따라서 우리는 심층 신경망 모델의 최적화를 위해 같은 데이터를 학습한 여러 개의 신경망 구조를 커버리지 수치를 이용하여 비교 분석하였다.

위 표 1은 실험에 대한 결과로 장단기 메모리 층의 커버리지와 각 피쳐의 CC, GC의 활성화 횟수에 대한 분산도와 평균을 나타낸다. 마찬가지로 표2는 완전 연결 층의 커버리지, 표3은 BNC에서 좋은 성능을 보이는 피쳐의 활성화 횟수에 대한 분산도와 평균을 나타낸다.

표 3 BNC의 뉴런 활성화 횟수에 대한 평균과 분산도

	4	6	8
평균	53.5	66.67	95.5
분산도	2511.25	1884.22	1652.73

4. 결론

먼저, 장단기 메모리 층을 1, 2, 3개를 계층적으로 쌓은 구조로 구성된 신경망을 비교하였다. 커버리지 결과와 활성화 횟수, 그리고 활성화 횟수의 평균과 분산도를 기반으로 분석하였다. 이에 대한 결과로 세 신경망 중 커버리지 성능이 높고, 활성화 횟수가 고루 분포한 2개를 계층적으로 쌓은 신경망 구조가 최적이라는 것을 알 수 있었다.

그리고 완전연결 층의 최적의 피쳐 수를 찾기 위해 다른 피쳐의 완전 연결 층을 가지는 10개의 신경망의 커버리지 결과와 활성화 횟수, 그리고 활성화 횟수의 평균과 분산도를 비교 분석하였다. 4, 6, 8 피쳐를 가진 신경망이 BNC의 성능이 좋았으며, 그 중 활성화 횟수의 평균과 분산의 성능이 좋은 8 피쳐 완전연결 층을 가진 신경망이 최적이라는 것을 알 수 있었다.

이를 바탕으로 현재 학습 데이터에 가장 최적의 신경망 구조는 2개의 계층 장단기 메모리 층과 8 피쳐 완전 연결 층이라는 것을 유추할 수 있다.

감 사 의 말

본 연구는 한국연구재단 기초연구사업 (과제번호: NRF-2017R1E1A1A01075803), 한국원자력연구원 원자력 ICT 안전성 검증체계 구축 및 운영과제(과제번호: 524320-18), 산업통상자원부 및 산업기술평가관리원(KEIT) 산업기술혁신사업 (과제번호:20003580)의 지원을 받았음.

참 고 문 헌

[1] K. Pei, Y. Cao, J. Yang, S. Jana, "DeepXplore: auto mated whitebox testing of deep learning systems,"in Proc. of SOSP, ACM, pp. 1-18, Oct, 2017.

[2] Lei Ma, Felix Juefei-Xu, Jiyuan Sun, Chunyang Chen, Ting Su, Fuyuan Zhang, Minhui Xue, Bo Li, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang, “Deepguage: Comprehensive and multi-granularity testing criteria for gauging the robustness of deep learning systems,” In Proc. of ASE, pp. 120-13, 2018.

[3] Wei Huang, Youcheng Sun, James Sharp, Xiaowei Huang, “Test Metrics for Recurrent Neural Networks,” CoRR, abs/1911.01952, 2019, URL: https://arxiv.org/abs/1911.01952

소프트웨어 결함 예측의 조선해양/해상운송 산업 적용 사례 연구

강종구[1] 류덕산[2] 백종문[3]
한국과학기술원[1,3] 전북대학교[2]
jjang9dr@kaist.ac.kr, duksan.ryu@jbnu.ac.kr, jbaik@kaist.ac.kr

요 약

소프트웨어 결함 예측(Software Defect Prediction)은 기계학습(Machine Learning) 기법을 적용하여 과거의 소프트웨어 결함 및 업데이트 정보를 학습한 모델을 기반으로 새로 개발된 소프트웨어 결함을 사전에 예측하는 연구이다. 이를 통해 실제 산업에서 소프트웨어 품질보증(QA) 자원을 효율적으로 운영/배치하기 위한 가이드로 활용할 수 있다. 최근 산업 적용 사례들이 일부 보고되고 있지만, 특성이 서로 다른 다양한 도메인 적용 및 이를 적용하면서 얻은 통찰을 실제에 반영하는 연구가 보다 활발하게 필요한 상황이다. 본 논문에서는 최근 고효율 친환경 선박, 커넥티드 선박, 스마트 선박, 무인 선박, 자율운항 선박 등 미래 운송수단으로의 변화에 직면해 있는 조선해양/해상운송 산업에 소프트웨어 결함 예측의 적용 가능성을 제시한다. 해당 도메인에서 수집된 실제 데이터를 활용하여 실험을 수행한 결과 0.91 Accuracy와 0.831 F-measure의 높은 결함 예측 성능을 보여 가능성을 확인하였고, 기존 사례가 없는 해당 산업으로의 적용 방안을 제시하여 QA 자원 배치를 효과적으로 지원하는 도구가 될 것으로 기대 된다.

1. 서 론

소프트웨어는 오늘날 많은 산업들에서 혁신을 위한 가장 중요한 추진 동인임과 동시에 지배적인 역할을 차지하고 있다. 육상/항공/해상 운송에 필요한 vehicle을 생산하는 기존의 산업들도 산업을 둘러싼 요구와 패러다임의 변화로 인해 고효율 친환경화, 무인 및 자율화, 초연결화와 같은 엔지니어링 도전들에 직면해 있으며, 소프트웨어는 변화를 실현하는 핵심적인 역할을 하고 있다. 최근 조선해양 및 해상운송 산업도 역시 유사한 엔지니어링 도전에 직면해 있다. 국제적인 환경 규제 요구를 충족하기 위하여 새로운 청정연료나 추진시스템을 탑재한 고효율 친환경 선박, 육상 서비스와 위성통신으로 연결되는 커넥티드 선박, 경제운항, 안전운항, 효율적인 운영을 제공하는 스마트 선박, 선박의 운항 및 운영이 자동화되어 인건비를 줄일 수 있는 무인 선박, 그리고 추가적인 인지시스템을 통해 운항 자체를 단계적으로 자율화 하는 자율운항선박 등의 실현이다.

이와 같은 소프트웨어 기반 혁신은 소프트웨어 품질이 가장 기본적으로 전제되어야 한다. 특히, vehicle의 안전 중요 (Safety-Critical) 특성으로 인하여 품질 문제의 발생은 곧 인명이나 재산 손실로 이어질 수 있어 그 중요성이 더욱 크다. 소프트웨어 결함 예측은 과거에 기록된 결함 및 업데이트 정보를 활용하여 새로 개발하는 소프트웨어의 결함을 사전에 예측함으로써, 결함 가능성이 높은 모듈에 보다 많은 품질보증 자원을 투입하도록 우선순위화 하는 효과적인 운영을 지원하는 기술 혹은 도구이다. 또한, 최근 소프트웨어공학 분야에서 활발하게 연구가 진행 중인 분야이다. 하지만, 소프트웨어공학 연구자들이 실제 산업 데이터를 접근하는데 제약이 많고 활용할 데이터셋도 제한되어 있어 실제 산업체에 적용을 목표로 하는 연구 사례가 많지 않은 상황이다. 따라서, 소프트웨어 결함 예측의 기법들을 적용하면서 얻은 통찰과 결과를 실제 산업 프랙티스에 반영할 수 있도록 분석과 가이드를 주는 연구가 보다 활발하게 이루어질 필요가 있다.

본 논문에서는 최근 소프트웨어 기술이 중심이 되는 미래 운송수단으로의 변화에 직면해 있는 조선해양 및 해상운송 산업의 소프트웨어 결함 예측의 적용 가능성을 실험을 통해 검증하고 기존 산업에의 적용 방안을 제시하고자 한다.

이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2017M3C4A7066212 & NRF-2019R1G1A1005047)

2. 조선해양 및 해상운송 산업의 소프트웨어 결함 예측 적용

그림 1은 조선해양 및 해상운송 산업에서의 프랙티스에 소프트웨어 결함 예측을 적용하는 방안의 예시를 나타낸다.

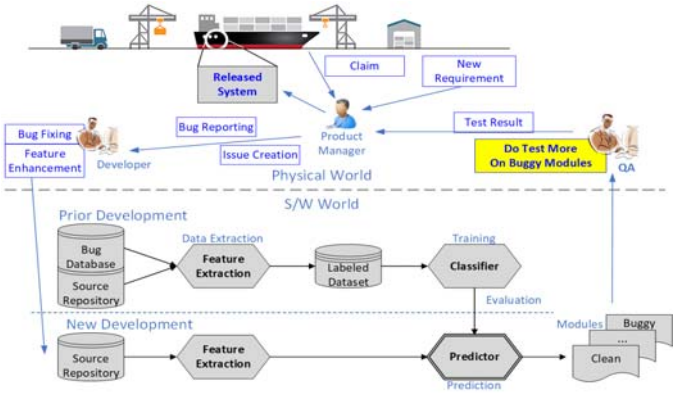


그림 1 소프트웨어 결함 예측 적용 예시

조선해양 및 해상운송 산업에서의 기존 소프트웨어 개발의 프랙티스를 아래와 같이 요약 정리하였다.

- 요구사항은 시스템과 S/W 관련 규정을 만족해야 하며, 공인된 기관으로부터 필수 규정에 대응하는 시험을 받고 인증을 받아야 함
- 설계는 다양한 선종 및 호선의 변경을 적시에 만족하기 위해서 commonality와 variability를 분리함. Variability는 모델주도 개발, IEC61131-3 규정을 만족하는 프로그래밍 언어, 혹은 이와 유사한 방식으로 구현함
- 시스템 통합, 품질보증 시 단위시험, Model In the Loop(MIL), S/W In the Loop(SIL), H/W In the Loop (HIL) 등 정교화된 시험 환경을 활용함
- 개발된 기능은 시스템통합 이후 요구했던 기능과 성능을 만족하는지 안벽시운전 및 해상시운전을 통해 검증하고 고객과 independent verifier에 의해 확인됨
- 배포 이후 발생된 결함에 대해서는 고객으로부터 claim report의 형태로 접수됨
- 결함은 수정, 보완, 시험 이후 원격 지원을 통해 패치, 패치된 H/W 모듈 교체, 서비스 엔지니어의 출장 및 방문 등의 적절한 방법을 통해 해결함

위와 같은 기존의 프랙티스에 추가적으로 소프트웨어 결함 예측 적용을 통해 feature enhancement나 bug fix 이슈 발생 시 과거 데이터로 학습하여 생성된 분류기로 수정된 모듈을 buggy나 clean 클래스로 결함 여부를 예측하도록 한다. Buggy로 예측된 모듈에 품질보증 자원을 보다 많이 할당하여 한정적인 자원의 전체적인 비용 효율을 높이는 것을 기대할 수 있다.

3. 실험 환경 구성 및 결과

앞 장에서 설명한 조선해양 및 해상운송 산업의 소프트웨어 결함 예측 적용 가능성을 실험하기 위해서 두 가지 연구질문을 수립하고, 대형선박에 적용되었던 S/W에서 축적된 결함과 업데이트 정보를 추출한 데이터셋을 가지고 실험을 구성하였다. 본 데이터는 원래 소프트웨어 결함 예측을 적용할 계획이 없이 과거부터 CVS 소스관리 툴에 기록되었다. CVS 상의 축적된 데이터로부터 총 14개의 변경 수준 척도를 추출하였다.[3] 데이터셋의 각 인스턴스는 14개의 척도와 하나의 결함 여부를 표기한 레이블을 포함하고 있다. 본 S/W로부터 학습할 데이터를 추출하기 위해 SZZ Unleashed 구현체를 일부 수정하여 사용하였다.[4] 학습 시에는 훈련 데이터와 검증 데이터 분리를 위해 10-fold cross validation을 적용하였다.

성능 평가를 위한 지표로 소프트웨어 결함 예측에 널리 사용되는 accuracy, precision, recall, f-measure의 4가지 지표를 사용하였다. 기계학습 알고리즘은 지도학습 중에서 Naive Bayes(NB), Logistic Regression (LR), Support Vector Machine(SVM), k-Nearest Neighbor(NN), Random Forest (RF), Multi-Layer Perceptron (MLP)을 선택하여 실험하였다. k-NN 알고리즘에서 k는 1,5,10을 사용하여 총 8가지 경우의 알고리즘에 대해서 실험을 수행하였다. 실험에는 모두 널리 알려진 기계학습 툴인 Weka를 사용하여 진행하였다. k-NN의 k를 제외한 모든 파라미터들은 Weka에서 제공하는 default 값을 사용하였다.

●RQ1: 제안하는 모델이 결함을 얼마나 잘 예측할 수 있는가? (How Well Can The Proposed Model Predict Defect-Inducing Changes?)

먼저 추출된 데이터셋을 4가지 방식으로 전처리한 후 실험을 진행하였다. N0는 원본 데이터셋 이고 N1, N2는 각각 min-max, z-score 정규화(normalization)로 전처리 하였다. N0에서는 MLP가 accuracy 0.889와 f-measure 0.652로 가장 좋은 성능을 보였다. N1에서는 SVM와 RF 성능 지표에 각각 소폭 감소, 증가가 있었으나 그 외 알고리즘에는 영향을 주지 않았다. N2에서는 RF 성능 지표가 소폭 상승하였고 다른 알고리즘은 영향을 받지 않았다. N3에서는 비결함 클래스가 결함 클래스 대비 다수를 차지하는 클래스 불균형을 일부 보완하기 위해서 SMOTE 기법을 적용한 oversampling 100%를 데이터셋에 적용하였다. 결과는 성능이 가장 좋은 RF의 경우, accuracy가 0.91, f-measure가 0.831이었다. Precision이 8.60, recall 또한 0.804이었다. 본 실험의 결과를 비교할 대상으로 accuracy, precision, recall이 모두 0.75 이상이면 성공적인 결함예측 모델이라는 검증 기준을 참고하였다.

본 실험 결과, 성능이 가장 좋은 N3의 RF 결과가 accuracy 0.91, f-measure 0.831, precision 8.60, recall 0.804로 모든 성능 지표에 대해 0.75 기준보다 크므로 "본 산업 데이터를 사용한 제안 모델의 결함 예측 성능이 0.75 기준보다 같거나 높다"는 결과를 도출하였다.

●RQ2: 본 산업에서 결함을 유입하는 변경들의 고유한 특성이 있는가? (Are There Unique Characteristics of Defect-Inducing Changes?)

추출된 14개의 척도에 대해서 t-test를 수행한 결과, buggy와 clean class 간 평균에 유의한 차이를 나타내며, 예측력에 대해 중요도가 높은 지표는 NF, Entropy, LT, NUC의 4가지 척도로 나타났다. Kamei et al. [3]은 상용 S/W 프로젝트에서 diffusion factor들이 예측력에 중요한 요소임을 발견하였다. 본 실험의 결과에서도 NF와 Entropy는 예측력에 중요한 요소임은 동일하였다. 하지만, 본 실험 결과는 본 산업의 프로젝트에서는 NF, Entropy, LT, NUC가 중요 척도로 나타났다. 따라서, "본 산업 프로젝트에 높은 결함 유입요소들은 다른 산업 프로젝트의 요소들과 동일하지 않다"는 결과를 얻을 수 있었다.

4. 관련 연구

Kamei et al.[3]는 Just-in-Time 결함 예측 기법을 제안하였다. 오픈소스와 상용 S/W를 포함하여 총 11개의 프로젝트에서 추출된 데이터를 기반으로 logistic regression 예측 모델과 14개의 변경 수준 척도를 활용하였다. 본 논문은 동일한 척도를 사용하였지만, 다양한 분류 모델로 실험하고 특정산업 프랙티스에서의 적용에 공헌하고자 했다.

Altinger et al.[1]는 자동차 도메인에서 모델주도 개발 방식으로 자동 생성된 C 코드로부터 추출된 데이터셋을 제시하고, 후속연구[2]에서 결함 예측 성능을 실험하였는데 precision과 f-measure가 각각 0.21와 0.34 이하로 낮은 성능 지표를 보여주었다. 본 논문은 다른 특성을 가진 조선해양/해상운송 산업의 도메인 데이터셋을 실험하였다.

5. 결 론

본 논문에서는 조선해양 및 해상운송 산업에서 적용 사례가 보고되지 않은 소프트웨어 결함 예측의 적용 가능성을 해당 도메인에서 수집된 데이터를 활용하여 실험한 결과 0.91 accuracy와 0.831 f-measure의 높은 예측 성능을 보여 가능성을 보여주었다. 개발 중인 혹은 배포된 S/W의 결함을 유입하는 수정 발생 시, 과거 수집된 데이터로 학습한 분류기로 수정 모듈에 대해 예측을 수행하여 결함 가능성이 높은 모듈에 품질보증 자원을 투입하는 소프트웨어 결함 예측은 집중개발과 품질보증이 필요한 현 프랙티스에 유망한 도구가 될 것으로 기대 된다. 산업의 프랙티스를 개선하고 비용효율을 높이기 위해 보다 많은 통찰과 가이드라인을 제시하는 것이 향후 과제이다.

참고 문헌

[1] H. Altinger, et al.,"A novel industry grade dataset for fault prediction based on model-driven developed automotive embedded software," In *Proceedings of the 12th Working Conference on Mining Software Repositories*, pp. 494-497, 2015

[2] H. Altinger, et al.,"Novel insights on cross project fault prediction applied to automotive software," In *Testing software and systems*, vol. 9447, pp. 141-157, 2015

[3] Y. Kamei, et al., "A large-scale empirical study of just-in-time quality assurance," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 7 57-773, June 2013

[4] M. Borg, et al., "SZZ Unleashed: An Open Implementation of the SZZ Algorithm," In *Proc. of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation*, pp.7-12, 2019

인공지능 의료기기 소프트웨어의 표준 준수를 위한 개발자 관점 품질관리 프로세스 정의

김동엽[1] 박예슬[1] 이병정[2] 이정원[1]
[1]아주대학교 전자공학과 [2]서울시립대학교 컴퓨터과학과
{dongdongy, yeseuly777, jungwony}@ajou.ac.kr[1], bjlee@uos.ac.kr[2]

Defining Quality Management Process for Complying with the Medical Device Software Standards for the Perspective of AI Software Developer

DongYeop Kim[1] Ye-Seul Park[1] Byungjeong Lee[2] Jung-Won Lee[1]
[1]Department of Electrical and Computer Engineering, Ajou University
[2]Department of Computer Science and Engineering, The University of Seoul

요 약

인공지능 의료기기 소프트웨어 개발자는 제품의 인허가를 위해 표준을 준수해야 하지만, 인공지능 기술에 대한 국제표준은 제정 중에 있기 때문에 의료기기 소프트웨어에 대한 국제표준인 IEC 62304를 준수해야 한다. 그러나 IEC 62304는 인공지능 관점의 생명주기 프로세스를 다루지 않으며, 개발자는 표준의 어느 명세를 준수해야할지, 산출물을 작성하기 위해 어떤 품질관리 방법을 적용할지 판단해야하는 어려움이 있다. 본 논문은 IEC 62304에 부합하도록 인공지능 의료기기 소프트웨어의 품질관리를 수행하는 개발자 관점의 프로세스를 제안한다. 또한 인공지능 모델의 학습 이전부터 전 주기에 걸쳐 품질을 관리하는 품질관리 방법을 제안한다. 마지막으로는 이를 실제 연구개발에 도입하여 적용 가능성을 확인한다.

1. 서론

의료기기 소프트웨어의 인허가를 위해서는 국제표준인 “IEC 62304, Medical Device Software – software life cycle processes”를 준수해야하며[1], 수명주기의 모든 순서에 대해 연구개발 프로세스를 수립하고 문서 산출물을 기록해야 한다. 인공지능 의료기기 소프트웨어도 단독형 및 임베디드 SW의 형태로 제공되기 때문에 IEC 62304를 준수해야 한다.

인공지능 의료기기 소프트웨어 개발자가 표준을 준수하기 위한 어려움은 두 가지로 설명할 수 있다. 첫째, 개발자가 임의로 인공지능의 특성을 반영하는 생명주기 프로세스를 수립하고 산출물을 작성해야 하며, 이들이 IEC 62304의 어느 명세를 만족시키는 것인지 연결해야 한다. 인공지능 의료기기 소프트웨어에는 기존의 테스팅, 성능 평가 기법과는 다른 접근이 필요하며, 생명주기에도 새로운 데이터를 학습하며 지속적으로 모델을 업데이트할 수 있는 특성을 반영해야 한다. 이를 소개하는 인공지능 기술의 국제표준은 제정 중에 있기 때문에 의료 규제를 만족시키기 위한 여러 연구와 문헌이 발간되고 있지만[2-3], 이들은 IEC 62304를 기준으로 하지 않기 때문에 개발자 입장에서 여전히 추상적이다.

둘째, IEC 62304에 의해 인공지능 모델의 학습 이후에 모델 성능을 검증하는 것만이 아닌 학습 이전부터 전 주기 동안의 품질 관리를 입증해야 한다. 모델의 성능을 평가하는 연구는 많이 이루어져 있지만[4], 고품질의 데이터를 확보하기 위한 데이터의 품질 메트릭과 측정 방법에 대한 연구는 미흡하다.

따라서 본 논문은 인공지능 의료기기 소프트웨어 개발자가 의료기기 소프트웨어에 대한 표준인 IEC 62304를 준수할 수 있는 품질관리 프로세스를 제안한다. 먼저 인공지능 의료기기 소프트웨어의 생명주기를 정의하고, IEC 62304의 명세를 분석하여 인공지능 기술에 대해 추가적인 활동이 필요한 것을 추출한다. 이 결과를 표로 요약하여 인공지능 의료기기 소프트웨어에 대한 모든 프로세스와 산출물을 IEC 62304의 명세에 연결시킬 수 있게 한다, 다음으로, 표준의 분석 결과를 기반으로 인공지능 의료기기 소프트웨어의 산출물 작성을 위한 품질관리 프로세스를 정의한다. 개발 초기 단계부터 데이터 수집 절차를 갖추고, 학습 이전에 데이터 품질을 검증하며, 학습 이후에 인공지능 모델의 성능을 평가할 수 있게 한다. 마지막으로, 본 논문의 적용 사례로서 위장관 교차점 식별을 위한 인공지능 의료기기 소프트웨어의 품질관리 결과를 살펴본다.

2. 인공지능 의료기기 소프트웨어의 표준 준수 방안

본 장에서는 인공지능 의료기기 소프트웨어의 생명주기를 정의하고 IEC 62304의 태스크를 분석하여 개발자가 표준에 근거하여 인공지능 기술을 설명할 수 있도록 안내한다.

2.1. 인공지능 의료기기 소프트웨어 생명주기 정의

IEC 62304는 5개의 연구개발 프로세스를 수립하도록 요구한다(개발, 위험관리, 형상관리, 결함관리, 유지보수).

1) 본 논문은 산업통상자원부 및 한국산업기술진흥원의 창의산업기술개발기반구축사업의 일환으로 수행하였음(N0002312, 디지털 헬스케어 소프트웨어 시험평가센터 구축)

데이터의 품질은 모델의 임상적 유효성과 직결되기 때문에 개발자는 데이터 관리 및 품질에 대한 절차를 모든 연구개발 프로세스에 반영할 필요가 있다. 또한 인공지능 모델은 제품의 출시 이후에도 신규 유입되는 임상데이터를 학습하여 인공지능 모델을 실시간으로 개선할 수 있는 특징이 반영되어야 한다. 본 논문은 이 특성을 생명주기로 제안한다.

2.2. 인공지능 기술 관점 표준 태스크(task) 분류

본 절은 연구개발 전 주기에 대한 프로세스를 다루는 방대한 범위의 표준 명세 중에서 인공지능 개발자가 집중해야 할 부분을 안내하기 위해 표준 명세 중 인공지능과 관계가 있는 것을 추출한다. 명세는 두 가지 유형으로 분류되며, '유지' 유형의 경우 기존 요구사항 이외에 별도로 수행할 내용이 없는 것이며, '확장' 유형의 경우 인공지능의 특성을 나타내는 추가적인 활동과 문서가 필요함을 의미한다.

2.3. 표준 태스크(task) 속성 추출

본 절에서는 표준의 각 태스크마다 속성을 추출하여 개발자가 본인이 수행해야 하는 태스크를 범위를 직관적으로 파악할 수 있게 한다. 태스크 속성 유형은 그림 1과 같이 3가지이며, 안전 등급, 강제 수준, 문서화 여부가 있다.

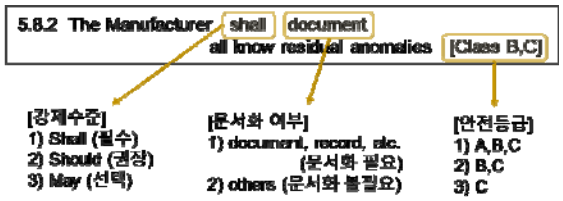


그림 1. 태스크 속성 체계

2.4. 인공지능 의료기기 소프트웨어의 표준 준수 방안

2.2절과 2.3절의 태스크 분석 결과를 기반으로 IEC 62304의 정보를 인공지능 ㄴ의료기기 소프트웨어의 관점에서 표의 형태로 제공한다. 적응형 혹은 고정형에 따라 확장이 필요한 범위가 다르며, 개발자는 각 태스크를 수행하며 '유지' 및 '확장' 유형에 따라 인공지능 기술에 대한 내용을 기존의 산출물 목록에서 기재하거나 새로운 산출물로 작성해야 한다.

3. 인공지능 의료기기 소프트웨어의 품질관리 프로세스

본 장에서는 2장의 인공지능 의료기기 소프트웨어의 표준 준수 가이드라인을 따를 때, 태스크 수행에 필요한 인공지능 기술의 품질관리 절차를 프로세스의 형태로 제안한다.

3.1. 개발자 관점 품질관리 프로세스

IEC 62304의 인공지능 의료기기 소프트웨어 품질 관련 태스크와 개발자 관점의 V모델의 각 순서를 알맞게 연결한다. 인공지능 모델은 소프트웨어의 단위 요소(unit component)로 간주한다. 본 논문이 제안하는 품질관리 절차는 그림 2와 같이 크게 '데이터 준비 과정', '데이터 품질 검증', '모델 성능 평가', 세 단계로 구성된다.

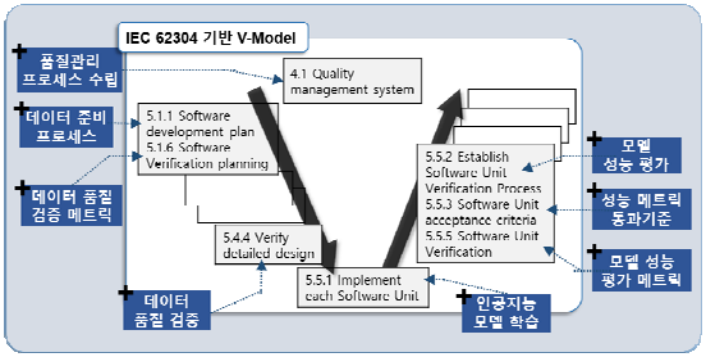


그림 2. 개발자 관점 AI 의료기기 SW 품질관리 프로세스

3.2. 품질관리 메트릭 및 적용 가이드라인

본 논문의 품질관리 프로세스는 그림 2와 같이 모델의 학습 전후에 걸쳐 진행된다. 인공지능 모델을 학습하기 전에는 데이터를 체계적으로 준비하고, 메트릭을 통해 정량적인 수치로 데이터의 품질을 검증한다. 학습 이후에는 인공지능 분야에서 널리 사용되는 모델 성능 평가 메트릭을 활용하여 모델의 성능을 평가한다. 데이터 품질 메트릭은 다음과 같다.

-데이터 정상성: 학습에 도움이 되는 정상 데이터와 모델 성능을 저해할 수 있는 노이즈 데이터의 비율을 의미한다.

-학습 적합성: 레이블링이 올바르게 되어있거나, 개발자가 레이블링 가능한 데이터에 대한 비율을 의미한다.

-해부학적 완전성: 데이터 세트에서 레이블링과 관련되어 필수적으로 관찰되어야 하는 해부학적 요소가 관찰되는 비율을 해부학적 완전성이라 한다.

4. 적용 사례

IEC 62304 체크리스트와 품질관리 산출물 예시를 통해 적용 가능성을 확인한다. 대상은 캡슐내시경 영상에서 위장관 교차점을 분류하기 위한 딥러닝 기반 인공지능 네트워크이다.

5. 결론

본 논문은 인공지능 의료기기 소프트웨어 개발자가 의료기기 소프트웨어 표준인 IEC 62304를 준수할 수 있도록 한다. 개발자는 인공지능 의료기기 소프트웨어의 전반적인 품질 관리를 수행하고 양질의 산출물을 작성할 수 있으며, 국제표준을 준수함으로써 인허가 자격을 갖출 수 있다.

6. 참고문헌

[1] IEC. "IEC 62304, Medical device software-software life-cycle processes". 2015

[2] 식품의약품안전처. "빅데이터 및 인공지능(AI) 기술이 적용된 의료기기의 허가·심사 가이드라인". 2019.10

[3] FDA. "Proposed Regulatory Framework for Modifications to Artificial Intelligence/Machine Learning (AI/ML)-Based Software as a Medical Device (SaMD)". 2019.06

[4] Gulshan, et al. "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs". Jama, 316(22). 2016

지능형 군집 무인체계의 개발을 위한 모델링&시뮬레이션 방법론

김희수[1] 성영화[2]
리얼타임비쥬얼㈜[1], 국방과학기술연구소 국방첨단기술연구원[2]
heemanz@gmail.com[1], yhsung@add.re.kr[2]

A Modeling & Simulation Methodology for Developing an Intelligent Swarm of Unmanned Vehicle Systems

Hee-Soo Kim[1] Young Hwa Sung[2]
REALTIMEVISUAL Inc.[1]
Institute of Defense Advanced Technology Research, Agency for Defense Development[2]

요 약

인공지능 및 무인체계 기술의 발전으로 국방 분야의 작전 및 임무 수행에 비용효과적이면서 운용성이 뛰어난 군집 무인체계를 활용하는 연구가 진행되고 있다. 하지만 대규모 군집 무기체계의 개발은 본질적으로 비용적 또는 물리적인 제약으로 인해 실 환경에서의 운영 및 실험을 통한 개발이 어려운 문제를 가지고 있다. 본 연구는 이와 같은 제약 상황에서 군집 무인체계를 효과적/효율적으로 개발하기 위한 모델링&시뮬레이션 시스템과 방법론을 제안한다. 제안하는 시스템과 방법론에서 개발자는 에이전트 지향 소프트웨어 공학 기법을 활용하여 군집 무인체계의 동역학과 제어기, 탑재 장비 등을 상위 수준의 임무 및 과업, 군집 행동, 대형 유지, 단위 행동 등의 행위를 분리하여 개발한다. 또한, 개발자가 향후에 기계학습 기법을 적용한 군집 무인체계를 개발할 수 있도록 지원하는 학습 데이터의 생성 및 강화학습 환경으로써 제안 시스템과 방법론을 활용하는 방안에 대해 논의한다.

1. 서 론

인공지능 및 자율화 기술의 발전으로 무인체계의 지능화/자율화 연구가 활발히 진행되고 있다. 특히, 국방분야에서 대한민국 육군은 첨단 미래군의 핵심 전력으로 드론봇 전투단을 창설하여 감시정찰을 위한 훈련을 수행하고 있다. 또한, 미국의 DARPA(Defense Advanced Research Projects Agency)는 Squad X 프로젝트를 통해 해병대원들이 자율 로봇들로부터 수집된 정보를 활용하여 임무를 수행하는 연구를 진행중이다. 더 나아가 DARPA의 OFFSET(OFFensive Swarm-Enabled Tactics) 프로그램은 복잡한 시가전 환경에서 다양한 임무를 수행하기 위해 250대의 자율 무인체로 구성된 대규모 군집을 활용하는 전술도출에 대한 연구를 진행 중이다.

이와 같이 무인체계 기술이 발전하는 상황에서, 대규모의 무인체계의 활용은 기존의 단일 무인체계에 비해 정보의 양과 획득 시간, 작전 수행 영역 등의 측면에서 더 많은 이점을 가진다. 하지만, 대규모의 무인체계를 효과/효율적으로 제어하기 위한 기술적 복잡도는 매우 높을 수 밖에 없다. 게다가, 대규모 무인체계를 개발하기 위해서는 넓은 실험 공간과 같은 환경적인 요소와 대규모 무인체계의 생산 비용의 문제를 해결해야 하며, 학습 데이터의 확보 및 무인체계의 안정성과 신뢰성이 보장되어야 한다.

본 연구는 지능형 군집 무인체계의 개발에 있어서 직면하게 되는 물리적 또는 비용상의 문제와 함께 학습 데이터 확보 문제로부터 개발자들을 돕기 위한 방법의 일환으로 모델링&시뮬레이션(M&S) 기술을 활용하여 지능형 군집 무인체계를 모델링하고 운용 및 실험이 가능한 시스템 및 개발 방법론을 제안한다. 제안하는 시스템은 기계학습 측면에서 지능형 군집 무인체계의 강화학습을 수행하기 위한 환경으로 사용을 지원하고, 시뮬레이션의 수행 결과로 생성되는 높은 충실도의 영상 데이터 및 지형, 지물, 무인체계들에 대한 부가 정보는 군집 무인체계의 학습 데이터로 사용되어 지도/비지도학습을 지원한다. 이와 같은 지능형 군집 무인체계를 개발하기 위한 M&S 시스템의 주요 요구 기능은 다음과 같다.

- 모델링 인프라스트럭쳐: 대규모 군집 무인체계의 임무/과업/행동과 제어기, 동역학, 탑재 장비 등의 체계적인 모델링 지원
- 분산/병렬 시뮬레이션: 대규모 군집 무인체계의 시뮬레이션과 학습을 지원
- 고품질 학습 영상 생성: 영상처리를 위한 학습용 데이터 생성을 지원

- 다양한 시나리오 자동 생성: 조건 파라미터를 기반으로 다양한 상황에서 학습 데이터를 수집하기 위한 시나리오 생성
- 학습 데이터 생성: 시뮬레이션 결과 데이터 및 이를 기반으로 생성된 실사 수준의 컴퓨터 영상을 활용해서 효과/효율적인 학습데이터의 생성을 지원
- 다양한 인공지능 기법과 상호작용: 다양한 추론 엔진 및 기계 학습 개발 프레임워크와 연동 지원
- 행위 모델의 개발 프로세스: 군집 임무/과업/행동 수행 모델 개발 과정부터 PILS(Process In the Loop Simulation) 적용 실험 후, 실 무인체계에 적용하는 과정을 지원

제안하는 방법론은 에이전트 지향 소프트웨어 공학(AOSE, Agent-Oriented Software Engineering) 기법[1]을 적용하여 요구분석 단계에서부터 구현, 테스트의 단계까지 일련의 과정을 설명한다. 특히, 본 연구에서 제안하는 모델링 인프라스트럭처를 구성하고 있는 요소들인 군집 무인체계의 임무 수행 그룹과 각 무인체계 모델, 컴포넌트 모델 등을 개발하는 방법에 초점을 맞춘다.

본 논문은 다음과 같이 구성된다. 2장은 먼저 관련연구에 대해 설명한다. 3장은 지능형 군집 무인체계 개발을 위한 모델링&시뮬레이션 시스템의 전체 구조를 설명하고, 고품질 학습 영상 생성과 다양한 시나리오 자동 생성, 학습 데이터 생성, 다양한 인공지능 기법과 상호작용 방법을 간략하게 설명한다. 4장은 군집 무인체계를 위한 모델링 인프라스트럭쳐와 시뮬레이션 실행 서비스에 대해 설명한다. 5장은 제안 시스템의 적용 분야에 대해 설명하고, 6장은 제안하는 시스템에서 군집 무인체계를 개발하는 방법을 설명한다. 7장은 군집 무인체계의 개발에서 기계학습에 활용 방안에 대해 논의하고, 마지막으로, 8장은 본 연구의 결론과 향후 연구 방향에 대해 설명한다.

2. 관련 연구

본 절은 현재 군집 무인체계와 관련된 오픈소스 기반의 시뮬레이션 소프트웨어에 대해 설명한다. 본 연구를 통해 조사한 설치 및 실행 가능한 소프트웨어의 목록과 설명은 다음과 같다.

- AirSim [1]: 마이크로소프트 사에서 Unreal과 Unity 게임 엔진을 기반으로 만든 대표적 자율주행 기계학습 환경[2], 지형과 지물 컨텐츠의 정밀도에 따라 실사에 가까운 학습용 카메라 영상 생성을 지원
- Webot [2] : 드론을 포함한 로봇의 모델링과 프로그래밍, 시뮬레이션을 위한 환경을 제공
- CoppeliaSim [3] : 로봇 개발 환경과 통합된 시뮬레이션 환경을 제공
- Gazebo[4] : 로봇과 센서 애플리케이션을 위한 무료 소프트웨어 도구, 모델을 구동하기 위해 타 소프트웨어 (Player와 ROS)와 연동을 지원
- MAgent[5]: 멀티 에이전트 강화학습 연구 플랫폼으로 비교적 단순한 이동 논리를 사용하며, 다수의 에이전트 시뮬레이션을 지원
- SLM Lab [6] : Pytorch 기반으로 모듈화된 심층 강화학습(Deep Reinforcement Learning)을 지원하는 플랫폼. OpenAI Gym, Unity ML Agent와 연동을 지원
- Deep RL for Swarm Systems[7]: 멀티 에이전트 군집 지능 구현. 2D 환경에서 동일 특성을 가지고 있는 에이전트들을 시뮬레이션
- Multi-Agent-Learning-Environments [8] : 단순한 구조의 멀티에이전트 강화학습 플랫폼

상기 소프트웨어는 각각 다른 목적에 의해 개발된 것들로 그 외형 및 기능에서 많은 차이를 보인다. 하지만, 각 소프트웨어는 군집 무인체계를 개발하기 위한 M&S 시스템의 개발에 유용한 기능들을 제공한다. 먼저, 앞의 네 개의 소프트웨어는 3D 공간에서 각 무인체계의 상세한 동작에 대한 관찰이 가능하며, AirSim이나 Webot은 카메라 영상 생성 기능을 통해 무인체계 주변의 환경 데이터로 활용이 가능하다. 다음에 열거된 네 개의 소프트웨어는 상위의 추상화 수준(충실도 및 해상도가 낮은)에서 군집의 전반적인 행동을 관찰할 수 있는 장점과 함께 강화학습에 특화된 특징을 보여준다. 본 연구는 이와 같은 특징들을 반영하여 1장에 언급된 지능형 군집 무인체계 개발을 위한 M&S 시스템의 요구 기능으로 도출하였다.

군집 무인체계의 개발 방법 관점에서, 본 연구가 제안하는 개발 방법론은 AOSE를 기반으로 이를 확장하고 구체화한다. AOSE는 다양한 에이전트들로 구성된 복잡한 시스템의 개발에 있어서 요구 분석부터 구현까지의 프로세스를 지원하며[3], M&S 분야에서 시뮬레이션 객체들을 모델링하기 위해 AOSE 기반의 개발 방법론을 적용한 연구[4]가 존재한다. 본 연구는 AOSE를 기반으로 군집 무인체계를 모델링하고, 그 구성요소 중 일부인 행위 모델을 실 무인체계에 탑재하여 테스트 하는 과정을 추가/확장한다.

[1] https://github.com/microsoft/AirSim
[2] https://cyberbotics.com
[3] http://coppeliarobotics.com
[4] http://gazebosim.org
[5] https://github.com/geek-ai/MAgent
[6] https://github.com/kengz/SLM-Lab
[7] https://github.com/LCAS/deep_rl_for_swarms
[8] https://github.com/Bigpig4396/Multi-Agent-Reinforcement-Learning-Environment

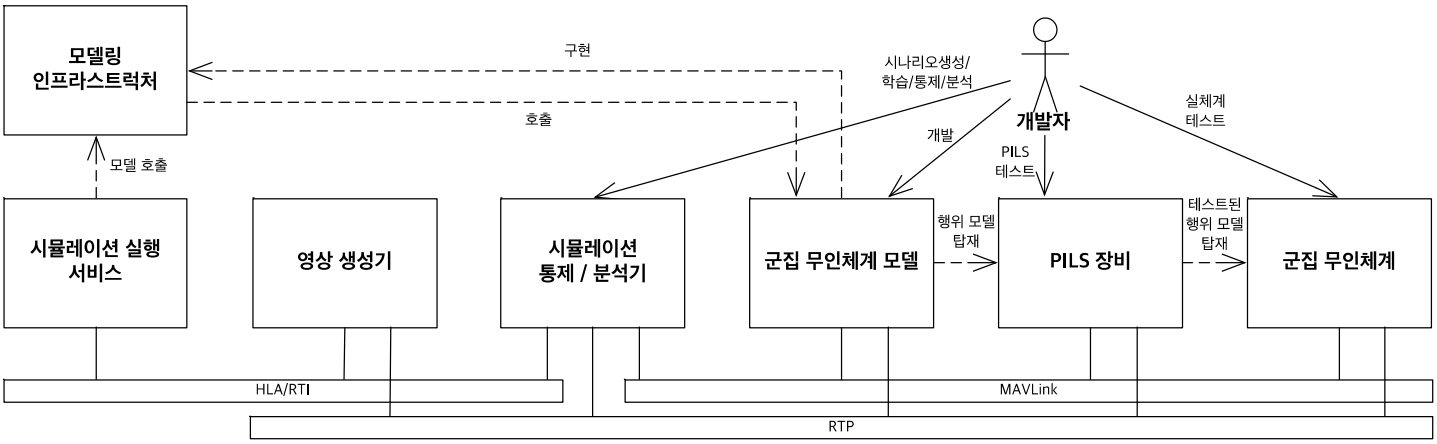


그림 1 군집 무인체계 모델링&시뮬레이션 시스템 구성도

3. 군집 무인체계 모델링&시뮬레이션 시스템

제안 시스템은 1장에 언급된 요구 기능을 만족하기 위해 1)모델링 인프라스트럭처와 2)시뮬레이션 실행 서비스, 3)영상 생성기, 4)시뮬레이션 통제/분석기, 5)군집 무인체계 모델, 6)PILS 장비, 7)군집 무인체계의 핵심 요소들로 구성된다. 제안하는 군집 무인체계 M&S 시스템은 이 요소들을 통해 개발자의 군집 무인체계의 개발을 지원한다.

그림 1은 개발자를 포함한 제안 시스템의 구조 및 관계를 설명한다. 다음의 각 절은 개발자와 핵심 요소들을 설명한다.

3.1. 개발자

군집 무인체계 M&S 시스템의 사용자로 실 세계에서 운용이 가능한 군집 무인체계의 개발을 목적으로 한다. 개발자는 먼저 모델링 인프라스트럭처를 기반으로 군집 무인체계의 시뮬레이션 모델을 개발한다. 그 다음 개발된 모델을 사용하여 시뮬레이션을 수행한다. 이를 위해 개발자는 시뮬레이션 통제/분석기 사용해서 시나리오를 작성하고, 시뮬레이션을 통제, 시뮬레이션 데이터를 분석, 필요한 학습 데이터 생성, 군집 무인체계 모델의 학습을 수행한다.

또한, 개발자는 개발(또는 학습)된 군집 무인체계의 행위 모델(6.3절 참조)을 PILS에 탑재하고 시뮬레이션과 연동하여 테스트를 수행한다. 마지막으로, 개발자는 PILS 장비에서 테스트된 군집 행위 모델을 실 군집 무인체계에 탑재하고 실 세계에서 테스트를 수행한다.

3.2. 모델링 인프라스트럭처

모델링 인프라스트럭처는 군집 무인체계 모델을 개발하기 위한 프레임워크로 시뮬레이션 실행 서비스의 기능(이벤트 및 시뮬레이션 시간 처리, 지형/지물과 충돌 확인, 주변 시뮬레이션 객체 조회 등)을 호출하는 기능과 함께 군집 무인체계를 구성하는 컴포넌트들(동역학 및 제어기, 통신 장비 및 센서, 임무/과업을 위한 행위 모델 등)의 기본 분류체계와 이들 사이의 관계를 정의한다. 개발자는 이를 기반으로 상속 및 구현 메커니즘 등을 통해 군집 무인체계의 모델을 개발한다.

3.3. 시뮬레이션 실행 서비스

시뮬레이션 실행 서비스는 시뮬레이션을 수행하는 주체로 시뮬레이션 객체 관리, 시간 관리, 이벤트 관리, 환경 관리 기능을 제공한다. 이를 기반으로 시뮬레이션 진행 과정에서 적절한 타이밍에 군집 무인체계 모델의 시뮬레이션 기능을 호출하고, 시뮬레이션 세계에 대한 조회 기능(무인체계의 위치 및 자세, 지형/지물과의 충돌 여부, 모의 시간)을 제공한다. 또한, 시뮬레이션 실행 서비스는 대규모 군집 무인체계의 시뮬레이션을 지원하기 위해 일종의 시뮬레이션 클라우드로 구현되며, HLA/RTI(High Level Architecture / Real-Time Infrastructure)를 통해 분산 시뮬레이션 기능을 제공한다.

시뮬레이션 실행 서비스가 실행하는 시뮬레이션은 시뮬레이션 통제/분석기에 의해 HLA/RTI를 통해 제어되며, 시뮬레이션 수행 과정에서 생성되는 데이터(위치/자세)는 각각 영상 생성기와 시뮬레이션 통제/분석기로 전송되어 카메라 영상의 생성과 시뮬레이션의 모니터링에 사용된다.

3.4. 영상 생성기

영상 생성기는 시뮬레이션 실행 서비스로부터 HLA/RTI를 통해 수신한 무인체계들의 정보를 기반으로 상용 게임 엔진인 Unreal Engine을 통해 각 무인체계가 카메라를 통해 센싱하는 영상을 생성하고, 그 결과를 RTP(Real-time Transport Protocol)을 통해 무인체계의 영상 처리 모델 또는 PILS 장비로 전송한다. 또한, 이 영상 정보는 RTP를 통해 시뮬레이션 통제/분석기로

전송되어 시뮬레이션 상태 모니터링 및 영상 처리용 학습 데이터를 생성하기 위해 사용된다.

3.5. 시뮬레이션 통제/분석기

군집 무인체계 M&S 시스템에서 시뮬레이션 통제/분석기는 기본적으로 개발자에게 시나리오 작성 및 시나리오 자동 생성, 시뮬레이션 세계 모니터링 및 통제, 시뮬레이션 결과 분석(학습 데이터 생성 기능을 포함) 기능을 제공한다. 추가로 시뮬레이션 통제/분석기는 MAVLink를 통해 군집 무인체계 모델과 PILS 장비, 실 군집 무인체계를 모니터링하고 명령을 전달하는 기능을 제공한다.

3.6. 군집 무인체계 모델

군집 무인체계 모델은 개발자에 의해 모델링 인프라스트럭처를 기반으로 개발되는 실 군집 무인체계의 모델이다. 따라서 군집 무인체계 모델은 실 무인체계의 물리적 특성을 반영한 동역학과 제어기 모델을 비롯하여, 통신 모듈 및 센서 등의 탑재 장비와 임무/과업을 달성하기 위한 행위 모델을 포함한다.

군집 무인체계의 모델은 시뮬레이션 세계에서의 물리적 위치 및 자세 등의 정보를 제공한다. 이 정보는 시뮬레이션 실행 서비스의 HLA/RTI 연동 기능을 통해 군집 무인체계 M&S 시스템내에서 공유되며, 학습 데이터 생성시 ground truth 값으로 사용된다.

또한 군집 무인체계의 모델은 실제 군집 무인체계가 사용하는 MAVLink 프로토콜 통해 타 모델 및 타 체계와 연동한다. 이와 같이 개발하는 목적은 기 개발된 제어/행위 모델이 수정없이 PILS 장비나 실 군집 무인체계에 탑재되어 사용 가능하도록 하기 위함이다.

3.7. PILS 장비

PILS 장비는 실제 군집 무인체계가 사용하는 동일한 연산 컴퓨터를 사용하며, 군집 무인체계 모델의 임무/과업을 수행하는 행위 모델을 탑재하여 실행시킬 수 있는 소프트웨어를 설치하여 운영된다. 개발자는 실 무인체계에 개발된 행위 모델의 적용 및 테스트에 앞서 PILS 장비를 사용하여 개발한 행위 모델의 기능을 테스트 한다.

3.8. 군집 무인체계

군집 무인체계는 실 세계에서 운용되는 무인체계를 의미한다. 본 연구에서 군집 무인체계는 개발자가 개발한 행위 모델과 제어 모델의 기능을 테스트 하기 위해 사용된다.

4. 모델링 인프라스트럭쳐 & 시뮬레이션 실행 서비스

본 장은 3.2와 3.3절에서 각각 설명한 모델링 인프라스트럭처와 시뮬레이션 실행 서비스에 대해 상세히 설명한다. 그림 2는 클래스 다이어그램을 사용하여 모델링 인프라스트럭처와 시뮬레이션 실행 서비스의 구성 요소와 이들간의 관계를 설명한다.

개발자가 시나리오를 로딩하여 시뮬레이션을 시작할 때, 시뮬레이션 실행 서비스에서 시뮬레이션 엔진을 구성하고 있는 객체 관리자는 군집 무인체계의 모델(즉, 개체 모델 또는 그룹 모델을 상속해서 개발자가 개발한 구현 모델)로부터 시나리오에 명시된 무인체계들을 개체화된다. 생성된 무인체계 객체는 시뮬레이션의 진행 과정에서 시뮬레이션 객체가 참조하고 있는 시뮬레이션 서비스를 통해 이벤트 및 시간 진행에 대한 처리를 수행한다. 다음의 절들은 모델링 인프라스트럭쳐와 시뮬레이션 실행 서비스를 각각 설명한다.

4.1. 모델링 인프라스트럭쳐

모델링 인프라스트럭처는 10개의 핵심 클래스로 구성된다. 개발자는 상기 클래스들을 상속/구현하여 개발하고자 하는 군집 무인체계를 모델링 한다. 다음은 각 클래스를 설명한다.

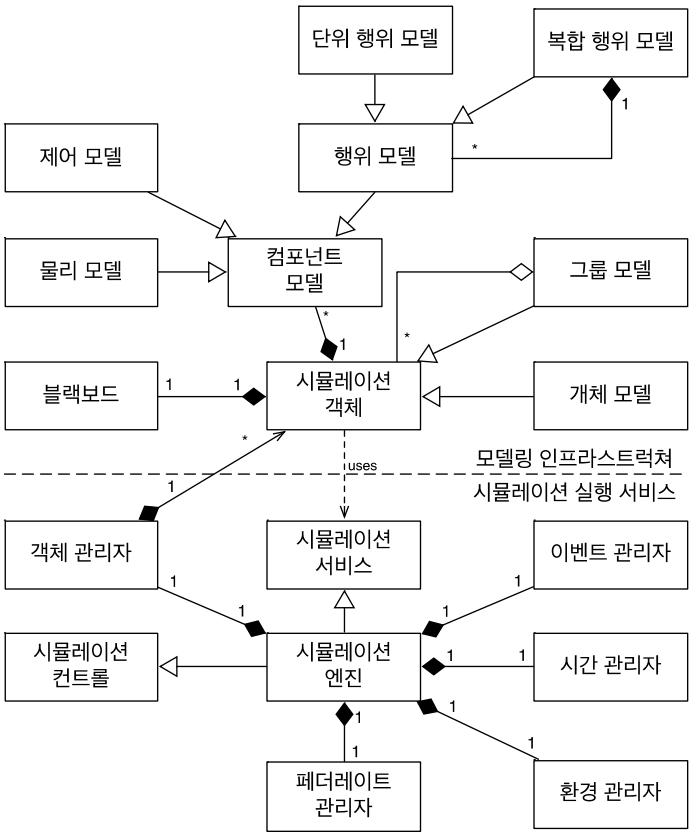


그림 2 모델링 인프라스트럭처와 시뮬레이션 실행 서비스

- 시뮬레이션 객체: 시뮬레이션되는 모든 논리적/물리적 개체의 추상 클래스로 시뮬레이션 서비스가 제공하는 관심 시간/이벤트의 등록 및 처리 기능을 제공, 컴포넌트 모델들을 가질 수 있음.
- 개체 모델: 단일 무인체계를 시뮬레이션 하기 위한 추상 클래스, 물리와 제어, 행위 모델들을 컴포넌트 모델로 가질 수 있음.
- 그룹 모델: 운영자에 의해 동적으로 생성되는 논리적 임무 수행 그룹을 시뮬레이션 하는 추상 클래스, 제어 및 행위 모델을 컴포넌트 모델로 가질 수 있으며, 그룹에 속한 시뮬레이션 객체(타 그룹/개체 모델)들이 물리 모델의 역할을 수행.
- 블랙보드: 시뮬레이션 객체를 구성하는 컴포넌트 모델들 간 이벤트 전달 및 정보 공유 기능을 제공
- 컴포넌트 모델: 시뮬레이션 객체를 구성하는 해당 분야의 물리적/논리적 기능을 구현하기 위한 추상 클래스
- 제어 모델: 무인체계의 물리 모델과 밀접하게 연관되어 무인체계의 움직임을 제어하기 위한 추상 클래스, 예를 들면, 드론에 사용되는 제어 컴퓨터인 FCS(Flight Control System) 역할을 수행.
- 물리 모델: 제어 모델의 제어 입력으로부터 무인체계의 동역학과 통신 장비, 센서 등의 탑재 장비를 시뮬레이션하는 모델들의 추상 클래스.
- 행위 모델: 단일 무인체계 또는 군집 무인체계가 수행해야 하는 임무 및 과업, 행동의 추상 클래스
- 단위 행위 모델: 제어 모델의 기능을 직접 호출하여 특정 행동을 수행하는 모델들의 추상 클래스
- 복합 행위 모델: 단위 행위 모델의 조합을 통해 복잡한 임무/과업/행동을 수행하는 모델들의 추상 클래스

4.2. 시뮬레이션 실행 서비스

시뮬레이션 실행 서비스는 8개의 핵심 인터페이스 및 클래스로 구성된다. 다음은 각각을 설명한다.

- 시뮬레이션 서비스: 개발자가 개발한 시뮬레이션 객체에게 시뮬레이션 세계에 대한 정보의 제공 및 이벤트/시간 진행에 대한 처리 함수를 등록하고 호출해주는 기능을 제공하는 인터페이스.
- 시뮬레이션 컨트롤: 시뮬레이션 통제/분석기에게 시뮬레이션을 통제(시작 및 일시정지, 재개, 종료)하는 기능을 제공하는 인터페이스.
- 시뮬레이션 엔진: 시뮬레이션 서비스와 시뮬레이션 컨트롤 인터페이스를 구현한 클래스, 객체 및 시간, 이벤트, 환경, 페더레이트 관리자들로 구성됨.
- 객체 관리자: 시뮬레이션 객체들의 라이프 사이클 및 상태를 관리하고, 시뮬레이션 서비스를 통해 시뮬레이션 객체에게 타 시뮬레이션 객체에 대한 정보를 제공하는 클래스.
- 이벤트 관리자: 시뮬레이션 세계에서 발생하는 이벤트에 대해 관심을 등록하고, 이벤트 발생시 등록된 처리 함수를 호출하는 클래스.
- 시간 관리자: 시뮬레이션 시간의 논리적 진행을 관리하고 페더레이트 관리자와 연동하여, 분산 시뮬레이션 시, 원격의 시뮬레이션 실행 서비스와 시간을 동기화하는 클래스
- 환경 관리자: 시뮬레이션 되는 무인체계와 지형/지물과의 충돌 및 접근 등의 상호작용 정보를 제공하는 클래스
- 페더레이트 관리자: HLA/RTI를 상에서 분산 시뮬레이션할 때, 페더레이션 내에서 시뮬레이션 엔진들이 각각 하나의 페더레이트로 동작[5]하도록 지원하는 클래스.

5. 적용 분야

현재 군집 무인체계를 활용하고 있는 분야는 서론에서 언급한 Squad X 프로젝트와 OFFSET 프로그램과 같이 정찰 및 첩보 수집 분야이다. 본 연구는 이들보다 더 고도화된 “공격용 무인체계(즉, UAV(Unmanned Aerial Vehicle) 드론 공격)에 대응하는 주요 거점에 대한 방어 임무”를 하나의 적용 도메인으로 군집 무인체계를 설명한다.

대표적인 드론 공격 사례는 2019년도에 UAV를 사용하여 사우디의 정유 시설을 공격한 사건이다. 정유 시설은 중요 기반구조(Critical Infrastructure)로 분류되는 중요한 시설로써 경제 및 생활 측면에서 사회 구성원들 전체에 영향을 주기 때문에 매우 중요하다. 이와 같은 이유로 드론 공격에 대한 대응은 이미 매우 중요한 이슈가 되었으며, 드론 격추용 무기로 방공포 및 레이저, 드론 등을 활용하는 다양한 방법들이 존재하나 그 효용성은 아직 검증이 필요한 상태이다. 본 연구는 중요 거점을 드론 공격으로부터 방어하기 위해 군집 무인체계를 활용하는 방법에 대한 연구의 일환으로 방어 시나리오를 설명한다.

중요 거점을 방어하기 위해서는 먼저 공격 드론의 진입 위치를 확인해야 한다. 이를 위해서 다수의 드론들은 중요 거점 주변의 정찰 임무를 수행할 필요가 있다. 드론이 수행하는 정찰 임무의 중요한 이슈는 배터리 또는 연료의 소모로 인한 비행 시간의 제약이다. 따라서, 드론들은 연료(또는 배터리)의 충전과 정찰 과업을 교대로 수행해야 한다. 정찰 임무를 수행 중, 일부 정찰 드론이 공격 드론을 탐지하면, 이 사실을 다른 드론들과 공유하고, 공격 드론의 예상 이동 경로를 추정하여, 현재 방어 드론들의 위치를 고려해 요격 지점을 예측해야 한다. 그 다음 다수의 방어 드론들이 요격 지점으로 도착 시간을 고려하여 최적의 방어 드론(또는 드론들)에 격추 과업을 드론에 할당하고,

드론은 요격 지점으로 이동하여 요격을 시도한다.

다음 장은 군집 무인체계를 활용하여 공격 드론에 대응하여 방어 임무를 수행하는 군집 무인체계의 개발 과정을 설명한다.

6. 군집 무인체계 모델의 개발 방법론

본 연구가 제안하는 군집 무인체계의 개발 방법론은 모델링 인프라스트럭처의 구성요소 중에서 개발자가 상속/구현해야 하는 그룹 및 개체, 행위, 물리, 제어 모델 별로 조금씩 다르게 적용된다.

표 1은 각 개발 요소에 적용되는 요구 분석에서부터 테스트까지 각 단계에서 사용되는 분석/개발 도구 및 산출물에 대한 전반적인 개념을 보여준다. 다음 절들은 각 모델의 개발 활동을 설명한다.

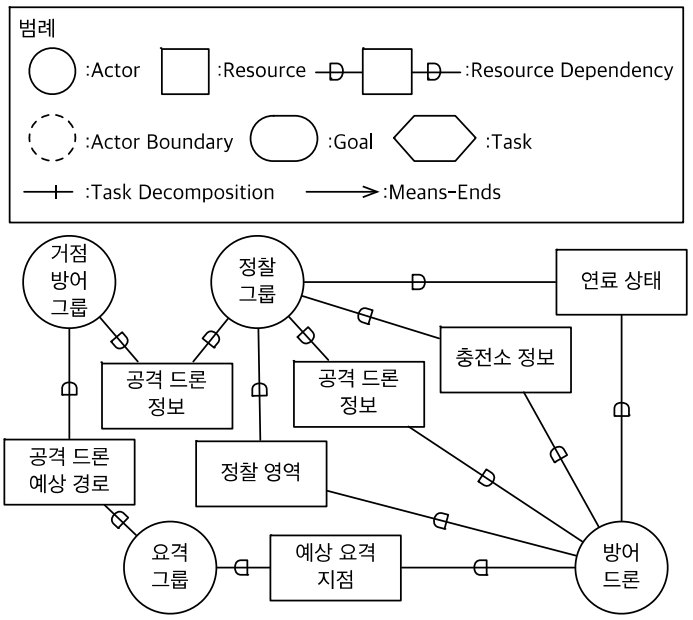
6.1. 그룹 모델의 개발

요구 분석 단계에서 사용하는 i* 프레임워크[6]의 전략적 의존성(Strategic Dependency) 모델은 액터(Actor)와 목표(Goal), 과업(Task), 자원(Resource)과, 액터들 사이에 존재하는 의존성을 식별을 하기 위해 사용된다. 또한 i* 프레임워크의 전략적 합리(Strategic Rationale) 모델은 액터가 과업을 수행하는데 있어서 그것의 의도를 이해하기 위해 사용된다. i* 프레임워크를 사용해서 분석된 위의 두 모델에서 군집 무인체계를 위한 그룹 모델은 어떤 정적인 조직을 표현하기 보다는 임무를 수행할 때 시시각각으로 그 구성을 변화하는 조직에 가깝다. 5장에서 설명된 적용 분야에서 최상위 수준에서 거점 방어 그룹과 그 하위의 정찰 그룹, 요격 그룹을 그룹 모델로 식별할 수 있다. 그리고 이 그룹들은 주어진 임무를 수행하기 위해 자체적으로 처리해야 하는 목표와 과업을 가지며, 컴포지션 관계에 의해 하위 목표와 과업으로 세분화된다. 그림 3의 a)와 b)는 각각 i* 프레임워크의 전략적 의존성과 합리 모델을 사용하여 적용 분야를 모델링한 예를 보여준다.

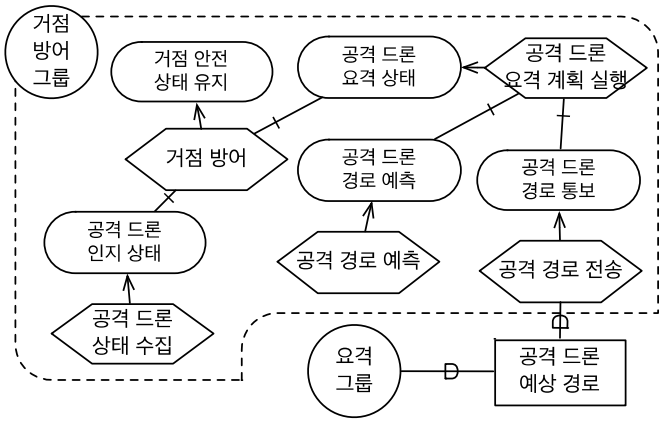
구조 설계 단계에서는 그룹 모델에 포함되는 하위 그룹(또는 무기체계를 의미하는 개체 모델) 모델과의 관계와 행위 모델로 모델링 된 목표 및 과업들이 클래스 다이어그램으로 명시된다.

표 1 개발 단계 별 분석/개발 도구(산출물)

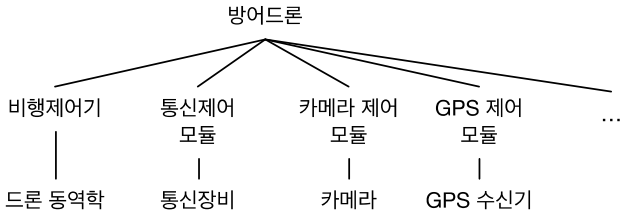
구분		그룹 모델	개체 모델	행위 모델	제어 모델	물리 모델
요구분석		i* (Actor, Goal, Task, Resource, Dependency, Decomposition, Means-end)		(Goal, Task, Decomposition, Dependency, Means-end)		
			Feature Mode (Feature)		(Feature, 입/출력 명세)	(Feature, 환경 고려사항 입/출력 명세)
구조 설계		UML (Class Diagram)	UML (Class Diagram)	UML (Class Diagram)	MATLAB Simulink (Simulink Model) 또는 UML	MATLAB Simulink (Simulink Model) 또는 UML
상세 설계		UML (Sequence Diagram)	UML (Sequence Diagram)	UML (Activity Diagram)		
구현		개발 IDE (Class)	개발 IDE (Class)	개발 IDE (Class)	MATLAB Code Generator 또는 개발 IDE (Class)	MATLAB Code Generator 또는 개발 IDE (Class)
테스트	시뮬레이션	O	O	O	O	O
	PILS	O	O	O PILS장비에 탑재	O	O
	실 무인체계	O 무인체계에 탑재	무인체계로 대체	O 무인체계에 탑재	O 무인체계에 탑재	무인체계로 대체



a) 중요 거점 방어를 위한 i* Strategic Dependency Model



b) 거점 방어 그룹의 i* Strategic Rationale Model



c) 방어 드론의 Feature Model

그림 3 적용 분야에서 i* 모델과 피처 모델의 예

상세 설계에서, 그룹의 행동은 행위 모델에서 개발되는 목표와 과업의 수행 절차로 결정되기 때문에, 그룹 모델의 상세 설계는 단순히 구조 설계에 명시된 구성 요소들의 라이프 사이클을 관리하는 절차가 시퀀스 다이어그램으로 표현된다.

구현 단계에서 개발자는 개발 IDE(Integrated Development Environment)를 사용하여 위 설계 내용을 코드로 구현한다. 시뮬레이션과 PILS 테스트 단계는 시뮬레이션 환경에서 타 모델들과 함께 그룹 모델의 기능을 검증한다. 실 무인체계 테스트 시에는 운영되는 군집 무인체계에 모델을 탑재하여 그 기능을 검증한다.

6.2. 개체 모델의 개발

개체 모델의 요구 분석 단계에서는 그룹 모델의 요구 분석에서 사용한 i* 모델이 재사용되며, 추가적으로 개발할 무인체계의 피처 모델(Feature Model)[7]을 명시한다. 이 모델에서 피처(Feature)는 무인체계의 구성 요소인 제어기를 비롯하여 동역학, 탑재장비, 센서 등을 의미하는 제어/물리 모델을 표현한다. 그림 3의 c)는 적용 분야에 사용되는 방어드론의 피처 모델을 보여준다. 방어드론은 물리적으로 드론 동역학, 통신장비, 카메라, GPS 수신기로 구성되고, 이들을 통제하기 위한 제어 모듈들을 포함한다.

구조 및 상세 설계에서는 그룹 모델의 개발 과정과 과 유사하게 클래스/시퀀스 다이어그램을 사용하여 설계한다. 이 때, 개발자는 무인체계를 구성하는 제어/물리 모델의 구현 클래스를 참조하여 설계한다.

구현 및 테스트 단계는 실 무인체계 테스트 시, 개체 모델이 실 무인체계로 대체된다는 점을 제외하고, 그룹 모델의 경우와 유사한 활동으로 진행된다.

6.3. 행위 모델의 개발

요구 분석 단계에서 행위 모델은 i* 모델을 통해 목표와 그것을 달성할 수 있는 과업들의 조합으로 식별된다. 이와 같은 조합은 목적-방법(Means-end) 관계로 식별된다. 과업은 다른 하위 행위 모델(하위 목표와 과업들의 조합)들로 구체화될 수 있다. 또한, 행위 모델의 상-하위 관계는 분해(Decomposition) 관계로 식별된다. 또한 의존성(Dependency) 관계는 타 시뮬레이션 객체(그룹/개체 모델)의 행위 모델을 수행하는 요청해야하는 기능을 나타낸다. 그림 3의 b)의 행위 모델의 요구 분석을 위한 전략적 합리 모델의 예를 보여준다. 이 모델에서 목표와 과업은 화살표로 표현되는 목적-방법 관계로 연결되고, 과업은 다시 분해 관계에 있는 하위 목표들로 구성된다. 또한, 이 전략적 합리 모델은 요격 그룹이 거점 방어 그룹에 공격 드론의 예상 경로를 의존하고 있음을 의존성 관계로 표현하고 있다.

구조 및 상세 설계, 구현 단계에서 개발자는 기존 연구의 UMAS 개발 방법론이 설명하고 있는 목표와 과업을 설계하고 구현하는 방법[4]을 따른다. UMAS의 구조 설계 단계에서 목표와 과업은 각각 Goal과 Task의 하위 클래스로 구현된다. 또한 Goal 클래스는 목적-방법 관계에 의해 그것을 달성할 수 있는 Task들로 구성하고, Task 클래스는 분해 관계를 기반으로 하위 Goal들로 구성되는 관계를 갖는다. UMAS의 상세 설계 단계에서 개발자는 다른 목표들로 구성된 과업을

수행하는 절차를 명시하고, 구현 단계에서는 이 설계 내용을 기반으로 구현 코드를 작성한다.

개발된 행위 모델들은 테스트 단계에서 개발자는 시뮬레이션을 통한 테스트를 수행한 후, 동일 행위 모델들을 PILS 장비에 탑재하여 테스트한다. 이 때, PILS 테스트는 시뮬레이션 환경의 제어/물리 모델과 연동하여 수행된다. 마지막으로 행위 모델들은 실 무인체계에 탑재하여 테스트된다.

6.4. 제어 모델의 개발

제어 모델은 물리 모델로 구현되는 무인체계의 동특성 및 통신장비, 센서 등의 탑재 장비를 제어하는 모델로 일반적으로 물리 모델과 쌍을 이루어 개발된다. 요구 분석 단계에서는 피처 모델로 식별되는 피처들 중에서 무인체계의 장치를 제어하는 논리(Logic)에 관련된 피처(그림 3의 c에서 비행제어기, 통신제어 모듈 등 참조)가 제어 모델로 개발되는 대상이 된다. 추가적으로 제어 논리의 입력과 물리 모델에 설정/전달되는 출력이 식별된다.

설계 단계에서 개발자는 MATLAB을 사용하거나 기존의 전통적인 OOP(Object-Oriented Programming) 개발 방법론에 따라 UML을 사용하여 제어 모델을 설계한다. 일반적으로 MATLAB은 지상 또는 공중 무인체계의 동역학 제어 모델을 설계하기 위해 사용되며, OOP 방법론은 그 밖의 탑재장비의 제어 모델을 설계/개발할 때 사용된다.

구현 단계에서, 개발자는 MATLAB 모델로부터 자동 생성되는 코드를 사용하거나 UML로 설계된 내용을 반영하여 코드를 구현한다.

시뮬레이션과 PILS 테스트 과정에서 개발된 제어 모델은 시뮬레이션 실행 서비스에서 실행된다. 실 무인체계 테스트에서 개발자는 제어 모델을 무인체계에 탑재된 연산 컴퓨터에 설치하여 실행한다.

6.5. 물리 모델의 개발

요구 분석 단계에서 물리 모델은 제어 모델과 유사하게 피처 모델로부터 식별된다. 단, 그 대상이 되는 것은 통신장비, 센서 등의 물리적 탑재장비이다. 또한, 제어 모델로부터의 입력이 식별되고, 해당 입력에 대한 결과와 물리 모델에 영향을 주는 환경 요소(바람 또는 노면의 특성)가 분석된다.

설계/구현 단계에서 개발자는 제어 모델과 동일한 방법으로 물리 모델을 개발한다.

마지막으로, 시뮬레이션과 PILS 테스트 단계에서 물리 모델은 시뮬레이션 실행 환경에서 실행된다. 실 무인체계 테스트 시, 물리 모델은 무인체계 및 탑재 장비들로 대체된다.

7. 기계학습에 활용 방안

본 연구가 제안하는 군집 무인체계의 개발 방법론에 따라 개발된 모델들과 함께, 제안 M&S 시스템은 크게 제어 모델과 행위 모델의 기계 학습을 위한 환경으로 활용될 수 있다. 특히, 본 연구가 제안한 모델링 인프라스트럭쳐와, 이를 기반으로 무인체계를 구성하는 컴포넌트들의 부분적인 개발을 지원하는 개발 방법론은 기계학습 기법을 적용하여 제어 모델과 다양한 수준의 행위 모델을 개발하기 위한 토대를 제공한다.

기계학습 기반의 제어 모델을 개발하는 과정에서 개발자는 어떤 입력(예, 이동 경로)이 제어 모델에 주어졌을 때, 높은 충실도의 공학급 물리 모델을 사용하여 수행되는 시뮬레이션을 통해 효율/효과적인 제어 모델을 학습시키기 위한 데이터를 생성할 수 있다. 또한, 강화 학습 기법을 적용하여 제어 모델을 개발할 경우, 강화 학습을 위한 환경으로 제어 모델을 제외한 타 모델들과 M&S 시스템을 사용할 수 있다.

이와 유사하게 기계학습 기반의 행위 모델을 개발하는 과정에서 개발자는 복합적인 요소로부터의 의사 결정을 학습하기 위해서 기 개발된 모델들과 M&S 시스템을 활용할 수 있다. 제안하는 시스템은 높은 충실도의 위치/자세 데이터로부터 게임 엔진을 통해 실사와 유사한(Photo-Realistic) 영상과 결합된 학습 데이터의 생성을 지원한다. 그리고, 타 모델들과 제안한 M&S 시스템은 특정 목표를 수행하는 행위 모델의 개발에서 효율/효과적으로 임무를 수행하는 의사결정을 위한 강화학습 환경으로의 활용이 가능하다.

8. 결론 및 향후 연구

본 연구는 군집 무인체계 개발을 위한 M&S 시스템의 구조와 기능을 제안했다. 제안한 M&S 시스템은 모델링 인프라스트럭처를 기반으로 개발되는 군집 무인체계 모델과 시뮬레이션 실행 서비스와의 상호작용을 통해 시뮬레이션을 진행한다. 또한, 본 연구는 제안한 모델링 인프라스트럭처 상에서 군집 무인체계의 모델을 개발하는 AOSE 기반의 개발 방법론을 제안하고, 임무별 그룹과 개체, 행위, 제어, 물리 모델을 개발할 때, 각 단계별 활동 및 산출물에 대해서 설명했다. 마지막으로 향후 개발될 기계학습 기반의 모델들이 제안한 방법론과 시스템을 토대로 제안 시스템 및 개발 방법론을 활용한 기계학습 방안에 대해 설명했다.

본 연구가 제안하고 있는 내용은 아직 초기 단계의 연구로 개발 방법론 측면에서 군집 무인체계의 개발에 실제로 적용한 구체적인 사례 연구가 필요하다. 또한, 제안한 시스템은 현재 모델링 인프라스트럭처를 기반으로 기본적인 시뮬레이션 기능을 개발한 상태이며, 영상 생성기를 비롯하여 시뮬레이션 통제/분석기, PILS장비, 실 무인체계 등을 추가 개발 중에 있다. 향후

지속적인 연구 개발을 통해 이와 같은 사항들을 보완할 계획이다. 추가로, 지속적인 연구를 통해 기계학습 기반의 행위 및 제어 모델을 개발하는 활동을 포함해서 제안한 군집 무인체계 개발 방법론을 확장할 계획이다.

마지막으로, 기계학습 기반의 행위 및 제어모델을 실 무인체계에 탑재하여 운용하려면 본 연구에서 다루고 있지 않은 신뢰성이 뒷받침되어야 한다. 이를 위해 기능 및 요구 성능의 테스트 및 검증 방법과 함께 안전성과 보안성, 무결성, 무인체계와 시뮬레이션 모델의 일치성 등을 보장하기 위한 연구가 요구된다.

감사의 글

본 연구는 국방과학연구소 국방첨단기술연구원(계약번호 UG190055RD)이 지원하는 연구비의 지원으로 수행되었습니다.

참고문헌

1. Nicholas R Jennings and Michael Wooldridge. 2001. Agent-Oriented Software Engineering. AGENT-ORIENTED Softw. Eng. Lect. Notes Comput. Sci. 2001, Vol. 1957/2001, 55-82 1957, (2001), 55-82. DOI:https://doi.org/10.1007/3-540-44564-1
2. Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. 2017. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. (May 2017), 621-635. DOI:https://doi.org/10.1007/978-3-319-67361-5_40
3. Lin Padgham and Michael Winikoff. 2003. Prometheus: A methodology for developing intelligent agents. Agent-oriented Software Engineering III 2585, (2003), 174-185. DOI:https://doi.org/10.1007/3-540-36540-0_14
4. Hee-Soo Kim and Seok-Won Lee. 2018. Dependability-Enhanced Unified Modeling and Simulation Methodology for Critical Infrastructures. Information and Software Technology. (June 2018). DOI:https://doi.org/10.1016/j.infsof.2018.06.002
5. IEEE Computer Society. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification, 2010.
6. Eric Yu. 1995. Modelling strategic relationships for process reengineering. University of Toronto.
7. K.C. Kang, Jeajoon Lee, and P. Donohoe. 2002. Feature-oriented product line engineering. IEEE Softw. 19, 4 (July 2002), 58-65. DOI:https://doi.org/10.1109/MS.2002.1020288

Seq-GAN 알고리즘을 활용한 자동 버그 정정 기법

양근석, 이철훈, 최현호, 이병정
서울시립대학교 컴퓨터과학과
{ypats87, cjfgnsdk12, gusgh3315, bjlee}@uos.ac.kr

A Novel Technique for Automatic Bug Repair by using Seq-GAN Algorithm

Geunseok Yang, Cheolhun Lee, Hyunho Choi and Byungjeong Lee
Department of Computer Science, University of Seoul

요 약

소프트웨어의 규모와 복잡성이 증가하면서, 크고/작은 소프트웨어 버그들의 존재는 불가피하게 되었다. 소프트웨어 결함을 정정하기 위해, 경우에 따라 개발자들은 디버깅에 많은 시간을 소요할 수 있다. 만약 소프트웨어 버그를 자동 정정을 한다면, 좋은 품질의 소프트웨어를 제공할 수 있다. 본 논문에서는 SeqGAN 알고리즘을 적용한 프로그램 결함 정정 기법을 제안한다. 본 기법에서는 Generative Model과 Discriminative Model을 통하여 두 모델의 오류를 줄이고, 비용을 절감하면서 좋은 품질의 프로그램 패치를 생성한다. 만약, 테스트 케이스가 있으면 테스트 케이스 기반 적합도 함수로 생성된 후보 패치가 적합한지 평가하고, 테스트 케이스가 없으면 컴파일 성공으로 적절한 패치가 되었다고 가정한다. 본 기법을 평가하기 위해, 오픈 소스 프로젝트를 사용하여 관련 연구와 성능을 비교하였으면 본 방법이 더 효율적임을 보였다.

1. 서 론

최근 소프트웨어는 다양한 산업에 적용되면서, 규모와 복잡성 또한 증가하였다. 이에 따라 크고/작은 소프트웨어 버그들의 존재가 불가피 하게 되었다. 이러한 소프트웨어 버그들을 수정하기 위해 개발자들은 디버깅에 많은 시간을 소요하고 있다[1].

일반적으로 소프트웨어 버그가 발견되었을 때, 개발자들은 프로그램의 어느 소스코드 파일에서 버그가 발생했는지 추적한다. 그리고 의심스러운 버기 라인을 찾고 해당 라인에서 버그 정정을 시도한다. 이 과정을 버그 정정 (Bug Repair)이라고 한다. 마지막으로 품질 보증자 (Quality Assurance)의 검증을 통해 소프트웨어 버그가 정확하게 정정되었는지 확인하고, 최종 소프트웨어 패치를 진행 (Release)한다.

자동 버그 정정 (Automatic Bug Repair)과 관련한 연구도 활발히 진행중에 있다. DeepFix [2]는 Attention Encoder-Decoder 기법을 이용하여 버그 정정 알고리즘을 제안했다. C언어 기반 프로그램의 버그를 정정 하기 위해 얼마나 정확하게 결함을 정정했는지 와, 컴파일러 오류 메시지 개수를 얼마나 줄였는지 에 대해 초점을 두었다. 하지만 아직까지 다양한 알고리즘을 활용하여 프로그램 결함 정정의 성능을 향상할 필요가 있다. GenProg [3]는 유전 프로그래밍 (Genetic Programming)을 활용하여 버그를 정정하였다. 자세히는 검색 공간을 확장하면서, 테스트 케이스 기반 적합도 함수를 통해 생성한 프로그램 후보 패치가 적합한지 아닌지 판단하였다. 하지만 적합한 후보 패치를 얻기 위해 많은 사이클이 필요하다.

이러한 문제를 해결하기 위해 본 논문에서는 프로그램 자동 정정 방법을 제안한다. 자세히는, Seq-GAN 알고리즘[4]을 이용하여, 프로그램 후보 패치를 생성한다. 품질 좋은 후보 패치를 생성하기 위해 Generative Model과 Discriminative Model을 활용하여 오류를 줄이고, 생성된 후보 패치가 적합한지를 판단한다. 이 과정에서 테스트 케이스가 존재한다면 테스트 케이스 기반 적합도를 계산하여, 모든 테스트 케이스가 통과된 후보 패치로 최종적으로 프로그램 버그를 정정한다. 만약 테스트 케이스가 존재하지 않다면, 컴파일 성공으로 적절한 패치가 되었다고 가정한다.

제안한 기법의 성능을 평가하기 위해, 프로그램 결함 정정의 관련 연구와 비교하였으며, 제안한 방법이 더 효율적임을 보였다.

본 연구의 기여도 (Contribution)는 다음과 같다.

- 버그 자동 정정에서 Seq-GAN 알고리즘을 이용하여 프로그램 결함을 정정하였다. 이 과정에서 Generative Model과 Discriminative Model을 활용하여 오류를 줄이고, 품질 좋은 후보 패치를 생성하였다.
- 제안한 모델의 효율성을 평가 하기 위해, 과거 관련연구들과 비교하였고, 제안한 방법이 더 잘 정정하는 것을 보였다.

본 논문은 다음과 같이 구성된다. 2장에서는 제안한 방법론을 제시하고, 사례 연구를 3장에서 보인다. 4장에서 토의를 진행하고. 5장에서 관련 연구를 보인다. 6장에서 제안한 방법의 결론과 향후 연구를 소개한다.

2. 프로그램 버그 정정 기법

전반적인 프로그램 버그 정정 도식도는 다음 그림 1과 같다.

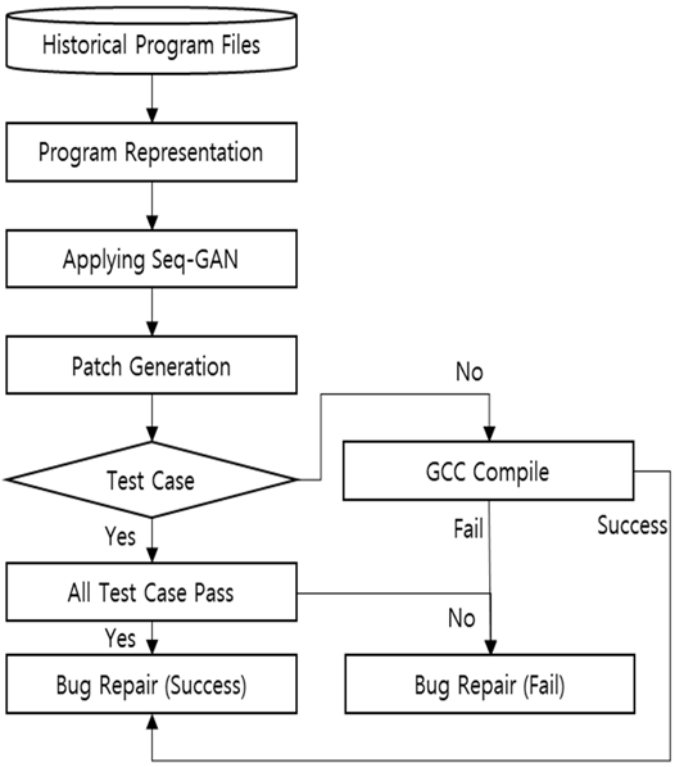


그림 1. 전반적인 도식도

자세히는, 주어진 프로그램 소스코드를 딥러닝에 적용할 수 있게 프로그램 재표현 (Program Representation) 과정을 진행한다. Seq-GAN 알고리즘을 통해 프로그램 후보 패치를 생성한다. 만약 테스트 케이스가 존재한다면, 테스트 케이스 기반 적합도 함수를 통해 생성된 패치가 적합한지 아닌지 판단한다. 테스트 케이스가 존재 하지 않는다면, GCC 컴파일러 통과 여부를 통해 패치가 성공적인지 아닌지 판단한다.

3. 사례 연구

3.1 테스트 케이스 기반 버그 정정

제안한 방법을 이해하기 위한 샘플 소스코드와 테스트 케이스를 다음 그림 2과 같이 보인다.

```
1.    #include <stdio.h>
2.
3.    int main()
4.    {
5.      int i = 0, j = 0;
6.
7.      printf("Input : ");
8.      scanf("%d", &i);
9.
10.     for (j = 1; j <= 10; j++)
11.     {
12.             printf("%d ", i * j);
13.     }
14.
15.     return 0;
16.   }
```

```
테스트 케이스 #1
IN 2
OUT 2 4 6 8 10 12 14 16 18

테스트 케이스 #2
IN 3
OUT 3 6 9 12 15 18 21 24 27
```

그림 2. 버그 소스코드와 테스트 케이스

위 버그 프로그램은 정수 형태로 표현된 '단'을 입력받아, 해당 구구단을 출력하는 프로그램이다. 자세히는 라인 8에서 사용자에게 정수 숫자를 입력 받아 'i 변수'에 저장하고, 라인 10의 반복문에서 입력받은 구구단을 출력한다. 하지만 위 버그 소스코드는 라인 10의 연산자 ("<=")로, 주어진 테스트 케이스 #1를 입력했을 때 "2 4 6 8 10 12 14 16 18 20 "가 출력이 되어 테스트 케이스를 통과할 수 없다.

해당 버기로 의심되는 라인 (라인 10, 라인 12)을 해결하기 위해 그림 3와 같이 세 가지 패치 후보들을 생성한다. 각 패치 후보에 주어진 테스트 케이스를 적용하면, 패치 후보 #1이 통과 되고, 패치 후보 #2와 #3은 통과되지가 않아 적합한 패치라고 간주하지 않는다. 따라서 최종적인 패치 후보 #1을 적용한다.

```
패치 후보 #1. j < 10 - 라인 10
패치 후보 #2. j > 10 - 라인 10
패치 후보 #3. i / j - 라인 12
```

```
패치 후보 #1 and 테스트 케이스 #1
결과: 2 4 6 8 10 12 14 16 18
평가: 통과

패치 후보 #1 and 테스트 케이스 #2
결과: 3 6 9 12 15 18 21 24 27
평가: 통과

패치 후보 #2 and 테스트 케이스 #1
결과: (blank)
평가: 실패

패치 후보 #2 and 테스트 케이스 #2
결과: (blank)
평가: 실패

패치 후보 #3 and 테스트 케이스 #1
결과: 2 1 0 0 0 0 0 0 0 0
평가: 실패

패치 후보 #3 and 테스트 케이스 #2
결과: 3 1 1 0 0 0 0 0 0 0
평가: 실패
```

그림 3. 생성된 패치 후보 및 적합도 평가 결과

3.2 구문 오류에 대한 버그 정정

구문 오류에 대한 버그 정정을 이해하기 위한 샘플 코드를 아래 그림 4과 같이 보인다. 이 프로그램은 정수 형태로 '시간'을 입력받아, 분으로 출력하는 프로그램이다. 자세히는 라인 7에서 사용자에게 정수 숫자를 입력 받아 'hour' 변수에 저장한다. 라인 9에서의 조건문에 따라 시간에 대한 분이 환산된다. 하지만 이 프로그램은 라인 13의 '}' 으로 의도치 않게 프로그램 구문 오류가 발생한다.

```
1. #include <stdio.h>
2.
3. int main() {
4.
5. int hour=0, min=0;
6. printf("Please Input :");
7. scanf("%d", &hour);
8.
9. if (hour == 1)
10. min = 60;
11. else
12. min = hour * 60;
13. }
14.
15. printf("%dmin ₩n", min);
16. return 0;
17.
18. }
```

그림 4. 구문오류 버그 소스코드

리눅스 GCC 컴파일러를 통해 컴파일을 진행하면 다음과 같은 메시지가 출력된다.

```
KCSE.c:14:8: error: expected declaration specifiers or '...' before string constant
 printf("%dmin ₩n", min);
        ^~~~~~~~~~
KCSE.c:14:20: error: unknown type name 'min'
 printf("%dmin ₩n", min);
                    ^~~
KCSE.c:15:1: error: expected identifier or '(' before 'return'
 return 0;
 ^~~~~~
KCSE.c:16:1: error: expected identifier or '(' before '}' token
 }
```

이 프로그램을 정정하기 위해서는 다양한 패치가 이루어져야 한다. 정정을 위한 생성된 패치는 다음과 같다.

```
패치 후보 #1. } 삭제 - 라인 13
패치 후보 #2. { 추가 - 라인 11
```

패치 후보 #1와 #2를 적용하면, 구문 오류 버그 프로그램을 정정할 수 있다.

4. 토 의

4.1 구조적 위협 요소

본 논문에서는 테스트 케이스 기반 적합도 함수를 모두 통과한 성공한 패치 개수만으로 평가를 진행하였다. 하지만, 사용한 평가만으로 제안한 프로그램 결함 정정의 성능이 향상 좋다고 일반화 할 수는 없다. 향후 다른 평가 척도를 이용하여, 다양하게 모델을 검증할 예정이다.

4.2 내부 위협 요소

제안한 모델에서 사용한 파라미터는 과거 연구에서 사용했던 파라미터로 구성되어 있다. 하지만, 사용된 파라미터가 항상 프로그램 결함 정정에서 좋은 성능을 보인다고 단정 지을 수 없다. 적절한 최적의 파라미터를 선택할 수 있도록 추가 연구할 예정이다.

4.3 외부 위협 요소

본 논문에서는 오픈 소스 프로젝트를 이용하여 연구를 진행하였다. 하지만, 비즈니스 프로젝트에 대해서는 추가 연구가 필요하다. 일반적으로 오픈 소스 프로젝트와 비즈니스 프로젝트의 데이터와 적용 프로세스가 다르기 때문이다. 그리고, 사용된 오픈 소스 프로젝트 이외의 다른 오픈 소스 프로젝트에 대해 적용가능 여부는 추가 연구가 필요하다.

5. 관련 연구

GenProg [3]은 유전 프로그래밍을 사용하는 테스트 케이스 기반 적합도 함수를 활용한 버그 정정 알고리즘이다. 자세히는 변이, 교차, 선택의 세 가지 유전 연산자를 사용한다. GenProg은 버그가 있는 소스 코드를 추상 구문 트리 (AST)로 변환한다. 유전 연산자를 통해 프로그램 후보 패치를 생성한다. 생성한 후보 패치의 적합성에 테스트 케이스를 적용하여 적합한지 아닌지 판단한다. 하지만 적합한 후보 패치를 얻기 위해 많은 사이클과 많은 시간이 소요된다.

PAR [5] 버그 정정 알고리즘은 복구 템플릿 기반으로 버그를 정정한다. 템플릿에는 버그를 수정하는 일반적인 방법이 포함되어 있다 (예: null 확인 추가). 무작위 검색에서 템플릿은 버그가 있는 프로그램에 적용되고 테스트가 되는 한계를 가진다.

6. 결 론

본 논문에서는 딥러닝 알고리즘을 활용하여 프로그램 결함 정정 방법을 제안했다. 자세히는 Seq-GAN 알고리즘에서 Generative Model과 Discriminative Model 기반으로 생성한 프로그램 후보 패치의 오차를 줄이고, 프로그램 후보 패치를 생성한다. 주어진 테스트 케이스를 적용하여 생성된 프로그램 후보 패치의 적합성을 판단한다. 모든 테스트 케이스를 통과한 경우, 프로그램 결함 정정이 되었다고 판단한다. 제안한 모델의 성능을 평가 하기 위해, 관련 관련 연구와 성능 비교를 진행하였으며, 제안한 모델이 더 효과적임을 보였다. 본 논문에서는 다양한 프로그래밍 언어에 적용해야하는 한계를 갖고 있고, 향후 다양한 알고리즘을 활용하여 모델과 성능을 확장할 예정이다.

Acknowledgement

이 논문은 2017년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2017R1A2B4009937).

참고 문헌

[1] T. Zhang, J. Chen, G. Yang, B. Lee, and X. Luo, "Towards More Accurate Severity Prediction and Fixer Recommendation of Software Bugs", In Journal of Systems and Software, vol. 117, pp. 166-184, 2016.

[2] R. Gupta, S. Pal, A. Kanade, and S. Shevade, "Deepfix: Fixing Common C Language Errors by Deep Learning", In Proc. of Thirty-First AAAI Conference on Artificial Intelligence, pp. 1345-1351, 2017.

[3] C. Le Goues, M. Dewey-Vogt, S. Forrest, and W. Weimer, "A Systematic Study of Automated Program Repair: Fixing 55 out of 105 Bugs for $8 each", In Proc. of 34th International Conference on Software Engineering, pp. 3-13, 2012.

[4] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence Generative Adversarial Nets with Policy Gradient", In Proc. of Thirty-First AAAI Conference on Artificial Intelligence, pp. 2852-2858, 2017.

[5] D. Kim, J. Nam, J. Song, and S. Kim, "Automatic Patch Generation Learned from Human-written Patches", In Proc. of the 2013 International Conference on Software Engineering, pp. 802-811, 2013.

캡슐내시경 영상 데이터의 학습 성능 강화를 위한 색 공간 군집 기반 특성 편향도 메트릭

임창남*, 박예슬*, 이광재**, 이정원*

*아주대학교 전자공학과, **아주대학교 의과대학 소화기내과

Chn0714@naver.com, yeseuly777@gmail.com, kjl@ajou.ac.kr, jungwony@ajou.ac.kr

Skewness Metric of Color Space Cluster-based Feature Selection for Improving Learning Performance using Capsule Endoscopic Image Data

Chang-nam Lim*, Ye-Seul Park*, Kwang-Jae Lee**, Jung-Won Lee*

*Department of Electrical and Computer Engineering, Ajou Universitiy

**Department of Gastroenterology, Ajou University Hospital

요 약

최근에 딥러닝 기법에 대한 연구와 적용이 여러 분야에서 활발하게 이루어지고 있다. 이러한 딥러닝 기법은 데이터에 의해 모델의 성능이 크게 좌우되는데, 분야에 따라 추가적인 데이터를 확보하기 어려운 경우도 존재한다. 이로 인해 추가적인 데이터의 확보 없이도 딥러닝 모델의 성능을 향상시키기 위한 여러가지 연구가 진행되고 있다. 그러나 이와 같은 방법들은 학습 성능(정확도, 정밀도 등)을 개선할 수 있지만, 성능을 평가하기 위해 사용된 데이터에 대한 일반성이나 편향성 등을 검증하기 어렵다. 따라서 본 논문에서는 학습 모델의 성능 향상을 위해 훈련 데이터 셋이 전체 데이터의 특성을 얼마나 포함하고 있는지 측정할 수 있는 특성 편향도 메트릭을 제안한다. 특성 편향도 메트릭은 학습에 사용되는 영상의 특성을 단위 특성으로 추출하여, 개별적인 단위 특성에 속해 있는 데이터 셋을 조합하고 학습 데이터 셋을 구성할 수 있게 한다. 모델의 높은 성능을 위해서는 훈련 데이터 셋이 실제 사용될 전체 데이터 셋의 모든 특성을 포함하고 있어야 하며, 제안하는 메트릭으로 이를 측정할 수 있을 것으로 기대된다. 제안하는 특성 편향도를 검증하기 위해 캡슐내시경 데이터에 대해 영상의 위, 소장, 대장의 위장관을 분류하기 위한 학습 모델에 대해 실험을 진행하였다. 캡슐내시경 영상의 경우 개인마다 영상의 색이 다를 수 있기 때문에 RGB 색 공간에서 특징을 추출해 군집을 형성하였으며, 이 군집을 단위 특성으로 정의하였다. 128개의 캡슐내시경 데이터에서 총 3,760,701장의 영상을 이용해 실험을 진행하였고, 훈련 데이터와 테스트 데이터의 개수를 71:29로 고정할 때 특성 편향도에 따른 모델 성능 변화를 실험하였다. 실험 결과 특성 편향도가 1로 좋을 때의 성능은 83%의 정확도를 보였고, 특성 편향도가 0.889로 떨어질 때는 66%의 정확도를 보인 것을 확인하였다. 이처럼 모델의 성능에 대해 더 세부적인 평가 분석을 할 수 있을 것으로 기대된다.

1. 서 론[1]

최근 인공지능 분야에서 딥러닝 기법에 대한 연구가 활발히 진행되고 있으며, 그 중에서도 영상 학습에 있어 전통적인 방식의 학습 모델에 비해 높은 성능을 확보할 수 있게 하여 더욱 활성화되고 있다. 의료 분야에 있어서도 마찬가지다. 근래 대부분의 질병의 진단을 영상을 기반으로 수행하기 때문에, 영상 진단을 보조하기 위한 많은 연구들이 수행 중이다. 예를 들어, 폐 결절을 검출하기 위한 CT 영상을 이용한 연구[1-2], 피부 병변을 발견하기 위한 피부 촬영 영상을 이용한 연구[3], 뇌종양을 진단하기 위한 T1 MRI, fMRI, DTI 영상들을 이용한 연구[4]와 유방종양의 진단을 보조하기 위한 전단파 탄성 검사 영상을 이용한 연구[5] 등

[1] "본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음" (IITP-2019-2016-0-00309)

영상을 이용한 진단 보조 연구들이 있다. 본 연구에서는 그 중에서 캡슐내시경 영상을 이용한 연구[6-8]에 주목하였다.

캡슐내시경은 위장관 전체를 관찰할 수 있고, 일반 내시경 검사나 대장 내시경 검사로 관찰을 할 수 없는 소장 기관을 관찰할 수 있어 활용성이 높아지고 있다. 캡슐내시경 영상 분석 연구는 크게 두 가지 방향으로 진행되고 있다. 캡슐의 위치를 추적하기 위해, 영상에 나타난 위장관이 식도, 위, 소장, 대장 중 어느 부분인지 분류하는 연구[6]나, 폴립 인식[7], 출혈 감지[8] 등 병변을 검출하는 연구들이 수행되고 있으며, 최근에 수행되는 연구는 대부분 딥러닝 기법들을 적용하고 있다.

딥러닝 기법은 학습에 사용되는 데이터에 의해 모델의 성능이 크게 좌우된다. 하지만 의료 분야의 특성상 많은 환자의 데이터를 확보하기 어렵고, 의료 전문가가 직접 레이블링 작업을 수행해야 하므로 신뢰성이 높은 의료 데이터의 확보가 어렵다. 따라서 기존 모델의 성능을 강화하기 위해 학습에 사용될 데이터를 추가적으로 확보하기 어렵다.

이러한 문제점 때문에 모델의 성능을 향상시킬 수 있는 새로운 방법이 필요하며, 이를 위하여 많은 연구들이 진행되고 있다. 활성화 함수를 통한 성능 개선 방법[9-10], 데이터 품질 개선을 통한 방법[11], 인셉션(Inception) 모델과 같은 새로운 형태의 모델을 생성하여 성능을 개선시키는 방법[12-13] 등이 있다. 그러나 이와 같은 방법들은 학습 성능(정확도, 정밀도 등)을 개선할 수 있지만, 성능을 평가하기 위해 사용된 데이터에 대한 일반성 및 편향성 등을 검증하기 어렵다.

모델의 성능과 일반성을 위해서는 전체 데이터의 특성을 모두 학습하여야 한다. 이를 위해서는 데이터의 특성을 파악하고 특정 특성으로만 구성되어 편향된 데이터 셋이 아닌 모든 특성을 고르게 반영한 훈련 데이터 셋을 생성해 모델을 학습시켜야 한다. 따라서 본 논문에서는 학습 모델의 성능 향상을 위해 훈련 데이터 셋이 전체 데이터의 특성을 얼마나 포함하고 있는지 측정할 수 있는 특성 편향도 메트릭을 제안한다. 특성 편향도 메트릭은 학습에 사용되는 영상의 특성을 단위 특성으로 추출하여, 개별적인 단위 특성에 속해 있는 데이터 셋의 조합을 통해 학습 데이터 셋을 구성할 수 있게 한다. 모델의 높은 성능을 위해서는 훈련 데이터 셋이 실제 사용될 데이터 셋의 모든 특성을 포함하고 있어야 하며, 제안하는 메트릭으로 이를 측정할 수 있을 것으로 기대된다.

그림 1은 색 공간에서 데이터의 특성 분포를 나타낸 것으로, 각각의 색깔은 하나의 레이블을 의미한다. 그림 1에서와 같이 하나의 레이블에서도 몇 개의 군집으로 나눠질 수 있으며, 각 군집은 군집 내부의 데이터끼리 색 공간 내에서 유사한 특징을 갖는다. 본 연구에서는 이와 같은 단위 군집을 영상의 특징으로 추출하였으며, 이를 단위 특성으로 정의하였다. 각 단위 특성의 적절한 조합을 통해 훈련 데이터 셋을 생성할 수 있다.

그림 2는 훈련 데이터 셋을 생성하는 방법을 그림으로 표현하였다. 전체 데이터에 Class A와 Class B라는 레이블이 존재하고, Class A내에 여러 특성이 존재한다. 그림 2(a)는 통상적인 방법처럼 임의로 훈련 데이터 셋을 생성했을 때, 훈련 셋과 테스트 셋에 Class A의 특성 분포를 나타낸다. 임의로 훈련 데이터 셋을 생성할 경우 그림 2(a)와 같이 훈련 셋이 전체 데이터의 모든 특성을 포함하지 못할 수 있다. 그림 2(b)는 제안하는 메트릭을 기반으로 하여 전체 데이터의 모든 특성을 포함하도록 훈련 데이터 셋을 생성한 것을 나타낸다. 전체 데이터의 특성을 모두 포함한다는 것은 모델의 학습이 편향되지 않도록 한다.

제안하는 메트릭을 검증하기 위하여 A병원과의 협력을 통해 330개의 캡슐내시경 검사 데이터를 확보하였으며, 이를 통해 캡슐내시경 영상의 위장관을 분류하기 위한 학습 모델을 설계하였다. 수집된 330개의 데이터 중에 위, 소장, 대장이 모두 레이블링 된 데이터는 128개로 이 데이터를 이용하여 실험을 진행하였다. 수집한 데이터에 대하여 특성 편향도를 1로 였을 때의 정확도는 83%, 0.944로 하였을 때는 70%, 0.889로 하였을 때는 66%의 정확도를 나타냈다. 이와 같은 결과는 훈련 데이터가 전체 데이터의 특성을

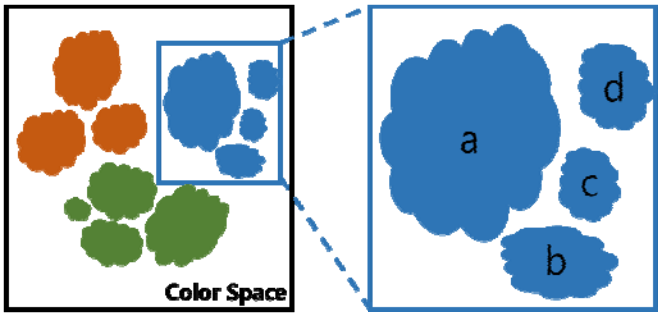


그림 1. 색 공간에서 데이터의 특성 분포

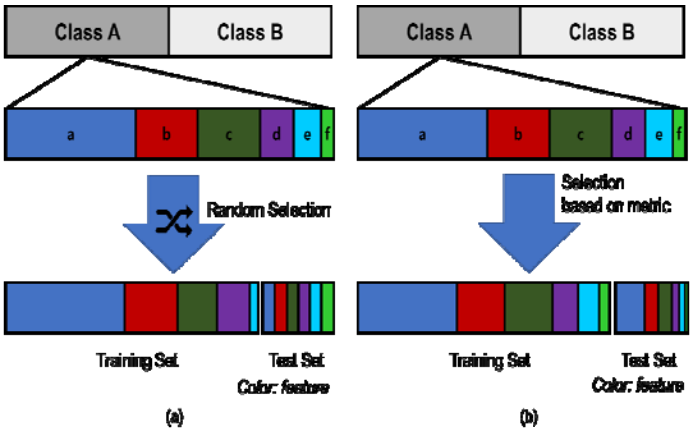


그림 2. 훈련 데이터 셋 생성 방식
(a)임의 생성, (b)특성 편향도 메트릭 기반 생성

반영하는 것이 학습의 성능에 미치는 영향이 크다는 것을 확인할 수 있으며, 특성 편향도 메트릭에 대한 중요성을 보여준다. 또한 그림 2(a)에서 표시한 훈련 데이터 셋을 이용해 Class A, Class B를 분류하는 모델을 학습할 경우 Class A의 f특성을 학습하지 못했음에도 불구하고 정밀도, 재현율, F1 점수 같은 기존의 평가 지표에 대해 높은 평가를 얻을 수도 있다. 하지만 제안하는 메트릭을 이용하면 이처럼 적은 데이터만 존재하는 특성 또한 고려할 수 있기 때문에 모델에 대한 더 세분화된 평가를 할 수 있을 것으로 기대된다. 향후에는 특성 편향도 메트릭의 변화에 따른 학습 성능의 변화와 특성 편향도를 1로 했을 때 학습 데이터와 테스트 데이터의 비율에 따른 학습 시간과 성능을 분석할 예정이다.

2. 관련 연구

2장에서는 본 연구에 대한 관련 연구를 소개하고자 한다. 2.1절에서는 딥러닝 기법의 성능 향상 연구, 이어 2.2절에서는 캡슐내시경 영상의 학습과 관련된 연구 동향에 대해 소개한다.

2.1. 영상 학습 모델의 성능 강화 기법

2.1절에서는 딥러닝 기법의 성능을 향상시키기 위한 방법들을 소개하고자 한다. 딥러닝 기법이 여러 분야에서 많이 사용됨에 따라 딥러닝 모델의 성능을 향상시키기 위한 여러 연구가 진행되고 있다. [9-10]에서는 딥러닝 모델의 성능을 향상시키기 위해 새로운 활성화 함수를 제안하며, [9]에서 제안된 ReLU(Rectified Linear Unit) 함수의 경우 많은 딥러닝 모델에서 사용되고 있다. 활성화 함수를 통한 연구의 경우 기존의 여러 모델에서 활성화 함수만 변경하여 적용할 수 있다는 장점이 존재한다. [11]의 경우 데이터 품질을 평가하기 위한 메트릭을 제안하며, 고품질의 데이터를 이용하여 학습할 경우 학습된 모델의 성능이 향상될 수 있음을 제시하였다. [12-13]의 경우 딥러닝 모델에 적용할 수 있는 인셉션 모듈을 제안한다. 제안하는 모듈을 여러 번 반복해서 모델을 구축할 수 있으며, 기존의 모델들보다 성능이 향상됨을 보였다. [14]의 연구의 경우는 테스트 데이터 셋의 품질을 평가하기 위하여 품질 메트릭을 제안한다. 하지만 이와 같은 방법들은 학습 성능을 개선하는 것에 초점을 맞추고 있어, 평가된 성능에 대한 일반성을 검증하기 어렵다. 즉, 훈련 데이터 셋이 전체 데이터의 특성을 얼마나 반영하고 있는지 확인할 수 없다. 따라서 본 논문에서는 이를 평가할 수 있는 특성 편향도 메트릭을 제안한다. 모델의 높은 성능을 위해서는 훈련 데이터 셋이 실제 사용될 데이터 셋의 특성을 모두 포함하고 있어야 하며, 제안하는 메트릭으로 이를 측정할 수 있을 것으로 기대된다.

2.2. 캡슐내시경 영상 학습 모델 기법

2.2절에서는 캡슐내시경 영상 데이터를 활용하여 학습을 수행한 연구들에 대해 소개한다. [6]의 경우 캡슐내시경 영상에 대해 위, 소장, 대장으로 구분하는 연구를 진행하였는데, 25개의 적은 캡슐 내시경 데이터로 실험을 진행하였다는 한계점이 존재한다. 또한, 실험 과정에서 전체 데이터에서 랜덤하게 데이터를 뽑아 훈련 데이터와 테스트 데이터를 뽑았는데, 캡슐내시경 영상의 경우 일정 구간 내의 영상은 거의 유사하기 때문에 훈련 데이터와 테스트 데이터에 거의 유사한 데이터가 존재해 정확한 검증이 이루어지지 않았을 가능성이 존재한다. [7]의 경우 위장관 점액에서 튀어나온 형태인 융기성 병변인 폴립을 인식하는 연구를 진행하였으며, 35개의 캡슐 내시경 데이터에서 3,000개의 정상 영상과 1,000개의 폴립 영상을 이용하여 CNN 모델을 학습하였다. [8]의 경우 위장관의 출혈을 검출하기 위해 CNN을 이용하였으며, 7,150장의 정상 영상과 2,850장의 출혈 영상으로 학습을 진행하였다. [15]의 경우 십이지장충을 검사하기 위하여 11개의 데이터에서 약 44만 장의 영상을 이용하여 CNN모델을 학습시켰다. [16]의 연구는 12만장의 캡슐내시경 영상으로 거품, 이물질, 주름 등의 형태를 분류하는 CNN 모델을 학습시켰다. 이와 같은 연구 모두 훈련과 테스트에 사용할 데이터를 비율만을 설정한 후 임의로 선택하여 진행하고 있어, 학습 성능의 향상에 한계가 존재한다. 따라서 적절하게 훈련 데이터 셋을 구성할 수 있는 방법이 필요하다.

3. 색 공간 군집 기반 특성 편향도 메트릭 설계

3장에서는 캡슐내시경 영상 데이터의 훈련 데이터의 품질을 평가하기 위한 메트릭을 제안한다. 3.1절에서는 캡슐내시경 영상의 특성 편향도를 정의하고, 3.2절에서는 색 공간에 기반하여 캡슐내시경 영상의 특성을 분석한다. 3.3절에서는 K평균 군집화를 이용하여 특성 편향도를 생성하는 기법에 대해 설명한다.

3.1. 캡슐내시경 영상의 특성 편향도 정의

특성 편향도 메트릭은 전체 데이터의 특성을 훈련 데이터 셋이 얼마나 포함하고 있는지를 나타낸다. 따라서 다음과 같은 수식으로 정의된다.

$$Feature\ Skewness(label_i) = \frac{\#\ Features\ in\ Learning(label_i)}{\#\ Total\ Features(label_i)}$$

하나의 레이블의 데이터 내에서도 데이터의 특성이 다양하게 존재할 수 있기 때문에, 하나의 레이블 내에서 발견될 수 있는 모든 특성의 개수 대비 학습에 활용된 특성의 개수가 특성 편향도로 측정될 수 있다. 이와 같은 특성 편향도를 고려하여 훈련 데이터 셋을 생성할 때는 표 1의 요소를 고려하여야 하며, 특성 편향도 값은

표 1. 특성 편향도 기반 훈련 데이터 셋 결정 요인

결정 요인	결정 요인 정의
데이터 고유성	전체 데이터의 특성과 차이가 많이 나는 데이터(예: 돌연변이 데이터, 노이즈 데이터 등)를 훈련 데이터 셋에 포함할 것인가?
데이터 균등성	훈련 데이터 셋을 생성할 때 모든 데이터의 특성에 대해 균등하게 생성하였는가(편향되지 않도록 데이터 셋을 생성하였는가)?

1이 되도록 한다.

데이터 고유성은 훈련 데이터 셋을 생성할 때, 다른 데이터와는 구별되는 특성을 가진 데이터를 포함시킬 것인지에 대한 고려사항이다. 훈련 데이터 셋에 포함시키지 않으면 해당 특성을 학습할 수 없기에 학습 성능이 저하될 수 있고, 노이즈 데이터나 돌연변이 데이터의 특성을 포함시킬 경우 또한 아웃라이어(Outlier)가 되어 학습 성능이 저하될 수 있다. 데이터 균등성은 생성한 훈련 데이터 셋이 각각의 특성들을 편향되지 않고 균등한가에 대한 요인이며, 각 특성을 균등하게 학습하여야 모델의 성능이 향상될 수 있다.

3.2. 캡슐내시경 영상 데이터 셋의 색 공간 분포 분석

본 논문에서는 캡슐내시경 영상 데이터 셋을 기반으로 특성 편향도를 정의하기 위해, 8명 환자의 샘플 데이터에 대해 분석하였다. 그림 3은 4개의 서로 다른 데이터에서 대장 부분의 영상이다. 그림 3의 (a)와 (b)는 검은 이물질이 많고, (c)와 (d)는 선홍색 영상으로 비슷한 양상을 보이는 것을 확인할 수 있다.

그림 3에서 확인할 수 있듯이, 같은 레이블(대장)을 갖는 데이터 내에서도 위장관 내에 포함되는 이물질의 정도, 점막의 주름 정도에 따라 매우 다른 색 분포를 보이기 때문에, 영상의 특성을 정의하기 위해 RGB 색 공간에서 데이터 분포를 분석하였다. 각 영상의 색 공간 특징을 추출하기 위하여 그림 4와 같은 절차로 색 공간 특징을 추출하였다. 각 영상의 특성을 추출하기 위해 영상마다 RGB 히스토그램을 계산한 후, 각각의 채널마다 평균과 분산을 계산하여 6개의 값을 추출하고 이를 영상의 색 공간 특성으로 정의하였다.

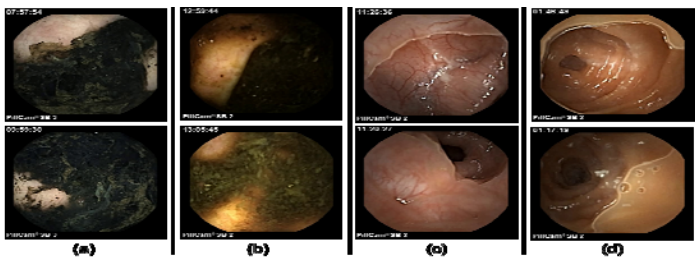


그림 3. 캡슐 내시경 별 대장 영상(사례 분석)

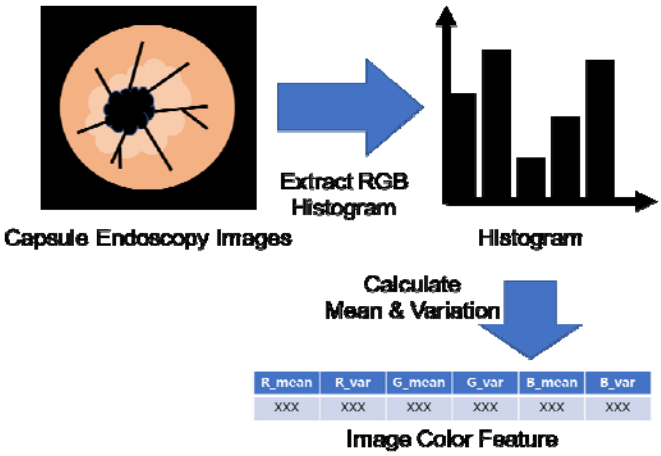


그림 4. 캡슐내시경 영상 색 공간 특징 추출 절차

그림 5는 8명의 환자 데이터에 대해 대장 영상들의 색 공간 특성을 시각화한 그림이다. 환자 8명의 데이터는 총 183,773장의 프레임으로 구성되어 있으며, 이 중 레이블 된 데이터는 183,456장으로 위 21,388장, 소장 104,881장, 대장 57187 장으로 구성되어 있다. RGB의 평균값을 표시한 그래프로, 각 색깔은 서로 다른 환자 데이터를 의미한다. 그림 3의 (a), (b), (c), (d)는 각각 그림 5에서 회색, 파란색, 빨간색, 보라색 데이터의 영상이다. 그림 5의 회색 데이터와 파란색 데이터가 다른 데이터에 비해 돌출되어 있는 것을 확인할 수 있는데, 이는 검은 이물질이 많기 때문에 돌출된 것으로 분석된다. 즉, 색 공간에서 비슷한 특성을 가진 영상끼리 군집이 형성된다. 따라서 본 논문에서는 이와 같이 추출된 군집을 캡슐내시경 영상 데이터의 특성으로 정의하였다.

3.3. K-평균 군집화를 이용한 특성 정보 생성 기법

캡슐내시경 영상 데이터의 특성 편향도 생성 기법은 그림 6과 같이 진행된다. 먼저, 캡슐내시경 데이터의 특성을 추출하기 위해 3.2절에서 정의한 영상의 색

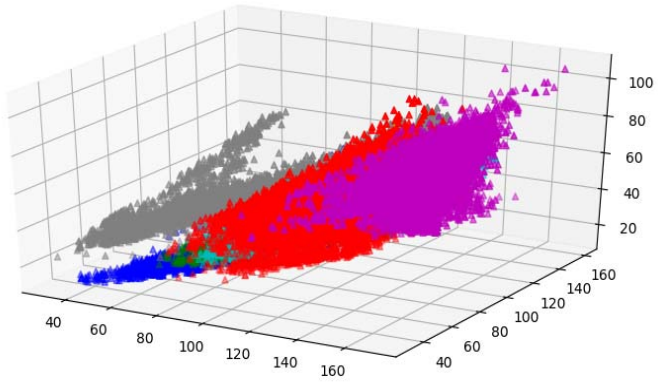


그림 5. 환자 별 RGB 색 공간 분포

공간 특성에 대해 K 평균 군집화를 이용해 군집을 생성한다. 생성한 군집에 대해 3.1절의 결정 요소를 고려하여 훈련 데이터 셋을 생성한 후, 훈련 데이터 셋의 특성 정보를 추출한다. 특성 정보는 생성한 훈련 데이터 셋에서 특성의 편향도를 평가하는데 도움이 될 수 있는 요소를 정의한 것으로 군집 인덱스, 군집 중심, 군집내 데이터 개수이며, 구체적인 내용은 표 2와 같다. 이러한 특성 정보를 통해 훈련 데이터 셋의 각 특성 별 데이터 개수, 각 특성 간의 거리 등을 알 수 있기 때문에 훈련 데이터 셋의 일반성을 평가하는데 도움이 될 수 있다.

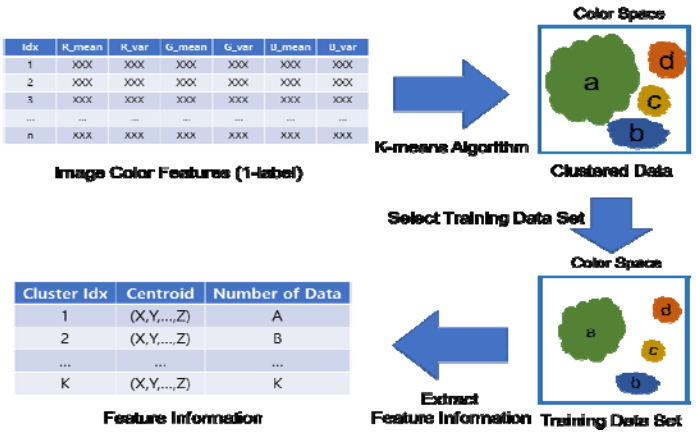


그림 6. K-평균 군집화 기반 특성 정보 생성 기법

표 2. 군집 별 특성 정보 정의

특성 정보 인덱스	특성 정보 번호
군집 인덱스	군집 번호로 1~k까지 부여됨.
군집 중심 좌표	군집의 중심점으로 RGB의 평균과 분산으로 이루어진 6차원 좌표.
군집 내 데이터 개수	군집에 포함된 데이터의 개수

K-평균 군집화에서 K를 결정하는 방법으로는 각 군집의 중심점에서 군집의 데이터 간의 거리를 합산한 총 거리의 합을 이용하였으며, 이 값이 작을수록 응집도가 높게 군집화가 이루어졌다고 평가할 수 있다. 그림 7은 8명의 데이터에 대해 군집 중심점과 데이터 간의 거리 총합을 표시한 그래프로 K가 1~3 일 때, 급격하게 감소하다가, K가 4~6 이후로 완만하게 감소하며, K가 7 이후인 구간부터는 군집의 개수가 많아져 확연이 감소한 것을 확인할 수 있다. 따라서 K는 4~6정도로 설정하면 적절하다고 판단할 수 있다.

그림 8은 그림 5에서 나타낸 데이터에 대해 K를 5로 하여 군집화를 한 그림으로, 각각의 색깔은 서로 다른 군집을 의미한다. 그림 5와 비교하였을 때, 파란색 데이터와 회색 데이터의 돌출된 부분이 같은 군집으로 나타난 것을 확인할 수 있는데, 이는 그림 3에서

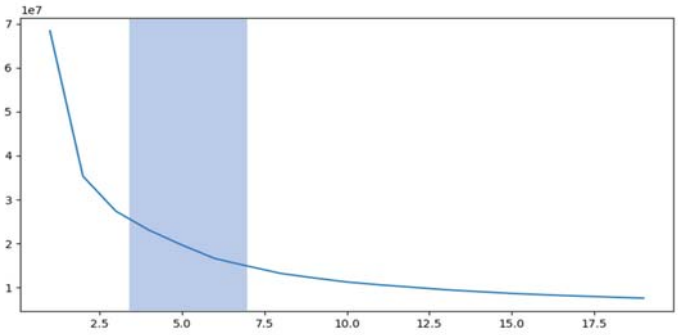


그림 7. 군집 중심점과 데이터 간 거리 총합

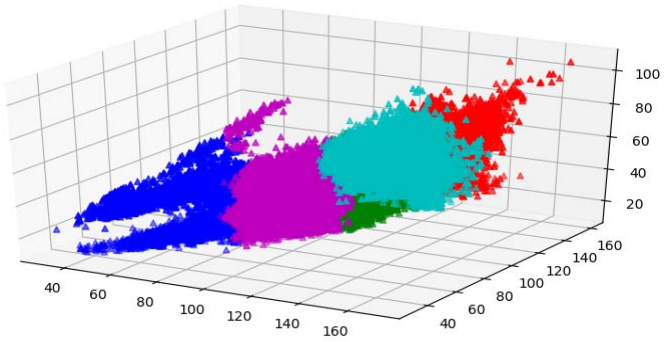


그림 8. 5-평균 군집 기반 RGB 색 공간 분포

확인하였듯이 검은색 이물질이 많은 영상이었기 때문에, 군집화 결과에서도 유사하게 평가된 것을 확인할 수 있다.

생성된 군집을 기반으로 편향도 결정 요인을 고려하여 훈련 데이터 셋을 선별한다. 그림 8에 표현한 8개의 데이터 중 파란색, 빨간색, 하늘색, 보라색, 회색의 5개의 군집을 훈련 데이터 셋으로 선택할 경우 특성 정보는 표 3과 같으며, 모든 군집 데이터를 포함했으므로, 특성 편향도는 1이 된다.

표 3. 훈련 데이터 셋의 5-평균 군집화 기반 특성 정보

Cluster_idx	Centroid	#Data
1(Blue)	(70.3,57.8,37.5,53.8,42.9,33.2)	3793
2(Green)	(118.4,97.8,45.2,36.9,30.3,20.2)	10448
3(Red)	(136.5,117.5,59.4,32.3,29.8,18.7)	10430
4(Cyan)	(137.4,89.2,64.4,36.9,27.5,23.4)	9519
5(Magenta)	(101.2,78.1,40.8,45.6,36.1,26.5)	10590
* Centroid: (R_mean,G_mean,B_mean,R_var,G_var,B_var)		

4. 특성 편향도 기반 캡슐내시경 학습 모델의 성능 비교

4장에서는 3장에서 정의된 특성 편향도를 기반으로 같은 데이터 셋을 다르게 분할하여 다른 편향도를 갖는 훈련 데이터 셋을 생성한 후, 각각의 학습 성능을 비교해본다. 이를 위하여 캡슐내시경 영상의 위장관 분류 모델로 실험을 진행하였다.

4.1. 캡슐내시경 영상 데이터 셋 수집 결과

캡슐내시경 데이터는 총 330개의 데이터를 수집하였으며, 이 중에 위, 소장, 대장이 모두 레이블링 된 128개의 데이터를 사용하였다. 캡슐내시경 검사를 실시한 환자의 수는 124명으로, 4명의 환자는 2번 검사를 받아 총 128개의 캡슐내시경 데이터가 존재한다. 환자의 성비는 남성 환자 86명, 여성 환자 38명으로 구성되어 있다. 128개 데이터의 총 영상 프레임 수는 3,793,718장이며, 이 중에서 위, 소장, 대장으로 레이블링 된 영상의 개수는 각각 313,403장, 2,543, 593장, 903,705장으로, 총 3,760,701장으로, 의사로부터 판독된 정보를 반영한 품질 좋은 데이터를 사용하였다.

4.2. 특성 편향도 평가 및 데이터 셋 구성

본 연구에서는 수집 및 정제된 128명의 데이터를 약 71:29으로 나누어 87개의 데이터를 훈련에, 41개의 데이터를 테스트에 사용하였다. 먼저, 특성 편향도를 생성하기 위해 128개의 데이터에 대해 색공간 분석을 진행하였다. 이 때, 군집 중심점과 데이터 간의 거리 총합에 대해 비교 분석을 진행하였으며, 이 때의 데이터 간의 거리 총합은 그림 9와 같다. 그림 9의 (a), (b), (c)는 학습 레이블인 위, 소장, 대장의 데이터를 나타내며, K가 5~8일 때 감소 폭이 완만한 것을 알 수 있으므로 K를 6으로 하여 군집화를 진행하였다. 전체 데이터 셋의 RGB 색 공간 분포는 그림 10, 군집화한 데이터의 RGB 색 공간 분포는 그림 11과 같으며, (a), (b), (c)는 학습 레이블인 위, 소장, 대장의 데이터를 나타낸다.

특성 편향도 평가 값에 따른 학습 성능을 비교하기 위하여 위, 소장, 대장 각각 6개의 군집을 합해 총 18개의 군집의 포함 비율을 다르게 하여 3개의 훈련 데이터 셋을 생성하였다. 각 데이터 셋은 모든 군집을 포함한 훈련 데이터 셋(FS=1), 1개의 군집을 제외한 훈련 데이터 셋(FS=0.944), 2개의 군집을 제외한 훈련 데이터 셋(FS=0.889)으로 구성된다. 3개의 훈련 데이터 셋 중 특성 편향도가 1인 훈련 셋의 특성 정보는 표 4와 같다.

또한, 군집별로 훈련 데이터 셋과 테스트 데이터 셋의 비율을 계산할 수 있는데, 전체 훈련 데이터의 개수와 테스트 개수의 비율인 71:29와 가까울 경우 3.1절에서 정의한 데이터 균등성이 높다고 판단할 수 있다. 계산한 군집 별 훈련 셋과 테스트 셋의 비율은 표 5와 같다.

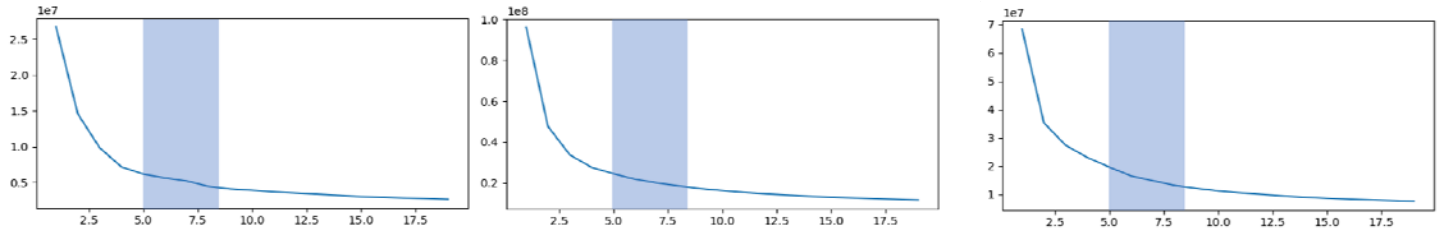


그림 9. 수집된 128명 환자 데이터의 군집 중심점과 데이터 간 거리 총합

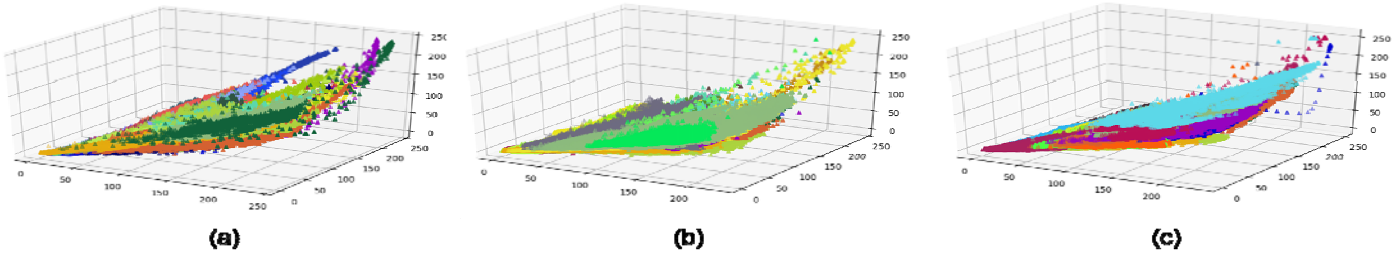


그림 10. 수집된 128명 환자 별 RGB 색 공간 분포

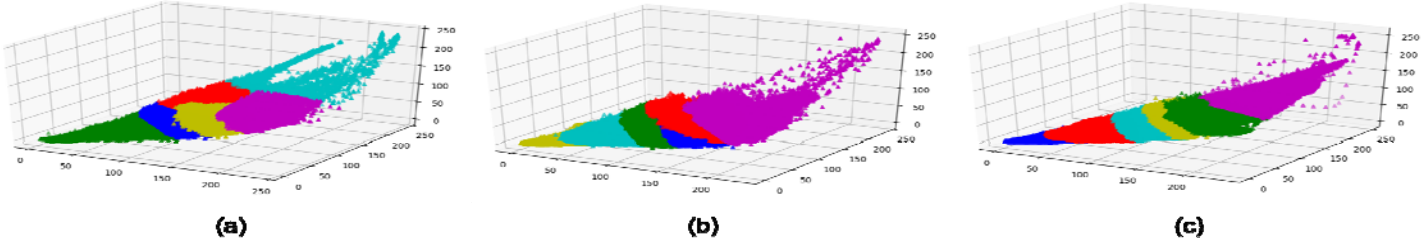


그림 11. 수집된 128명 환자 데이터의 6-평균 군집 기반 RGB 색 공간 분포

표 4. 훈련 데이터 셋(FS=1)의 6-평균 군집화 기반 특성 정보

Label	Cluster_idx	Centroid	#Data
Stomach	1(Blue)	(51.7,32.4,71.0,38.6,109.6,52.9)	59178
	2(Green)	(31.9,28.3,46.7,35.1,75.6,50.9)	23612
	3(Red)	(87.6,41.2,100.6,42.5,126.9,43.8)	40106
	4(Cyan)	(141.4,51.7,159.5,52.9,179.1,50.8)	3114
	5(Magenta)	(75.0,31.7,105.3,38.9,167.3,57.4)	44923
	6(Yellow)	(60.8,23.8,84.3,28.6,133.8,40.2)	77642
Small Intestine	1(Blue)	(33.6,21.9,81.5,35.1,125.1,48.1)	563606
	2(Green)	(27.3,22.4,68.1,38.8,99.3,53.6)	453261
	3(Red)	(47.2,18.8,92.0,26.5,136.0,34.7)	425640
	4(Cyan)	(19.5,20.0,48.6,36.9,69.8,52.9)	269992
	5(Magenta)	(53.6,25.0,104.5,37.2,161.9,53.7)	255539
	6(Yellow)	(6.9,10.1,15.3,16.5,24.3,24.4)	49093
Large Intestine	1(Blue)	(10.1,14.9,20.0,20.2,30.8,27.2)	40418
	2(Green)	(65.1,30.8,112.1,40.5,162.7,55.5)	87486
	3(Red)	(26.4,29.1,52.6,42.1,73.9,54.4)	85436
	4(Cyan)	(39.4,26.0,81.7,37.6,109.3,46.9)	170389
	5(Magenta)	(100.3,38.9,146.1,48.6,191.2,58.4)	57366
	6(Yellow)	(50.4,19.0,101.8,28.8,131.8,35.1)	227890
* Centroid: (R_mean,G_mean,B_mean,R_var,G_var,B_var)			

표 5. 군집 별 훈련 셋과 테스트 셋 비율

Learning Coverage	Label	Cluster_idx					
		1	2	3	4	5	6
1	Stomach	0.73	0.81	0.90	0.91	0.96	0.72
	Small Intestine	0.76	0.79	0.76	0.83	0.94	0.72
	Large Intestine	0.96	0.92	0.66	0.68	0.85	0.71
0.944	Stomach	0.63	0.45	0.67	0.35	0.19	0.68
	Small Intestine	0.69	0.64	0.80	0.50	0.28	0.05
	Large Intestine	0.00	0.28	0.20	0.56	0.09	0.83
0.889	Stomach	0.49	0.28	0.24	0.00	0.01	0.58
	Small Intestine	0.66	0.74	0.70	0.70	0.02	0.09
	Large Intestine	0.14	0.03	0.70	0.57	0.00	0.69

특성 편향도가 1일 때의 대장 레이블의 6번 군집을 보면 훈련 셋과 테스트 셋 비율이 0.71인 것을 볼 수 있는데, 이 경우는 전체 훈련 데이터의 개수와 테스트 개수의 비율과 동일하기 때문에 데이터 균등성이 높다고 볼 수 있다.

각각의 캡슐내시경 데이터마다 영상의 개수가 다르고, 위, 소장, 대장의 데이터 개수가 다르기 때문에 이것이 모델의 성능에 영향을 끼칠 수 있다. 이를 방지하기 위해 세 개의 훈련 데이터 셋의 데이터 개수와 레이블 별 데이터 개수를 동일하게 구성했다. 세개의 훈련 데이터 셋의 부위 별 개수 중 가장 개수가 적은 FS=0.889 데이터 셋의 위의 개수에 맞추어 각 부위마다 122,083개의 데이터에 맞추어 각 훈련 데이터 셋에서 부위마다 무작위로 선택하여 훈련 데이터 셋을 구성했다. 따라서 각 훈련 데이터 셋의 개수는 각 부위별로 122,083장의 데이터로 구성되며 총 366,249개의 데이터로 이루어졌다.

4.3. 특성 편향도 비율에 따른 모델 성능 비교 분석

4.3절에서는 앞서 생성된 3개의 훈련 데이터 셋을 사용하여 학습을 수행할 위장관 랜드마크 분류 모델을 설계하고, 특성 편향도 비율에 따른 모델의 학습 성능을 비교하도록 한다. 학습 모델의 구조는 그림 12와 같이 총 5개의 합성 곱 신경망 층과 2개의 완전 연결 층으로 구성되어 있다.

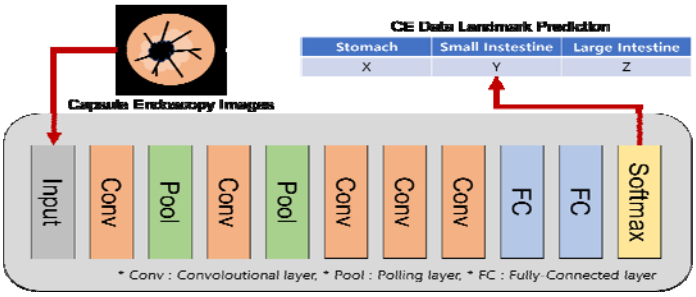


그림 12. 위장관 랜드마크 분류기 모델 구조

해당 모델을 4.2절에서 생성한 훈련셋에 대해 위장, 소장, 대장의 장수를 각각 같게 임의로 선택한 다음 50에폭만큼 학습을 진행하였다. 이때 각각의 훈련셋에 대해 정확도, 정밀도와 재현율은 표 6과 같으며, 특성 편향도가 높을수록 더 높은 정확도를 보였음을 확인할 수 있다. 또한 정확도와 재현율의 조화평균인 F1 점수를 계산하였을 때, 학습에서 제외된 군집이 있는 위장관의 경우 점수가 낮게 나오는 것을 확인할 수 있었다. 예외의 경우가 존재하였는데, 편향도가 0.944인 훈련 셋을 학습한 경우에 위의 F1 점수보다 편향도가 0.889인 훈련 셋을 학습한 경우에 위의 F1점수가 더 높게 나타났다. 이러한 경우는 위의 군집 5 때문에 나타난 것으로 보이며, 그림 11 (a)에서 청록색인 군집 5의 경우 군집에 포함된 데이터의 개수가 3114개에 불과하지만 데이터가 넓게 퍼져서 분포된 것을 확인할 수 있다. 이처럼 데이터의 개수가 적지만 넓게 분포되어 모델의 학습에 방해가 된 것으로 분석된다.

표 6. 모델의 정확도, 정밀도와 재현율

Learning Coverage	Accuracy	Label	Precision	Recall
1	0.83	Stomach	0.78	0.74
		Small Intestine	0.85	0.91
		Large Intestine	0.80	0.68
0.944	0.70	Stomach	0.40	0.85
		Small Intestine	0.81	0.80
		Large Intestine	0.64	0.44
0.889	0.66	Stomach	0.67	0.82
		Small Intestine	0.90	0.56
		Large Intestine	0.47	0.83

표 7의 경우 각각의 특성 편향도로 학습시킨 모델에 대해 실제 레이블과 모델이 예측한 레이블의 비율을 표시하였다. 전체적으로 잘못 판단할 경우 위와 대장은 소장으로, 소장은 대장으로 잘못 판단하는 경우가 많은 것을 확인할 수 있다. 이는 위와 소장이, 소장과 대장이 각각 연결되어 있기 때문이라고 추측할 수 있다. 표 7의 특성 편향도가 0.944인 경우에 대장에 대한 성능이 좋지 않음을 확인할 수 있는데, 이는 대장에 대한 군집 중 하나의 군집에 속한 데이터를 학습할 수 없었기 때문에 대장에 대한 모델의 성능이 저하된 것으로 판단된다. 편향도가 0.889인 경우에는 실제 레이블은 소장이지만 대장으로 잘못 분류한 경우가 많은데, 이 또한 대장에 대한 정보가 부족하게 학습됐기 때문이라고 판단된다.

표 7. 모델의 실제 레이블과 예측 레이블 비율

Learning Coverage	Predicted Label	Actual Label		
		Stomach	Small Intestine	Large Intestine
1	Stomach	0.74	0.01	0.03
	Small Intestine	0.23	0.91	0.29
	Large Intestine	0.03	0.08	0.68
0.944	Stomach	0.85	0.10	0.17
	Small Intestine	0.07	0.80	0.39
	Large Intestine	0.08	0.10	0.44
0.889	Stomach	0.82	0.05	0.06
	Small Intestine	0.07	0.56	0.11
	Large Intestine	0.11	0.39	0.83

5. 결 론

본 논문에서는 훈련 데이터 셋이 전체 데이터 셋의 특성을 포함한 정도를 나타내기 위한 메트릭인 특성 편향도를 제안하였으며, 이를 기반으로 하여 캡슐내시경 데이터의 위장관 분류기 모델의 성능에 대해 분석하였다. 분석 결과 전체 데이터 셋의 특성 18개 중 훈련 데이터 셋이 모든 특성을 반영할 경우 83%, 17개의 특성을 반영할 경우 70%, 16개의 특성을 반영할 경우 66%의 정확도를 확인할 수 있었다. 이와 같은 결과는 특성 편향도가 높은 훈련 데이터 셋의 경우 모델의 학습 성능이 높다는 것을 나타내며, 특성 편향도의 중요성을 보여준다. 향후에는 훈련에 사용하는 데이터의 개수를 줄일 때 특성 편향도에 따른 모델 학습 시간과 정확도, 군집 별 중요도 등을 분석할 예정이다.

참고 문헌

[1] Shen, Wei, et al. "Multi-scale convolutional neural networks for lung nodule classification." *International Conference on Information Processing in Medical Imaging*. Springer, Cham, 2015.

[2] Setio, Arnaud Arindra Adiyoso, et al. "Pulmonary nodule detection in CT images: false positive reduction using multi-view convolutional networks." *IEEE transactions on medical imaging* 35.5 (2016): 1160-1169.

[3] Kawahara, Jeremy, and Ghassan Hamarneh. "Multi-resolution-tract CNN with hybrid pretrained and skin-lesion trained layers." *International Workshop on Machine Learning in Medical Imaging*. Springer, Cham, 2016.

[4] Nie, Dong, et al. "3D deep learning for multi-modal imaging-guided survival time prediction of brain tumor patients." *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, Cham, 2016.

[5] Zhang, Qi, et al. "Deep learning based classification of breast tumors with shear-wave elastography." *Ultrasonics* 72 (2016): 150-157.
[6] Zou, Yuexian, et al. "Classifying digestive organs in wireless capsule endoscopy images based on deep convolutional neural network." *2015 IEEE International Conference on Digital Signal Processing (DSP)* (pp.1274-1278). IEEE, 2015.
[7] Yuan, Yixuan, and Max Q-H. Meng. "Deep learning for polyp recognition in wireless capsule endoscopy images." *Medical physics* 44.4 (2017): 1379-1389.
[8] Jia, Xiao, and Max Q-H. Meng. "A deep convolutional neural network for bleeding detection in wireless capsule endoscopy images." *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (pp.639-642). IEEE, 2016.
[9] Maas, Andrew L., Awni Y. Hannun, and Andrew Y. Ng. "Rectifier nonlinearities improve neural network acoustic models." *Proc. icml*. Vol. 30. No. 1. 2013.
[10] Jin, Xiaojie, et al. "Deep learning with s-shaped rectified linear activation units." *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.
[11] Park, Ye-Seul, Byungjeong Lee, and Jung-Won Lee, . “Quality Evaluation Metric of Data Set to Develop Data-driven Software for Capsule Endoscopic Images.” *Proceedings of the 21st Korea Conference on Software Engineering(KCSE)* (pp.78-87). 2019
[12] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
[13] Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.
[14] Mani, Senthil, et al. "Coverage Testing of Deep Learning Models using Dataset Characterization." *arXiv preprint arXiv:1911.07309* (2019).
[15] He, Jun-Yan, et al. "Hookworm detection in wireless capsule endoscopy images with deep learning." *IEEE Transactions on Image Processing* 27.5 (2018): 2379-2392.
[16] Seguí, Santi, et al. "Generic feature learning for wireless capsule endoscopy analysis." *Computers in biology and medicine* 79 (2016): 163-172.

순환 인공 신경망을 활용한 코드 변경 추천 시스템의 학습 시간 단축 방법 연구

*배병일 ○, *강성원, **이선아
*한국과학기술원, **경상대학교
bae.b.i@kaist.ac.kr, sungwon.kang@kaist.ac.kr, saleese@gnu.ac.kr

1. 서 론

소프트웨어 개발 과정에서 많은 개발자는 소프트웨어 진화를 위해 또는 버그를 고치기 위해 기존 코드를 수정한다. 수정 작업에서 개발자들은 수정이 필요한 파일을 찾기 위해 많은 시간을 소모한다. 그래서 연구자들은 파일 찾는 시간을 줄이기 위해 개발자의 현재 작업 중에 다음에 수정할 코드를 추천해주는 코드 변경 추천 시스템을 연구하고 있다. 코드 변경 추천 시스템은 다양한 방법을 활용하여 연구되고 있는데 그중 하나가 사용자의 동작 히스토리를 활용하는 방법이다. 대표적인 도구로 Mylyn 데이터를 이용한 MI가 있다. MI는 동작 히스토리의 연관규칙을 마이닝 하여 수정할 파일을 추천하는 시스템이다. 이러한 추천 시스템에서는 정교한 문맥을 구성하는 윈도우 설정이 정확도를 결정한다. 우리는 최적의 윈도우 크기를 효율적으로 결정할 수 있는 방법을 제안한다.

2. 기존 연구

우리는 기존에 Mylyn 데이터에 기계 학습을 적용한 연구를 진행하였다. 최신의 연구이며 더 좋은 정밀도로 추천하는 순환 인공 신경망을 이용한 코드 변경 추천 시스템(RNN-CRS)은 사용자 동작 히스토리 데이터를 순서 정보를 유지한 채 기계 학습을 하여 코드 변경을 추천 시스템이다. 개발자가 하나의 소프트웨어 진화 작업을 하는 동안 기록된 파일 방문과 편집 히스토리들을 동작 트레이스의 기본 단위로 정의하고 여기에는 저장된 개발자의 동작을 변경(Edit)과 열람(View)으로 나눈다. 변경은 개발자가 변경 행위를 기록한 로그를 말하며, 열람은 개발자가 열람 행위를 기록한 로그 말한다. 추천 시스템과 관련된 연구에서 사용되는 '문맥'이란 단어는 '추천을 발생시키는 질의문(Query)을 의미한다. 이를 순환 인공 신경망을 이용한 추천 시스템은 슬라이딩 윈도우를 적용하여 최근 동작 순서 정보를 유지한 문맥을 구축하고 추천 모델을 만든다. 슬라이딩 윈도우를 적용하는 이유는 같은 양의 데이터를 활용하더라도 더 많은 문맥을 생성하여 학습할 수 있기 때문이다.

하지만 RNN-CRS로 적절할 슬라이딩 윈도우를 가진 추천 모델을 형성하는 과정에는 많은 컴퓨터 자원이 필요하다. 슬라이딩 개수만큼 추천 모델을 만들려면 같은 노력을 반복해야 하기 때문이다. 적절한 슬라이딩 윈도우는 학습했던 추천 모델의 슬라이딩 윈도우 크기 중에서 가장 정밀도가 높은 슬라이딩 윈도우를 말한다. 순환 신경망을 이용한 코드 변경 추천 시스템의 경우 슬라이딩 윈도우 크기를 3에서 6까지 변경해가며 정밀도가 가장 높은 슬라이딩 윈도우 크기를 찾고 그 추천 모델을 추천 시스템에 활용하는 방법을 제안하였다.

소프트웨어 제품마다 적절한 슬라이딩 윈도우 크기가 다르기 때문에 각자의 적절한 슬라이딩 윈도우 크기를 구하여 학습해야 한다. 그리고 적절한 슬라이딩 윈도우 크기는 데이터가 축적되면서 바뀌기 때문에 하나의 슬라이딩 윈도우 크기만 활용하여 새로운 데이터를 학습해서는 안 된다.

RNN-CRS는 데이터 축적에 따른 변하는 슬라이딩 윈도우 크기를 반영하지 못하였고 반영하기 위해서는 슬라이딩 개수만큼의 추천 모델을 모두 학습해야 하는 문제점이 있다. 즉, 변하는 적절한 슬라이딩 윈도우 크기를 고려하기 위해서는 기존 연구의 방법은 많은 시간이 소모하게 된다.

3. 제안 방법

만약 학습 전에 정밀도가 높은 추천 모델을 찾아낼 수 있다면, 슬라이딩 윈도우 개수만큼 추천 모델을 학습하지 않아도 될 것이다. 이 방법을 이용하면 기존 연구보다 학습에 걸리는 시간을 최대 약 1/(학습에 이용했던 슬라이딩 윈도우 개수)까지 줄일 수 있다. 즉, 앞으로 제안할 방법에서는 선택한 슬라이딩 윈도우만 학습하고 필요 없는 윈도우에 대해 학습하지 않기 때문에 시간이 줄어들 것이다.

제안 방법에서는 슬라이딩 윈도우 크기가 다른 추천 모델 중에서 가장 높은 정밀도를 가진 추천 모델을 추측하여 선택 학습해서 추천 시스템에 활용하기 위해 'W 테스트'와 'P 테스트'라는 개념을 소개한다. W 테스트는 처음 만들어진 추천 모델을 활용하여 새롭게 추가된 데이터를 테스트하는 것을 말하고 P 테스트는 일반적인 테스트 과정으로 추천 정밀도를 계산하기 위한 테스트이다. W 테스트 결과 가장 큰 값을 보인 추천 모델을 선택하여 그 모델만 학습하고 시스템에 활용한다. P 테스트는 검증 과정이기도 하지만 제안 방법에서 W 테스

트 결과를 보조하는 판단의 근거로 P 테스트 값을 사용하기 때문에 제안 방법에 포함을 시켰다.

4. 실험 방법

본 연구의 목적은 기존 연구에서 고려하지 못하는 변화하는 슬라이딩 윈도우 크기를 고려하면서 학습량을 줄여 시간을 아끼는 방법을 활용할 수 있는지에 대해 확인하고 시간을 얼마나 줄일 수 있는지 확인한다.

연구 질문 1: W 테스트 결과가 가장 정밀도가 높은 추천 모델을 선택할 수 있는가?

연구 질문 2: W 테스트를 학습에 활용한다면 얼마나 많은 시간이 줄일 수 있는가?

연구 질문 1을 평가하기 위한 기준으로 추천의 정확도인 정밀도, 재현율, F1 점수를 평가 기준으로 사용하였다.

연구 질문 2를 평가하기 위한 기준으로 데이터의 학습을 끝낸 뒤 소요되었던 학습 시간을 측정한다. 기존 방법은 각 소프트웨어 제품을 슬라이딩 윈도우 크기 3 ~ 6의 추천 모델의 생성하는 시간을 모두 더한 값을 측정하고 제안 방법은 각 슬라이딩 윈도우 크기에서 초기 추천 모델을 생성하는 시간과 새로운 데이터를 학습하는데 사용한 선택된 추천 모델의 학습 시간만 더한 값을 측정한다.

실험에 사용할 데이터는 버그 질라 시스템에서 제공하는 오픈소스 프로젝트에 대해 수집한 1,300여 개의 동작 트레이스를 활용한다. 실험 데이터가 이전에 활용했던 데이터로 한정되어 있음으로 기존 있던 데이터를 나누어서 데이터가 쌓이는 과정을 재현하였다. 기존 데이터를 시간 순서대로 나누어 첫 번째 데이터 집합을 초기 추천 모델로 생성하여 활용하고 두 번째, 세 번째 데이터 집합을 새로 축적되는 데이터라 가정하고 학습한다.

5. 실험 결과

ECF에서는 가장 효율적인 결과를 보여준다. 가장 큰 W 테스트 결과가 가장 큰 P 테스트 결과를 보여준다. MDT와 Platform의 경우에도 W 테스트의 값이 가장 큰 값이 P 테스트 결과값도 크다.

PDE는 적절한 슬라이딩 윈도우 크기가 변하는 것을 보여주는 예이다. 초기 180개 데이터와 후기 177개 데이터의 적절한 슬라이딩 윈도우 크기가 다르기 때문이다. 그래서 PDE의 경우 New2 단계에서 슬라이딩 윈도우 크기 6이 윈도우 크기 4보다 0.0009 큰 값이 나왔으나 P 테스트 결과는 슬라이딩 윈도우 크기 4의 추천 모델이 0.0101 높은 것으로 나온다. 그 이유는 이 전 학습 단계의 P 테스트 값(정밀도)이 해당 학습 단계에 영향을 주기 때문이다.

연구 질문 1에 대한 답변은 W 테스트만으로 적절한 슬라이딩 윈도우를 가지는 추천 모델을 찾을 수 없다. 이를 확인하는 실험은 PDE 실험 결과이다. W 테스트 결과값이 높더라도 W 테스트 결과값이 낮은 값의 추천 모델보다 정밀도(P 테스트 값)가 낮을 수 있다. 그 이유는 이전 단계에서 측정된 정밀도(P 테스트)가 해당 학습 단계에 영향을 끼치기 때문이다. 그렇기 때문에 이전 학습 단계의 P 테스트도 판단의 근거로 포함되어 추천 모델을 선택해야 한다.

연구 질문 2에 대해서는 제안 방법을 사용하면 시간을 줄일 수 있다. 표1을 참고하면 가장 안 좋은 실험 결과였던 MDT의 경우에는 약 10%의 시간을 절약할 수 있었다. 그 이유는 학습 시간이 다른 소프트웨어에 비해 매우 적은 시간 안에 학습이 끝난다. 열람과 변경의 비율이 10:1로 다른 소프트웨어보다 그 비율이 균등하지 않기 때문에 생성된 문맥이 적고 이를 추천에 이용되지 못한다. 게다가 최초 추천이 일어나는 시기가 매우 늦기 때문에 초기 추천 모델을 계속 업데이트하기 때문에 비효율적이다. 반면에 실험 결과가 좋은 ECF와 PDE, Platform은 약 68%, 49%, 67%의 시간 줄일 수 있었다.

6. 결론

본 논문에서는 학습 시간을 단축하기 위해 최초로 W 테스트를 고안하였다. 이 W 테스트를 이용하여 RNN-CRS의 학습에 적절한 슬라이딩 윈도우 크기를 구해 선택된 추천 모델만 학습함으로써 시간을 절약할 수 있다. 향후 연구로는 이 방법을 다양한 소프트웨어 분야의 추천 시스템에 적용하여 본 연구의 폭을 넓혀보고 싶다. 슬라이딩 윈도우를 사용하는 다른 분야의 예측 시스템이나 데이터에 Frame을 씌운 시스템에 본 연구 방법을 적용하여 적절한 슬라이딩 윈도우 크기를 빠른 시간에 찾아 시간 소모를 줄일 수 있는지 확인해야 한다.

표 1 각 소프트웨어 제품의 실험결과

윈도우 크기		3	4	5	6		3	4	5	6		절약 시간
소모 시간	ECF	00:28:53	00:27:29	00:26:05	00:26:13	**01:48:40**	00:28:48	00:01:53	00:01:58	00:01:56	**00:34:36**	68.15%
	MDT	00:07:57	00:07:25	00:07:16	00:08:02	**00:30:40**	00:07:05	00:06:50	00:06:47	00:06:42	**00:27:24**	10.66%
	PDE	18:05:21	17:40:00	16:53:45	15:53:15	**68:32:21**	00:55:14	17:24:22	00:44:16	15:59:28	**34:54:01**	49.08%
	Platform	16:58:02	18:14:27	18:14:07	17:02:57	**70:29:33**	16:44:48	00:40:36	00:42:12	00:39:22	**23:29:18**	66.68%

ResNet의 뉴런 활성 값과 Epoch 간 상관관계 분석을 통한 화이트 박스 방법의 모델 평가

박재현 ○ 최현재 채흥석

부산대학교 전기전자컴퓨터공학과

wogus230vvv@naver.com, gonoki@pusan.ac.kr, hschae@pusan.ac.kr

Model Evaluation of White Box Method based on through Correlation Analysis between Neuron Activation Value and Epoch in ResNet

Jae-Hyun Park○ Hyun-Jae Choi Heung-Seok Chae

Dept. of Electrical & Computer Engineering, Pusan National University

요 약

ResNet은 이미지 분류를 위한 대표적인 딥러닝 모델 중의 하나이다. 오늘날 딥러닝은 다양한 분야에서 연구되고 있다. 화이트 박스 평가 방법은 모델 내부 구조와 내부 뉴런 정보를 고려한 모델 평가가 가능하다. 그러나 기존 화이트 박스 평가 연구에서는 뉴런 기반의 커버리지 평가 방법을 제시하였지만 평가 결과와 결함과의 관계에 대해서는 제시하지 않았다. 본 논문에서는 모델의 출력 결정에 영향이 적은 작은 활성 값을 가지는 뉴런 수의 비율과 epoch 간 상관관계를 통하여 안정화 추세 모델 평가 방법을 제안한다. 본 연구에서는 이미지 분류의 3가지 데이터 세트 CIFAR-10, CIFAR-100, Fashion-MNIST에 대해 사례 연구를 수행하였다. 사례 연구에서는 전체 모델 분석과 layer 별 모델 분석 두 가지 방법으로 분석하였다. 모델 전체 분석 결과 CIFAR-10, CIFAR-100, Fashion-MNIST의 모든 데이터 세트에서 0.934 이상의 상관 계수를 가지며 높은 안정화 추세를 보였다. Layer 별 모델 분석 결과 각각 0.930, 0.948 정확도를 보인 CIFAR-10과 Fashion-MNIST 데이터 세트는 모든 구간에서 높은 안정화 추세를 보였다. 그러나 0.705의 정확도를 보인 CIFAR-100의 데이터 세트에서는 후반부 layer 구간에서 0.143 상관 계수를 가지며 낮은 안정화 추세를 보였다.

1. 서 론

합성곱 신경망(Convolutional Neural Network)은 딥러닝 모델의 한 종류로 layer 중 하나 이상에서 합성곱을 사용하는 신경망이다[1]. 합성곱 신경망은 필터를 이용하여 입력 데이터로부터 특징(feature)을 추출하고, 추출한 특징을 기반으로 출력 데이터를 결정한다.

ResNet(Residual neural network)[2]은 대표적인 합성곱 신경망 모델 중 하나이다. ResNet은 ImageNet에서 개최하는 2015 ILSVRC 대회에서 오류율 3.57%로 우승한 뛰어난 성능을 지니고 있는 모델이다. ResNet은 악성 소프트웨어 분류[3], 바이러스 감지[4], 차량 분류[5] 연구 등에서 활용되고 있다.

대부분의 ResNet 기반 연구들은 블랙박스 평가 방법과 화이트 박스 평가 방법을 사용하고 있다. 블랙박스 평가 방법은 정확도와 같은 분류 결과만을 고려하여 모델을 평가한다. 화이트 박스 평가 방법은 모델의 내부 구조와 layer 간의 뉴런의 가중치, 활성 값과 같은 정보를 기반으로 평가하는 방법이다.

딥러닝 모델은 복잡한 특징을 가지고 있어 뉴런의 활성 값 같은 값을 고려한 평가가 필요하다. 이에 따라 화이트 박스 평가 방법이 연구되고 있다[6][7][8].

기존 화이트 박스 평가 방법은 뉴런의 활성화 여부[6] 및 뉴런 활성 값의 조합[7], 활성화 값의 범위[8]를 기반으로 평가하였다. 그러나 기존 연구는 뉴런 기반의 커버리지 평가 방법을 제시하였을 뿐, 평가 결과와 결함과의 관계를 충분히 보이지 않은 약점이 있었다.

본 연구에서는 기존 연구들의 약점을 보완하여 뉴런의 활성 값(activation value)과 모델의 학습 세대를 의미하는 epoch을 이용하여 모델의 안정화 추세를 평가하는 방법을 제안한다. 모델의 안정화는 출력에 영향이 작은 뉴런의 수가 증가하면 모델 출력이 안정화된다. 모델 출력의 안정화를 분석하기 위하여 작은 활성 값을 갖는 뉴런과 epoch 간 상관 관계 분석을 수행하고 모델의 안정화 추세를 평가하는 방법을 제안한다. 제안하는 평가 방법을 CIFAR-10[9], CIFAR-100[10], Fashion-MNIST[11] 3개의 데이터 세트에 사례 연구를 수행하여 안정화 추세를 평가하였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 화이트 박스 모델 평가 연구를 소개하고 본 연구 기법과의 차이점을 설명한다. 3장에서는 본 연구에서 제안하는 ResNet 뉴런과 epoch 간 상관관계 분석을

통한 화이트 박스 모델 평가 방법을 보인다. 4장에서는 본 논문에서 제안한 모델 평가 방법을 3개의 데이터 세트에 수행한 환경 설정을 소개하고 사례 연구를 수행한 분석 결과를 기술한다. 5장에서는 본 연구의 결론과 향후 연구 방향에 대해 기술한다.

2. 관련 연구

Pei et al.[6]은 뉴런의 활성 값이 threshold value 이상의 값을 가지면 커버 되었다고 측정하는 뉴런 커버리지 분석 방법을 제안하였다. Pei가 제안한 뉴런 커버리지는 실세계의 딥러닝 시스템을 체계적으로 테스트하기 위한 화이트 박스 프레임워크로 딥러닝 모델의 정확도를 향상시키기 위해 사용할 수 있다.

Sun et al.[7]은 Modified Condition/Decision Coverage를 DNN에 맞게 변형하였다. 뉴런의 sign 또는 일정 이상의 뉴런 활성 값 변화가 다음 layer의 뉴런 활성 값이나 sign의 변화를 유발하는지 측정하는 커버리지 분석 방법을 제안하였다.

Ma et al.[8]은 학습 데이터에 대한 뉴런의 활성 값으로 범위 및 경계로 뉴런 범위를 제한하였다. 제한한 뉴런 범위를 이용하여 뉴런 수준의 커버리지 방법과 layer 수준의 커버리지 방법을 제안하였다.

그러나 Pei[6], Sun [7], Ma[8]이 제안한 기존의 뉴런 커버리지 기법들은 제안한 평가 기준을 달성하였을 때 모델 평가 결과와 결함과의 관계에 대해서 제시하지 못하는 약점이 있다.

본 연구에서는 기존 연구에서 평가 결과와 결함과의 관계를 제시하지 못한 약점을 epoch이 증가함에 따라 작은 활성 값을 가진 뉴런의 비율이 늘어남과 정확도를 비교하여 모델의 안정화 추세 평가 방법을 제시한다.

3. ResNet의 뉴런 활성 값과 epoch 간 상관관계 분석을 통한 화이트 박스 방법 모델 평가

본 장에서는 모델 평가 기준 결정과 상관관계 분석을 통한 모델 평가 방법을 제안한다. 모델 평가 기준 결정은 모델을 구성하는 뉴런의 활성 값의 변화를 분석하여 모델을 평가하는 기준을 결정한다. 상관관계 분석을 통한 모델 평가 방법은 평가하기 위한 절차와 상관관계를 이용한 안정화 추세 평가 방법을 설명한다.

3.1 모델 분석 기준 결정

활성 값은 테스트 데이터 입력에 따른 뉴런이 갖는 값을 나타낸다. 활성 값은 들어오는 입력 값과 가중치(weight)의 곱과 바이어스(bias)의 합, 활성 함수를 통하여 계산된다. 활성 함수는 입력 값을 활성화할 것인지 하지 않을 것인지에 대하여 결정한다. 활성 값이 매우 작은 뉴런은 출력 데이터 결정에 거의 영향을 주지 않고, 그렇지 않은 뉴런은 출력 데이터 결정에 영향을 준다. 이에 따라 본 연구에서는 epoch이 증가함에 따라 출력 데이터 결정에 거의 영향을 주지 않는 뉴런과 영향을 주는 뉴런이 점차 구분될 것이라 가정한다. 뉴런이 구분되는 것을 확인 하기 위하여 본 연구에서는 평균 활성 값을 이용한다.

$\mathrm{MeanAV(n)}$은 n번 뉴런이 가지는 평균 활성 값이다. 뉴런의 $\mathrm{MeanAV(n)}$은 테스트 데이터가 입력되었을 때 뉴런이 가지는 활성 값들의 합을 전체 데이터 수로 나눈 값이다. n번째 뉴런이 가지는 평균 활성 값을 반환하는 식은 수식 (1)과 같다.

$$\mathrm{MeanAV(n)} = \frac{\sum_{x \in TD} \phi(x,n)}{|TD|} \quad \cdots\cdots\cdots\cdots \quad (1)$$

$\phi(x,n)$은 데이터 x가 입력되었을 때 n번 뉴런이 가지는 활성 값이다. TD는 테스트 데이터들의 집합을 의미한다. 그림 1은 CIFAR-10의 airplane class 테스트 데이터를 입력할 때 모델 내부 뉴런이 가지는 평균 활성 값의 분포를 분석하기 위해 이용한 히스토그램을 나타낸 것이다.

그림 1에서 epoch이 증가함에 따라 작은 활성 값을 가지는 뉴런의 수가 늘어나는 경향을 보였다. CIFAR-10의 airplane class 테스트 데이터뿐만이 아니라 다른 9가지 class에서도 작은 활성 값을 가지는 뉴런의 수가 증가하는 같은 경향을 보였다. 또한 CIFAR-100과 Fashion-MNIST 데이터 세트의 class 테스트 데이터들에 대해서도 같은 경향의 결과를 보였다.

학습이 진행됨에 따라 뉴런의 활성 값 변화를 분석하여 출력 데이터 결정에 거의 영향을 주지 않는 뉴런이 점차 증가함을 확인하였다. 따라서 epoch 증가에 따라 출력 데이터 결정에 거의 영향을 주지 않는 뉴런과 그렇지 않은 뉴런이 점차 구분됨을 확인할 수 있다.

본 논문은 히스토그램을 통해 확인한 영향을 주지 않는 뉴런을 구분하기 위해 최대 평균 활성 값을 이용한다. 최대 평균 활성 값을 구하기 위해서 뉴런의 평균 활성 값의 집합을 이용한다. $\mathrm{AV_{mean}}$은 모델을 구성하는 모든 뉴런의 평균 활성 값의 집합이다. 뉴런의 평균 활성 값 집합 $\mathrm{AV_{mean}}$은 수식 (2)와 같다.

$$\mathrm{AV_{mean}} = \{\mathrm{MeanAV(n)} \mid \mathrm{n} \in \mathrm{N}\} \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (2)$$

최대 평균 활성 값은 모델을 구성하는 각 뉴런이 가지는 평균 활성 값의 최대값이다. 출력 데이터 결정에 거의 영향을 주지 않는 활성 값이 매우 작은 뉴런을 구분하는 기준을 위해 최대 평균 활성 값을 사용하였다. 수식 (3)은 최대 평균 활성 값의 계산을 위한 식이다.

$$\text{최대 평균 활성 값: } \mathrm{Max(AV_{mean})} \quad \cdots\cdots\cdots\cdots \quad (3)$$

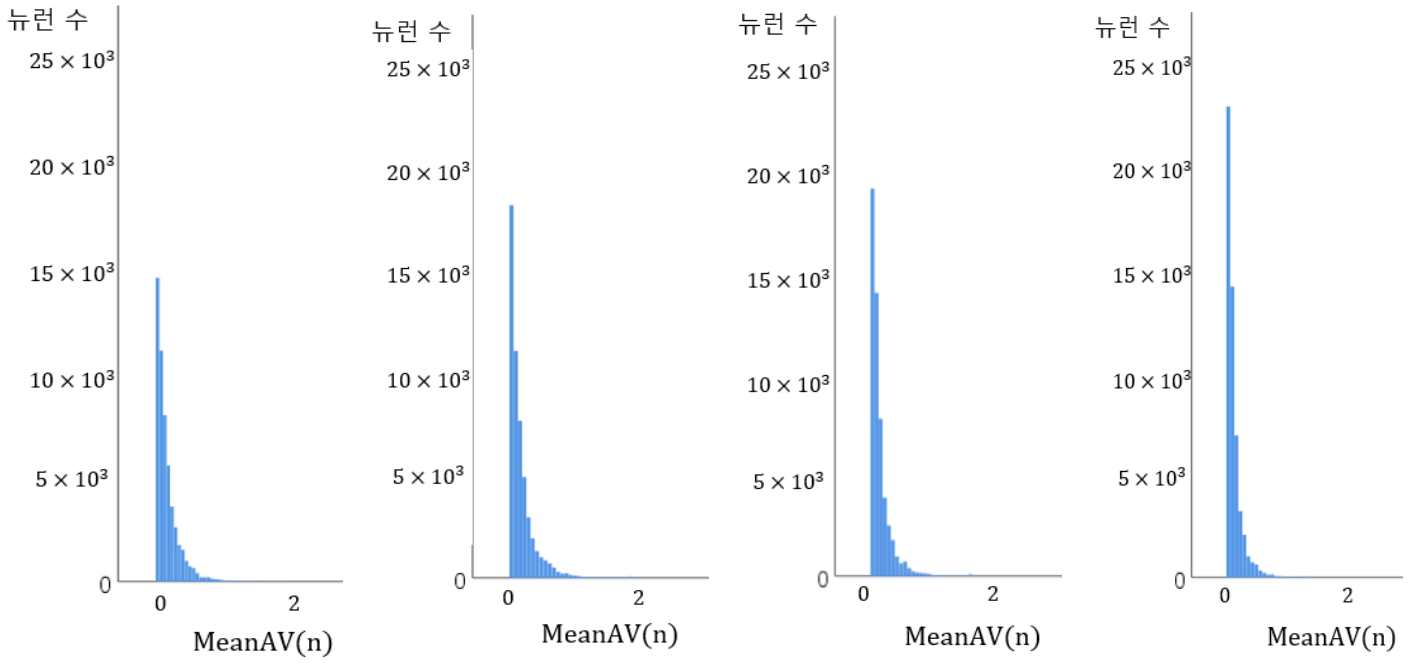


그림 1. CIFAR-10 airplane class 테스트 데이터 입력에 대한 뉴런의 평균 활성 값

영향을 주지않는 작은 활성 값을 가지는 뉴런을 구분하기 위하여 최대 평균 활성 값의 k% 미만을 이용한다. 최대 평균 활성 값 k% 미만의 값을 가지는 뉴런의 집합을 하위 k% 뉴런으로 정의한다. 수식 (4)는 하위 k% 뉴런 집합을 구하기 위한 정의이다.

하위 k% 뉴런

$$= \{n \mid n \in N \text{ and } MeanAV(n) < \max(AV_{mean}) \times 0.01 \times k\} \quad \cdots (4)$$

그러나 하위 k% 뉴런 수는 모델의 layer의 종류와 layer의 수에 따라서 크기가 변하여 구성하는 뉴런 수가 변할 수 있다. 따라서 뉴런 수를 비율로 변환하여 사용하였다. 하위 k% 뉴런의 비율을 계산하는 식은 수식 (5)와 같다.

$$\text{하위 k\% 뉴런 비율} = \frac{|\text{하위 k\% 뉴런}|}{|N|} \times 100\% \quad \cdots\cdots\cdots\cdots (5)$$

3.2 상관관계 분석을 통한 모델 평가 방법

3.1절에서 학습이 진행 될수록 출력 데이터 결정에 거의 영향을 주지 않는 작은 활성 값을 가지는 뉴런의 수가 늘어나는 것을 확인하였다. 학습이 진행되면 될수록 뉴런의 활성 값이 0에 수렴하는 하위 k% 뉴런의 비율이 늘어나, 작은 활성 값을 가지는 뉴런이 구분되면 전체적인 모델이 안정화가 된다. 따라서 본 연구에서는 epoch이 증가함에 따라 하위 k% 뉴런 비율을 이용하여 모델의 안정화 추세를 평가하였다. epoch이 증가함에 따라 하위 k% 뉴런 비율의 변화를 확인하기 위해 일정 epoch이 증가함에 따라 모델을 생성한다. 생성한 모델은 각 class 별 테스트 데이터에 대한 뉴런 비율을 구할 때 마지막 layer는 출력데이터를 표현하기 때문에 분석 대상에서 제외하였다.

상관관계 분석 방법은 두 변수 간에 어떤 선형적 또는 비선형적 관계를 갖고 있는지를 분석하는 방법이다. 본 논문에서 상관관계 분석을 이용할 때에는 대표적인 상관 계수인 피어슨 상관 계수를 이용하였다. 상관 계수 값의 크기가 [0.7-1]의 값을 가지면 높은 상관관계, [0.4-0.7)의 값을 가지면 평범한 상관관계 [0-0.4)의 값을 가지면 작은 상관관계를 가진다[12].

따라서 하위 k% 뉴런 비율과 epoch 간의 상관관계 분석을 수행하였을 때 상관 계수가 [0.7-1]의 값을 가지면 모델에 영향을 주지 않는 뉴런이 늘어남으로 모델이 안정화 추세가 높다고 평가하고 상관 계수가 [0.4-0.7)의 값을 가지면 모델의 안정화 추세가 평범하다고 하고 평가하고 상관 계수가 [0-0.4) 값을 가지면 안정화 추세가 낮다고 평가한다. 유의 확률은 데이터의 오류가 관찰될 확률로 반대말로 데이터의 신뢰도로 해석할 수 있다. 예를 들어, 0.05의 유의 확률을 가지면 95%의 신뢰도를 가진다.

예를 들어 테스트 데이터에 대한 40, 80, 120, 160 epoch 모델의 각 하위 k% 뉴런 비율이 0.46, 0.47, 0.53, 0.61을 가진다고 하면 이 값들을 이용하여 상관분석을 수행하여 안정화 추세를 평가하였다.

본 논문에서는 전체 모델 분석과 layer 별 모델 분석으로 크게 두 가지 방법으로 모델을 분석한다. 전체 모델 분석은 하위 k% 뉴런 비율을 epoch이 증가함에 따른 변화를 epoch과 상관관계 분석을 이용하여 분석하고 epoch이 증가함에 따라 하위 k% 평균 뉴런

비율 변화를 분석한다. Layer 별 분석은 모델을 분석 대상에서 제외되는 마지막 출력 layer를 제외하고 layer 수에 따라 4등분한다. 4등분 한 layer를 각각 epoch이 증가함에 따른 하위 k% 뉴런 비율 변화를 상관관계 분석을 하고 안정화 추세를 평가한다.

4. 사례 연구

본 장에서는 4.1절에서 CIFAR-10, CIFAR-100, Fashion-MNIST 대상 데이터 세트에 대해서 설명하고, 4.2절에서 본 논문에서 제안한 하위 k% 뉴런 비율을 통하여 전체 모델 분석, layer 별 모델 분석을 위한 환경 설정에 대해서 기술한다. 4.3절은 데이터 세트마다 하위 k% 뉴런 비율과 epoch에 따른 상관관계 분석과 하위 k% 평균 뉴런 비율 표를 통하여 하위 k% 뉴런의 분포 변화를 모델 전체와 layer 별로 구분하여 수행하였을 때의 분석한 결과를 소개한다.

4.1 사례 연구 대상

본 연구에서 제시한 뉴런의 활성 값과 모델의 학습 정도의 관계를 활용한 모델 평가 방법을 평가하기 위해 CIFAR-10, CIFAR-100, Fashion-MNIST를 대상으로 사례 연구를 수행하였다. 표 1은 사례 연구에 이용한 대상 데이터 세트를 기술한다.

표 1. 사례 연구 대상 데이터 세트

데이터 세트명	클래스 수	학습 데이터 수	테스트 데이터 수	Scale	이미지 크기
CIFAR-10	10	50,000	10,000	RGB	32X32
CIFAR-100	100	50,000	10,000	RGB	32X32
Fashion-MNIST	10	60,000	10,000	Gray	28X28

CIFAR-10[9]과 CIFAR-100[10]은 Alex Krizhevsky, Vinod Nair, Geoffrey Hinton가 공개한 이미지로 가로 32픽셀, 세로 32픽셀 사이즈를 가지고 RGB 3가지 채널 값을 가지며 60,000개의 입력 데이터로 50,000개의 학습 데이터와 10,000개의 테스트 데이터를 가진다. 비행기, 사과, 단풍나무, 컵, 말, 개와 같은 종류의 데이터로 이루어져있고 CIFAR-10은 총 10개의 class, CIFAR-100은 총 100개의 class로 구성되어 있다.

Fashion-MNIST[11]는 Kashif Rasul, HanXiao가 공개한 이미지 데이터로 가로 28픽셀, 세로 28픽셀 사이즈로 구성된 회색 조 이미지 데이터들의 집합이다. 70,000개의 입력 데이터를 가지며 60,000개의 학습 데이터와 10,000개의 테스트 데이터를 가진다. 이미지의 종류는 티셔츠, 가방, 바지, 코트와 같은 종류의 데이터로 총 10개의 class로 구성되어 있다.

4.2 사례 연구 환경 설정

대상 데이터를 학습하는 모델은 ResNet을 이용하였다. Layer 수는 56으로 설정하였고 테스트 데이터 입력에 대하여 분석을 하기 위해 모델 내부의 각 뉴런이 가지는 활성 값을 추출하는 코드를 작성하였다. epoch 횟수는 모델의 정확도 값이 수렴하는 세대인 160 epoch으로 설정하였다. 매 epoch 40 마다 모델을 생성하여 각 데이터 세트마다 epoch 40, epoch 80, epoch 120, epoch 160 학습한 4개의 모델을 만들어 총 12개의 모델을 생성하였다. 생성한 후 각 데이터 세트마다 생성된 12개의 모델에 테스트 데이터를 입력한 후 각 뉴런 마다 가지는 활성 값을 추출하였다.

예를 들어 CIFAR-10 데이터 세트에 대해 수행하였을 때 학습 데이터를 이용하여 생성된 epoch 40, epoch 80, epoch 120, epoch 160 4개의 모델 각각 마다 CIFAR-10의 10개의 class 별 테스트 데이터 1,000개씩 넣고 각 입력 마다 각 뉴런이 가지는 활성 값의 평균을 구하였다. 그 중 class 별 테스트 데이터에 대한 가장 큰 평균 활성 값을 가지는 뉴런을 최대값으로 설정하였다. 출력 결과에 영향을 주는 뉴런을 구분하기 위해 하위 뉴런 비율은 0.5%, 1%, 2% 중 epoch과 평균적으로 큰 상관 관계를 보인 1%로 사례연구를 수행하였다. 표 2는 하위 뉴런 비율 0.5%, 1%, 2%과 epoch 간 상관 관계를 나타낸다.

표 2. 하위 k% 뉴런 비율과 epoch 간 상관 관계

데이터 세트명		하위 0.5%	**하위 1%**	하위 2%
CIFAR-10	상관 관계	0.761	**0.940**	0.937
	유의 확률	0.000	0.000	0.000
CIFAR-100	상관 관계	**0.944**	0.934	0.884
	유의 확률	0.000	0.000	0.000
Fashion-MNIST	상관 관계	0.761	**0.943**	0.936
	유의 확률	0.000	0.000	0.000

4.3.1 하위 1% 뉴런 비율과 epoch 간 관계 분석

CIFAR-10, CIFAR-100, Fashion-MNIST 데이터 세트를 대상으로 하위 1% 뉴런 비율과 epoch 간 분석을 수행하였다. 그림 2-4는 본 논문에서 수행한 데이터 세트의 각 class 별 테스트 데이터에 대한 epoch 별 변화에 따른 하위 1% 뉴런 비율의 박스 플롯이다.

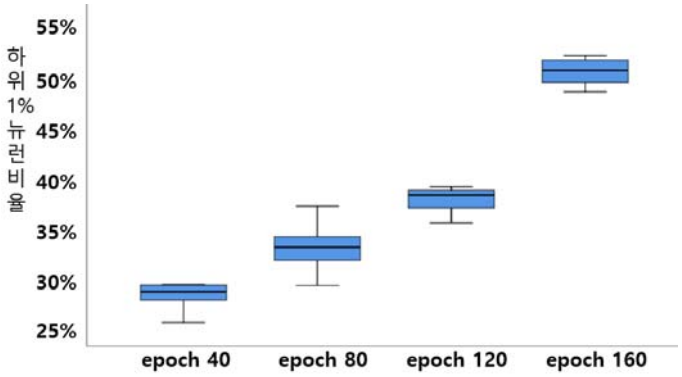


그림2. CIFAR-10 epoch 변화에 따른 하위 1% 뉴런 비율

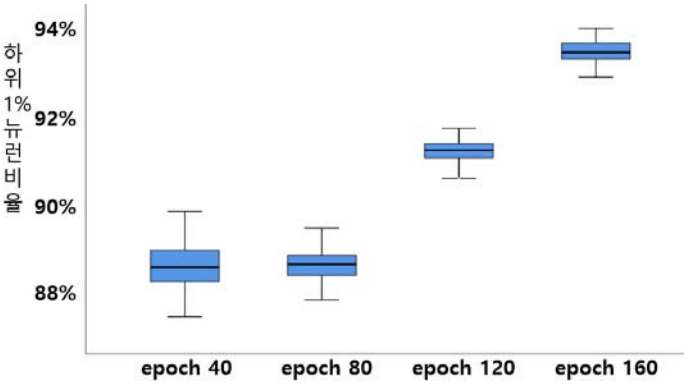


그림 3. CIFAR-100 epoch 변화에 따른 하위 1% 뉴런 비율

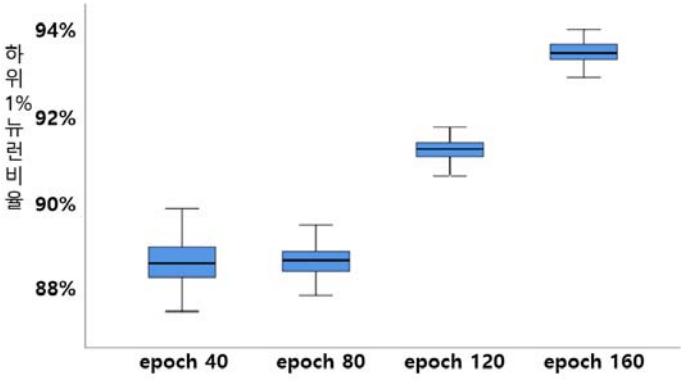


그림 4. Fashion-MNIST epoch 변화에 따른 하위 1% 뉴런 비율

전체 모델 분석 시 CIFAR-10, CIFAR-100, Fashion-MNIST 모두 epoch이 증가함에 따라 하위 1% 뉴런 비율 중간 값이 증가하는 경향을 보였다. CIFAR-10의 데이터 세트에서는 각 epoch에서 28.6%, 32.9%, 38.0%, 50.1%를 가졌고 CIFAR-100은 88.5%, 88.6%, 91.1%, 93.2% 가지며 Fashion-MNIST는 74.7%, 74.8%, 79.5%, 83.7%의 비율을 가졌다. 하위 1% 평균 뉴런 비율도 증가하는 경향을 보였다. CIFAR-10의 데이터 세트에서는 각 epoch에서 42.8%, 44.2%, 50.8%, 59.5%를 가졌고 CIFAR-100은 88.5%, 88.6%, 91.0%, 93.2% 가지며 Fashion-MNIST는 74.6%, 74.9%, 79.6%, 83.8%의 비율을 가졌다. 표 3은 각 데이터 세트마다 하위 1% 미만의 뉴런 비율과 epoch 간 상관관계 값을 보여준다.

표 3. 전체 모델 분석 하위 1% 뉴런 비율과 epoch 간 상관관계

데이터 세트명	상관관계	유의 확률	안정화 추세
CIFAR-10	0.940	0.000	높음
CIFAR-100	0.934	0.000	높음
Fashion-MNIST	0.943	0.000	높음

표 3은 학습이 진행되는 과정에 따라 CIFAR-10, CIFAR-100, Fashion-MNIST 모두 0.934 이상의 강한 상관 계수를 가지고 0.000의 유의 확률을 가졌다. ResNet에 대하여 수행한 전체 모델 분석 데이터 세트는 학습을 진행하면 테스트 데이터에 대한 하위 1% 미만 뉴런 비율이 증가하며 높은 안정화 추세를 가지고 있다. epoch이 증가함에 따라 데이터 세트 CIFAR-10, CIFAR-100, Fashion-MNIST 모두 출력 결과에 거의 영향을 주지 않는 하위 1% 뉴런 비율이 증가하며 epoch과 0.7이상의 상관 계수 값을 가지므로 높은 안정화 추세를 갖고 있다.

4.3.2 layer 구간 별 하위 1% 뉴런 비율과 epoch 간 관계 분석

CIFAR-10, CIFAR-100, Fashion-MNIST 데이터 세트의 학습이 진행되는 과정을 모델의 56 layer 중 1-14 layer, 15-28 layer, 29-42 layer, 43-55 layer 각각에 테스트 데이터 입력에 대한 하위 1% 뉴런 비율과 epoch 간 분석을 수행하였다. Layer을 구분 하였을 때 CIFAR-10과 Fashion-MNIST는 모델 전체 분석 결과와 큰 변화가 없었지만 CIFAR-100은 다른 결과를 보였다. 그림 5-8는 CIFAR-100을 layer 1-14, layer 15-28, layer 29-42, layer 43-55로 구분하여 epoch 변화에 따른 하위 1% 뉴런 비율 박스 플롯이다.

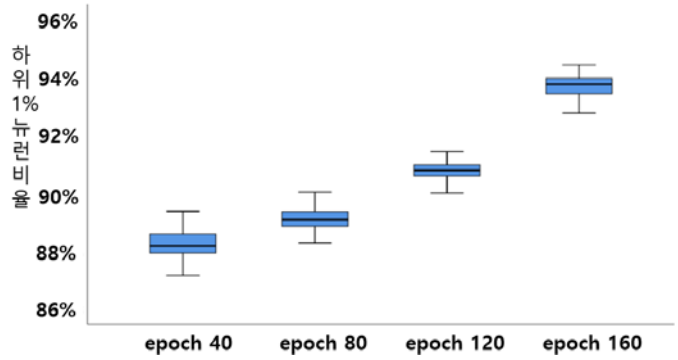


그림 5. CIFAR-100 layer 1-14 구간의 epoch 변화에 따른 하위 1% 뉴런 비율

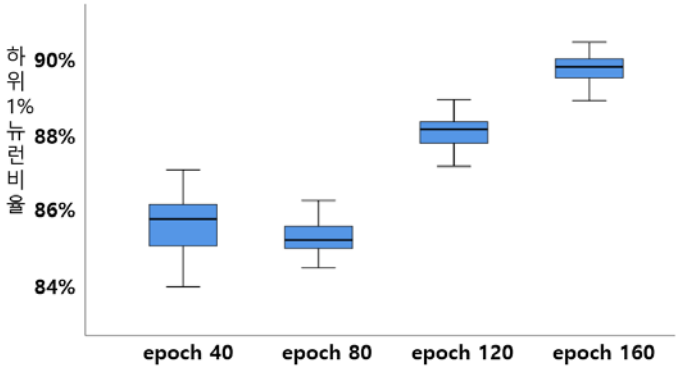


그림 6. CIFAR-100 layer 15-28 구간의 epoch 변화에 따른 하위 1% 뉴런 비율

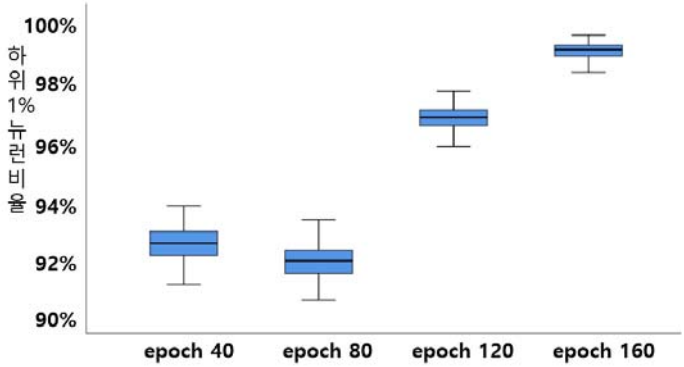


그림 7. CIFAR-100 layer 29-42 구간의 epoch 변화에 따른 하위 1% 뉴런 비율

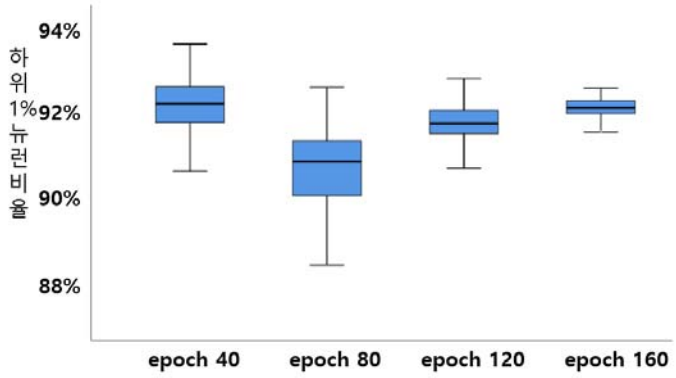


그림 8. CIFAR-100 layer 43-55 구간의 epoch 변화에 따른 하위 1% 뉴런 비율

그림 5는 1-14 layer 구간에서는 각 epoch이 증가함에 따라 하위 1% 뉴런 비율이 증가하는 경향을 보였고, 그림 6-7은 15-28 layer 구간과 29-42 layer 구간은 하위 1% 뉴런의 비율은 epoch이 증가함에 따라 하위 1% 뉴런의 비율이 epoch 40에서 epoch 80으로 갈 때 감소하고 다시 증가하는 경향을 보였다.

1-14 layer는 epoch 40, epoch 80, epoch 120, epoch 160 각각 88.1%, 89.0%, 90.7%, 93.7%의 하위 1% 뉴런 비율 중간 값을 가지며 epoch이 증가할 때 마다 중간 값도 증가하는 경향을 보였다. 15-28 layer는 epoch 40에서 85.6% 하위 1% 평균 뉴런 비율 중간 값을 가지며 epoch 80에서 85.1%로 감소하고 epoch 120에서는 88.0%, epoch 160에서는 89.7%의 하위 1% 평균 뉴런 비율 중간 값을 가졌다. 29-42 layer 구간에서는 epoch 40에서는 92.4%, epoch 80에서 91.8%으로 감소하였고 epoch 120에서는 96.7%, epoch 160에서 99.0%의 하위 1% 평균 뉴런 비율 중간 값을 가졌다. 그림 8은 43-55 layer 구간에서는 epoch이 증가함에 따라 92.1%, 90.7%, 91.6%, 92.0%의 하위 1% 평균 뉴런 비율 중간 값을 보이며 어떤 경향도 보이지 않았다. 표 4-6은 CIFAR-10, CIFAR-100, Fashion-MNIST에 대해 epoch 증가에 따른 하위 1% 평균 뉴런 비율을 layer 별로 구분하여 분석 한 것을 나타낸다.

표 4. CIFAR-10 layer 별 하위 1% 평균 뉴런 비율

Layer 구간	epoch 40	epoch 80	epoch 120	epoch 160
Layer 1-14	29.3%	33.2%	38.4%	50.1%
Layer 15-28	46.4%	43.7%	54.3%	59.7%
Layer 29-42	56.2%	57.9%	62.2%	71.0%
Layer 43-55	67.3%	71.1%	74.4%	79.3%

표 4는 CIFAR-10 데이터 세트로 만든 모델의 layer 별 하위 1% 평균 뉴런 비율이다. 하위 1% 평균 뉴런 비율이 layer 15-28구간에서 epoch 40에서 epoch 80으로 증가할 때를 제외하고는 모두 epoch이 증가함에 따라 하위 1% 평균 뉴런 비율이 증가하는 경향을 보였다.

표 5. CIFAR-100 layer 별 하위 1% 평균 뉴런 비율

Layer 구간	epoch 40	epoch 80	epoch 120	epoch 160
Layer 1-14	88.1%	89.0%	90.8%	93.6%
Layer 15-28	85.5%	85.2%	87.9%	89.7%
Layer 29-42	92.4%	91.8%	96.6%	98.9%
Layer 43-55	92.0%	90.5%	91.7%	92.0%

표 5는 CIFAR-100 데이터 세트로 만든 모델의 layer 별 하위 1% 평균 뉴런 비율이다. 하위 1% 평균 뉴런 비율이 layer 15-28 구간과 layer 29-42 구간, layer 43-55 구간에서 epoch 40에서 epoch 80으로 증가할 때를 제외하고는 모두 epoch이 증가함에 따라 하위 1% 평균 뉴런 비율이 증가하는 경향을 보였다.

표 6은 Fashion-MNIST 데이터 세트로 만든 모델의 layer 별 하위 1% 평균 뉴런 비율이다. 하위 1% 평균 뉴런 비율이 layer 1-14 구간과 layer 15-28, layer 29-42 구간에서 epoch 40에서 epoch 80으로 증가할 때를 제외하고는 모두 epoch이 증가함에 따라 하위 1% 평균 뉴런 비율이 증가하는 경향을 보였다.

표 6. Fashion-MNIST layer 별 하위 1% 평균 뉴런 비율

Layer 구간	epoch 40	epoch 80	epoch 120	epoch 160
Layer 1-14	68.0%	68.7%	74.9%	79.6%
Layer 15-28	73.8%	72.9%	76.8%	81.4%
Layer 29-42	84.9%	84.4%	89.7%	92.7%
Layer 43-55	87.2%	89.9%	90.9%	92.3%

표 7. Layer 별 하위 1% 뉴런 비율 및 epoch 간 상관관계

데이터 세트명		layer 1-14	Layer 15-28	layer 29-42	layer 43-55
CIFAR-10	상관관계	0.936	0.875	0.918	0.974
	유의확률	0.000	0.000	0.000	0.000
	안정화 추세	높음	높음	높음	높음
CIFAR-100	상관관계	0.951	0.893	0.909	**0.143**
	유의확률	0.000	0.000	0.000	**0.004**
	안정화 추세	높음	높음	높음	**낮음**
Fashion-MNIST	상관관계	0.955	0.874	0.885	0.901
	유의확률	0.000	0.000	0.000	0.000
	안정화 추세	높음	높음	높음	높음

표 7은 각 데이터 세트의 layer 별 하위 1% 뉴런 비율과 epoch 간 상관 계수와 유의 확률을 나타낸 것이다. CIFAR-10과 Fashion-MNIST에서는 전체 layer 구간에서 epoch이 증가함에 따라 하위 1% 평균 뉴런 비율이 증가하는 경향을 보였다. 그러나 CIFAR-100에서는 다른 경향을 보였다. CIFAR-10과 Fashion-MNIST는 테스트 데이터에 대하여 모든 layer 구간이 0.874 이상의 강한 상관관계를 보였고 0.000의 유의 확률을 가졌다. CIFAR-10 **0.9307**의 정확도를 가지고 Fashion-MNIST는 **0.9487**의 정확도를 가졌다.

CIFAR-100은 43-55 layer 구간 부분을 제외한 layer 구간에서는 모두 0.7이상의 강한 상관 계수를 보였고 0.000의 유의 확률을 가졌다. layer 43-55의 구간에서는 0.143의 낮은 상관 계수 값을 가지고 0.004의 유의 확률을 가졌고 CIFAR-100은 **0.7058**의 정확도를 가졌다.

CIFAR-10과 Fashion-MNIST는 모든 layer 구간에서 0.7이상의 상관관계 계수를 가져 모든 구간이 높은 안정화 추세를 보였다. CIFAR-100은 Layer 1-14, layer 15-28, layer 29-42 구간 모두 0.7 이상의 상관 계수를 가져 높은 안정화 추세를 보였다. 그러나 CIFAR-100의 layer 43-55 구간은 0.143 낮은 상관 계수를 가져 낮은 안정화 추세를 보였다.

5. 결론 및 향후 연구

본 연구에서는 ResNet의 뉴런 활성 값과 epoch 간 상관관계 분석을 통하여 모델 내부 뉴런의 조합이 어떤 관계로 출력이 결정되는지를 고려한 화이트 박스 모델 평가 방법을 제안하였다. 본 연구에서는 ResNet을 이용하여 CIFAR-10, CIFAR-100, Fashion-MNIST 데이터 세트에 대해 히스토그램을 이용하여 모델이 학습을 진행함에 따라 작은 활성 값을 가지는 뉴런의 수가 증가함을 확인하였다. 본 논문은 작은 활성 값을 가지는 하위 1% 뉴런 비율을 이용하여 epoch 간 상관관계 분석을 통한 모델의 안정화 추세 평가 방법을 제안하였다. 모델의 안정화 추세 평가는 전체 모델 분석과 layer 별로 구분하여 평가를 수행하였고 하위 1% 평균 뉴런 비율을 이용하여 epoch 간 상관관계 분석을 통하여 epoch에 따른 모델 내부의 뉴런 활성 값 변화를 분석하였다.

전체 모델 분석 시 데이터 세트 CIFAR-10, CIFAR-100, Fashion-MNIST 모두 하위 1%의 뉴런 비율이 epoch이 증가함에 따라 비율이 증가하는 높은 상관관계를 보였다. 그러나 layer 별 모델 분석은 **0.9307** 이상의 높은 정확도를 가지는 CIFAR-10과 Fashion-MNIST는 모델 전체를 보았을 때와 같은 경향인 높은 안정화 추세를 보였으나 **0.7058**의 상대적으로 낮은 정확도를 가지는 CIFAR-100에서는 43-55 layer의 구간에서 낮은 안정화 추세를 보이는 상이한 결과를 보였다.

CIFAR-10과 Fashion-MNIST는 테스트 데이터에 대하여 하위 1% 뉴런 비율이 모든 layer 구간이 강한 상관관계를 보여 높은 안정화 추세를 가지고 있다고 평가되었다. CIFAR-100의 43-55 layer 구간 부분을 제외한 layer 구간에서도 모델 전체 분석과 같이 높은 안정화 추세를 보였지만 layer 43-55의 구간에서는 낮은 상관 계수 값을 가지며 모델이 낮은 안정화 추세를 보였다.

앞으로 본 논문에서 제안한 뉴런 분포 비율이 ResNet 모델에서 STL-10, ImageNet과 같이 다양한 데이터 세트와 상관관계가 낮은 데이터 세트 모델에서 정확도가 낮을 때의 상관관계 분석에 대해서도 사례 연구를 추가 수행하여 작은 활성 값을 갖는 뉴런과 정확도와의 관계를 연구할 예정이다.

Acknowledgement

이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No.2019R1F1A1063336).

참고 문헌

[1] Krizhevsky Alex et al, “IMAGENET classification with deep convolutional neural networks,” Proceedings *of the Advances in Neural Information Processing Systems*, pp. 1097-1105, 2012.

[2] Kaiming He et al, “Deep Residual Learning for Image Recognition,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016.

[3] Rezende, Edmar, et al, "Malicious software classification using transfer learning of resnet-50 deep neural network." *2017 16th IEEE International Conference on Machine Learning and Applications*. IEEE, 2017.

[4] Yu Weize, et al, "ResNet-based Trojan detection methodology for protected ICs." *Electronics Letters* Vol.55, No.21, pp 1116-1118, 2019.

[5] Heechul Jung, et al, "ResNet-based vehicle classification and localization in traffic surveillance systems." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017.

[6] Kexin Pei, et al, "Deepxplore: Automated whitebox testing of deep learning systems," *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017.

[7] Youcheng Sun, et al, "Testing deep neural networks." *arXiv preprint arXiv:1803.04792,* 2018.

[8] Lei Ma, et al, "Deepgauge: Multi-granularity testing criteria for deep learning systems." *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. ACM, 2018.

[9] Krizhevsky Alex and Hinton Geoff, “Convolutional deep belief networks on CIFAR-10,” *Proceedings of the Unpublished manuscript*, Vol. 40, No. 7, 2010.

[10] Wang Fei, et al, “Residual attention network for image classification,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156-3164, 2017.

[11] Xiao Han et al, “Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms,” *Proceedings of the arXiv preprint arXiv.17.8.07747*, 2017.

[12] J. B. Stroud. Fundamental statistics in psychology and education, Journal of Educational Psychology, pp. 42-318, 1951.

소프트웨어 위험원 분석을 위한 상황 분류 기반의 조합을 통한 STPA 문맥표 최적화

양현수 O, 권기현
경기대학교 컴퓨터과학과
{sean1538, khkwon}@kyonggi.ac.kr

Optimizing STPA Context Table using situation classification Based combination for Software Hazard Analysis

Hyunsoo YangO , Gihwon Kwon
Department of Computer Science, Kyonggi University

요 약

최근 안전 필수 시스템의 복잡도가 증가하며 이를 제어하기 위한 소프트웨어의 비중 또한 높아지고 있다. 안전 필수 시스템에서 제어의 중심이 되는 소프트웨어는 제어 알고리즘에 따라 적절한 제어 명령을 생성하고, 제어 명령으로 발생한 피드백을 다시 입력 받는 상호작용을 반복한다. 이로 인해 소프트웨어를 적절하게 제어하지 못하여 발생하는 안전 사고가 많아지게 되며, 시스템 전체의 안전성을 확보하기 위한 소프트웨어 위험원 분석이 요구된다. STPA(System Theoretic Process Analysis)는 구성요소의 상호작용을 중심으로 복잡한 시스템의 위험원을 식별하는 위험원 분석 기법이다. STPA에서는 제어 명령을 생성하기 위한 판단 근거가 되는 제어 알고리즘을 표현한 프로세스 모델을 기반으로 문맥표를 작성한다. 문맥표는 프로세스 모델이 갖는 모든 정보들을 조합하여 소프트웨어가 생성할 수 있는 모든 제어 명령들을 생성하여 부적절한 제어 알고리즘을 시나리오 형태로 식별한다. 안전 필수 시스템의 복잡도가 증가함에 따라 프로세스 모델의 정보들이 증가하게 되고, 작성되는 문맥표의 크기 또한 증가한다. 이번 연구에서는 시스템의 복잡도에 따라 증가하는 문맥표의 크기를 최적화하여 위험원 분석의 비용과 노력 측면에서 효율성을 증가시키는 것이 목표이다. 이를 위하여 상황 분류 기반의 문맥표 조합 방법을 제안하고, 안전 필수 시스템의 축소인 디오라마 열차 시스템의 소프트웨어 위험원 분석에 적용하여 제안하는 방법을 확인한다.

1. 서 론

최근 안전 필수 시스템의 복잡도가 증가하며 이를 제어하기 위한 소프트웨어의 비중 또한 높아지고 있다[1]. 소프트웨어는 안전 필수 시스템을 제어하기 위해 제어 알고리즘에 따라 적절한 제어 명령을 생성하는 역할을 한다. 소프트웨어에 의해 생성된 제어 명령은 시스템에 적용되어 피드백을 발생시키고, 이를 다시 입력 받는 상호 작용을 반복한다[2]. 이로 인해 소프트웨어의 제어알고리즘을 부적절하게 설계하여 발생하는 안전 사고가 많아지게 되며, 시스템 전체의 안전성을 확보하기 위한 소프트웨어 위험원 분석이 요구된다[1].

STPA(System Theoretic Process Analysis)는 구성요소의 상호작용을 중심으로 시스템 수준에서 부적절한 제어 명령을 식별하는 위험원 분석 기법이다[2]. 시스템에서 발생하는 모든 제어 명령을 식별 후 해당 제어 명령이 부적절하게 제어되는 경우를 식별하고, 이러한 부적절한 제어 명령들이 발생하는 원인들을 사고 시나리오 형태로 식별한다[3,4]. STPA는 부적절한 제어 명령들이 발생하는 원인 중 하나로, 제어 알고리즘의 오류를 분석하여 소프트웨어 위험원을 식별한다[4]. STPA에서 제어 알고리즘은 제어 명령을 내리는 판단의 근거가 되는 다양한 정보를 가진 프로세스 모델로 표현된다. 프로세스 모델을 기반으로 시스템을 제어하기 위한 정보들을 모두 조합하여 문맥표를 작성한다. 작성된 문맥표는 시스템에서 소프트웨어에 의하여 발생할 수 있는 모든 제어 명령을 식별한다[5-8].

안전 필수 시스템의 복잡도가 증가하면, 이를 제어하기 위한 제어 알고리즘의 복잡도 또한 증가한다. 이에 따라 제어 알고리즘을 표현하기 위한 프로세스 모델이 갖는 정보들의 양이 증가한다. 결과적으로 소프트웨어 위험원 분석을 위해 작성되는 문맥표의 크기가 프로세스 모델의 정보의 양만큼 기하급수적으로 증가하게 된다. 이번 연구에서는 안전 필수 시스템의 복잡도에 따라 증가하는 문맥표의 크기를 최적화하여 위험원 분석의 비용과 노력 측면에서 효율성을 증가시키는 것이 목표이다. 이를 위하여 상황 분류 기반의 문맥표 조합 방법을 제안하고, 안전 필수 시스템의 축소인 디오라마 열차 시스템의 소프트웨어 위험원 분석에 적용하여 제안하는 방법을 확인한다.

2. 배경 지식

2.1 프로세스 모델(Process Model)

STPA는 분석 범위를 세부적으로 설정한 후 그림 1과 같은 제어 루프를 작성하여 제어 명령이 부적절하게 발생하는 경우를 식별해낸다. 제어 명령을 내리는 주체인 제어기에서 제어 명령을 전달하고 제어 대상이 제어된 후 감지기를 통하여 발생하는 피드백을 받아들인다[3,4]. 이러한 제어 루프에서 제어기는 시스템을 제어하기 위한 제어 알고리즘과 프로세스 모델을 갖는다. 제어 알고리즘은 제어 명령을 내리기 위한 제어 상의 의사 결정 프로세스이다. 프로세스 모델은 이러한 의사 결정 프로세스를 판단하기 위한 근거가 되는 다양한 정보를 가지고 있다[9].

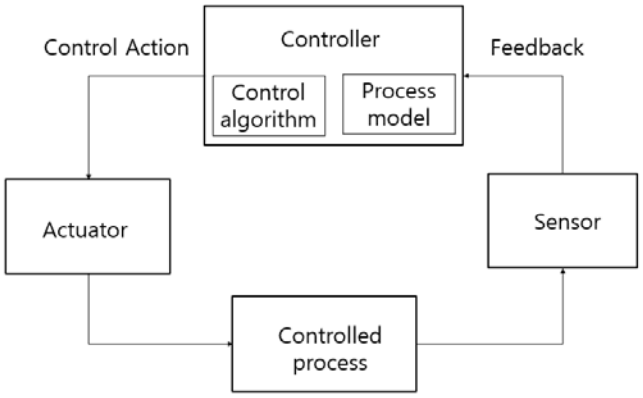


그림 1. STPA에서의 일반적인 제어 루프

앞서 설명한 프로세스 모델이 갖고 있는 의사 결정 프로세스를 판단하기 위해 근거가 되는 다양한 정보는 그림 2와 같은 관점에서 시작하여 구성된다. 시스템을 제어하기 위한 제어기의 소프트웨어는 현재 시스템의 상태 변수들을 조합하여 적절한 제어 변수를 제공한다[10]. 이를 위하여 다양한 입력 변수들을 입력받아 알고리즘 변수들과 조합하여 시스템 제어에 적절한 출력 변수를 제공한다[8].

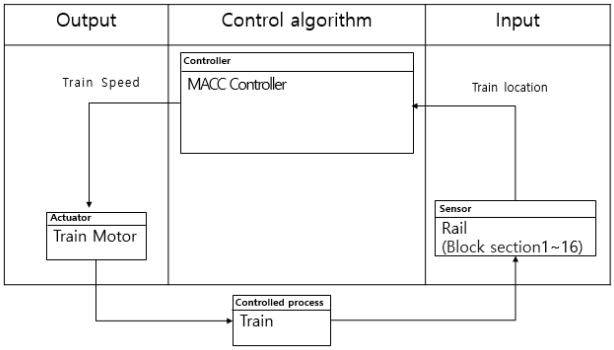


그림 2. 프로세스 모델 식별 관점을 표현한 제어 루프(MACC 시스템)

그림2는 열차 모형 디오라마로 구현된 MACC (Multiple Adaptive Cruise Control)시스템의 제어 루프이다. MACC 시스템은 선행 열차의 속도에 맞춰 후행 열차가 설정된 열차 간의 간격(폐색 구간)을 유지하며 운행이 가능한 시스템이다. MACC 시스템에서 요구하는 프로세스 변수들을 표1과 같이 식별한다[8].

표 1. 프로세스 변수 식별 표(MACC 시스템)

	Input	Control algorithm	output
State variable	Train location	Block section between trains	
Control variable			Train speed

시스템 운영에 필요한 프로세스 변수들의 식별한 후에는 각각의 프로세스 변수들이 가져야할 모든 프로세스 값들을 식별한다. MACC(Multiple Adaptive Cruise Control)[6] 시스템에서 MACC 제어기는 열차위치, 열차 속도, 폐색 구간 계산 프로세스 모델을 가진다. 열차 위치 프로세스 모델은 선로로부터 열차 위치를 “1 ~ 16(총 16개의 폐색 구간)” 프로세스 값을 입력 프로세스 값으로 가진다. 이때 열차 위치 값으로 정의하지 않은 프로세스 값이 입력이 되었을 때는 안전장치(Fail-safe)를 작동시켜 모든 열차를 정지시켜 사고를 회피하기 위한 “알 수 없음” 값을 입력 프로세스 값으로 갖도록 한다. 폐색 구간 계산 프로세스 모델은 열차 위치 프로세스 모델에 의해 파악된 각각의 열차 위치를 기반으로 선행 열차와 후행 열차의 거리가 “1(폐색구간 1)”로 유지할 수 있도록 제어한다. 열차 속도 프로세스 모델은 폐색 구간 프로세스 모델에 의해 계산된 선행 열차와 후행 열차 사이의 거리를 기반으로 각각의 열차에 “가속”, “감속”, “정지” 프로세스 값을 출력 프로세스 값으로 가진다. 식별된 모든 정보들을 프로세스 모델로 표현한 제어구조도는 그림3과 같다[8].

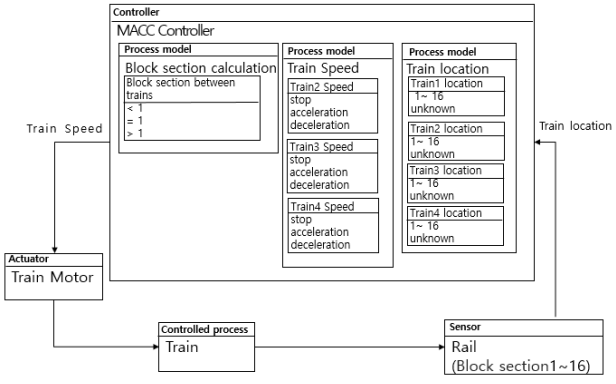


그림 3. 프로세스 모델이 표현된 제어 루프 예시(MACC 시스템)

표 2. STPA에서의 일반적인 문맥표

Control Action	Controlled process	Control variable	State variable	State variable	Software state	Scenario list
	Control Action commands to controlled process by Control variable		Combine state variables to create situations centered on the controlled process		Safe Normal Hazardous	SS 1

표 3. 문맥표 예시(MACC 시스템)

Control Action	Controlled process	Control variable	State variable	State variable	Software state	Scenario list
	Following Train	Train speed	Block section between Trains	Train speed (Leading Train)		
CA 3.2	Following train stop command	acceleration	< 1 = 1 > 1	deceleration acceleration	Hazardous	SS 2
				stop	Safe	-

2.2 문맥표(Context table)

문맥표는 프로세스 모델을 작성하여 식별된 의사결정 프로세스를 판단하기 위해 근거가 되는 다양한 정보들을 조합하여 모든 상황을 생성하도록 표 2와 같이 구성되어 있다[8,10]. 문맥표 작성의 목적은 생성되는 모든 제어 명령들을 분석하는 것이다. 표3은 프로세스 값들을 조합하여 생성하는 모든 상황 중 일부이다. 제어 대상인 후행 열차의 제어 프로세스 변수인 열차 속도의 프로세스 값이 적용이 되었을 때, 상황을 생성해내는 상태 변수들의 프로세스 값들을 조합하여 상황을 생성한다. 제어 대상인 후행 열차를 기준으로 후행 열차와의 폐색 구간 상태 변수와 선생 열차의 속도 변수가 가질 수 있는 프로세스 값들을 조합하여 상황을 생성한다. 작성된 문맥표에 따라 일부 상황에서는 위험한 상황이 발생하는 것을 확인할 수 있다. 발생하는 상황여부에 따라 소프트웨어 상태를 표기하고 소프트웨어 사고시나리오 “SS2 SS 2. 선행 열차가 감속한 상태에서 후행 열차가 가속하지 않으면 열차가 추돌한다.” 를 식별해낼 수 있다. 식별된 소프트웨어 사고 시나리오는 소프트웨어 안전 요구사항 “SR 2. 선행 열차가 감속하면 후행 열차는 반드시 감속하거나 정지해야 한다.”로 도출된다.

3. 기존 문맥표의 한계점

문맥표는 프로세스 모델이 필요한 모든 정보를 가지는 프로세스 변수의 조합을 표현한 표로써 제어기로부터 발생할 수 있는 모든 상황을 생성할 수 있도록 작성된다. 모든 상황을 생성하여 분석을 한다는 점은 사고 시나리오 식별에 있어서 큰 장점으로 작용한다. 안전 시스템의 복잡도가 증가할수록 이를 제어하기 위한 소프트웨어의 복잡도 또한 증가하게 된다. 따라서 앞서 말한 장점이 복잡한 안전 필수 시스템에서는 분석 비용과 노력의 측면에서 단점으로 작용할 수 있다. 소프트웨어의 복잡도는 그림 4의 제어기의 계층적 구조에 따라 구분하여 정의할 수 있다[4,11,12].

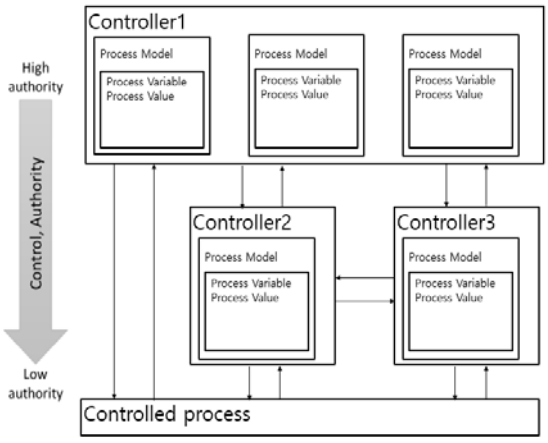


그림 4. 계층적 제어 구조도

여러 제어기의 소프트웨어가 상호작용을 하는 경우에는 제어기의 계층적 구조에 따라 두 가지 상황이 발생할 수 있다. 첫 번째로는 그림 5의 제어기2(Controller2)와 제어기3(Controller3)과 같은 동일 계층의 관계일 때이다. 동일 계층의 제어기는 서로 동일한 권한으로 자유롭게 상호작용을 하며 제어 명령을 생성한다. 따라서 작성되는 문맥표의 크기가 변수의 수만큼 조합되며 그림 5와 같이 문맥표의 크기가 기하급수적으로 증가한다.

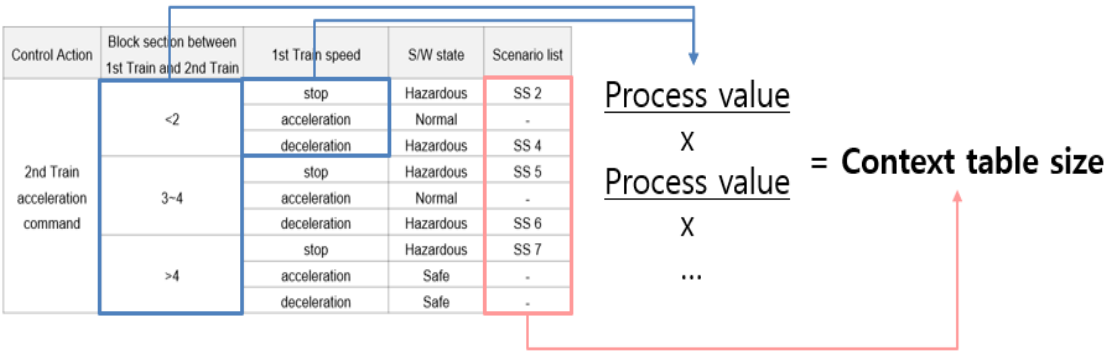


그림 5. 동일 계층 제어기에서 작성되는 문맥표 크기 증가

두 번째로는 그림 4의 제어기1(Controller1)과 제어기2(Controller2)와 같은 서로 다른 계층의 관계일 때이다. 이러한 구조에서는 상위 계층의 제어기가 하위 계층의 제어기를 제어하기 위한 전제조건을 생성한다. 이러한 구조에서는 하나의 문맥표로 모든 상황을 생성하는데 제약사항이 발생한다. 이때 발생하는 제약사항은 상위 계층에서 생성된 전제 조건에 따라 하위 계층의 제어기의 제어 명령 생성이 서로 다를 수 있기 때문이다.

4. 문맥표 조합

문맥표의 최종 크기를 최적화하며 모든 상황을 생성해내기 위해 상황 분석을 기반으로 문맥표를 조합하는 방법을 제안한다. 문맥표 조합의 컨셉은 그림 6과 같다. 프로세스 모델의 분석 결과에 따라 상황을 분류하여 기본 기능(Foundation function) 문맥표를 작성한 후 조합해 나가며 문맥표의 전체 크기를 최적화한다. 문맥표 조합의 기반이 되는 상황 분류 결과를 상황 분류(situation classification) 문맥표로 작성하고 기본 기능 문맥표 또는 확장 기능(Extended function) 문맥표를 조합한다. 각각의 문맥표 작성 시에는 규칙 기반 (Rule-based) 접근법[13]을 적용하여 문맥표 자체의 크기를 최적화하도록 한다.

문맥표 조합을 하기 위해서는 프로세스 모델을 분석하는 과정이 중요하다. 프로세스 모델 분석은 그림 6의 문맥표 조합 컨셉을 적용하기 위한 설계 단계이다. 분석 시스템에서 제어기들의 구조가 계층적인지 동일계층인지를 파악하고 문맥표 조합이 가능한지를 판단해야 한다. 프로세스 모델 분석을 실시한 후에는 시스템을 제어하기 위한 최하위 계층 제어기의 기본 기능 문맥표를 작성한다. 이후 상위 계층 제어기에 의한 전제 조건 발생 여부를 상황 분류 문맥표를 통해 식별한다. 전제 조건 발생 여부가 식별되었을 경우 전제 조건에 따라 분류된 확장된 기능을 분석하기 위한 확장 기능 문맥표를 작성한다. 최초 기본 기능 문맥표를 작성한 이후 에는 작성한 문맥표를 조합할 수 있는지를 상황 분류를 기반으로 판단하여 문맥표를 조합하고 최종 문맥표의 크기를 최적화한다. 이러한 과정은 시스템을 제어하기 위한 모든 기능들의 문맥표가 완성될 때까지 그림 7과 같이 반복적으로 수행한다.

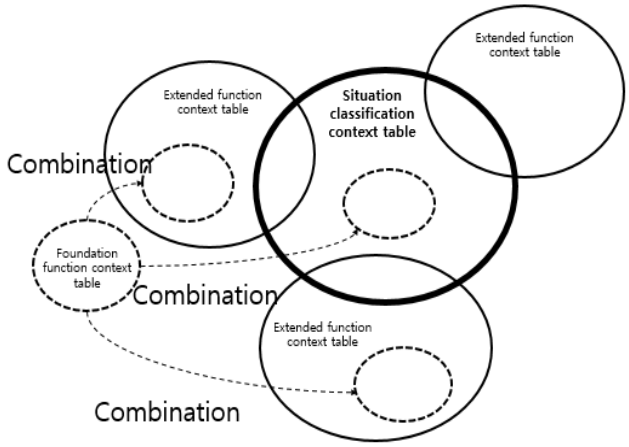


그림 6. 문맥표 조합 컨셉

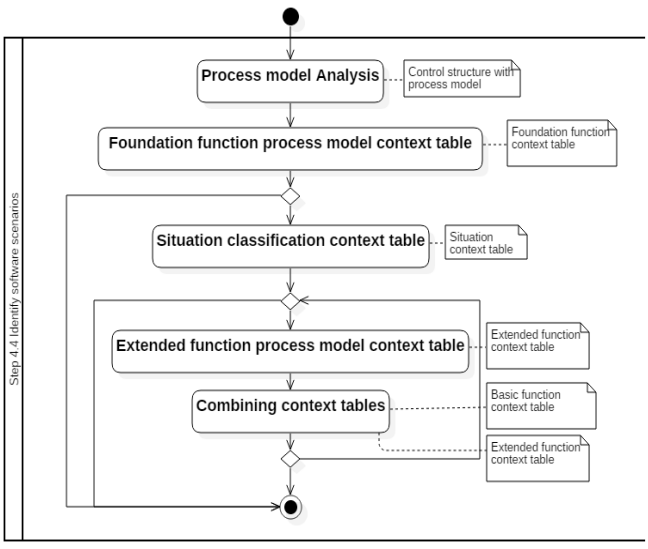


그림 7. 문맥표 조합 절차

5. 적용 사례

5.1 선로전환기가 추가된 MACC 시스템

제안 내용을 확인해볼 시스템은 Asim의 ACC (Adaptive Cruise Control)[6]를 응용하여 열차에 적용한 MACC(Multiple Adaptive Cruise Control)[8]이다. MACC는 4대의 열차가 서로 설정된 폐색 구간을 유지하며 운행하고 이를 제어할 수 있는 철도 시스템이다. 열차 추돌 상황이 발생 가능한 폐색 구간 거리가 1미만 이 될 시에는 열차 추돌 상황이 발생 가능하므로 안전장치(Fail - Safe)를 적용하여 모든 열차를 정지하게 한다. 이전 연구에서 STPA 분석기법으로 분석이 완료된 MACC 시스템을 확장하여 이번 연구의 적용 사례로 활용한다. 이는 시스템에서 제어의 중심이 되는 소프트웨어의 복잡도를 증가시키기 위함이다. 선로전환기가 추가된 MACC 시스템의 구성은 그림8과 같다.

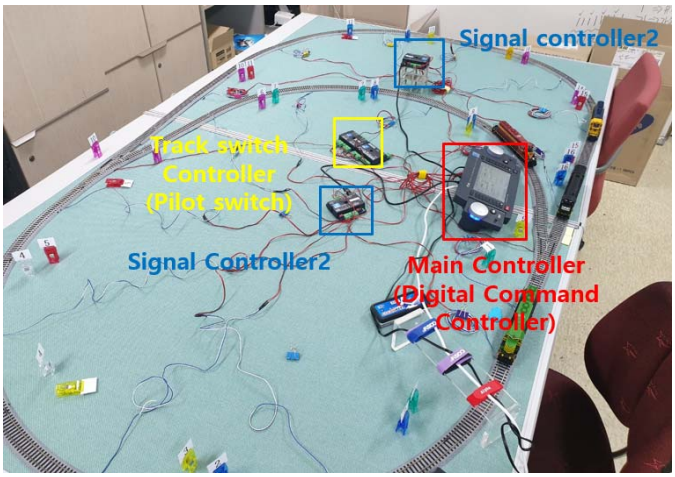


그림 8. 선로전환기가 추가된 MACC의 시스템 구성

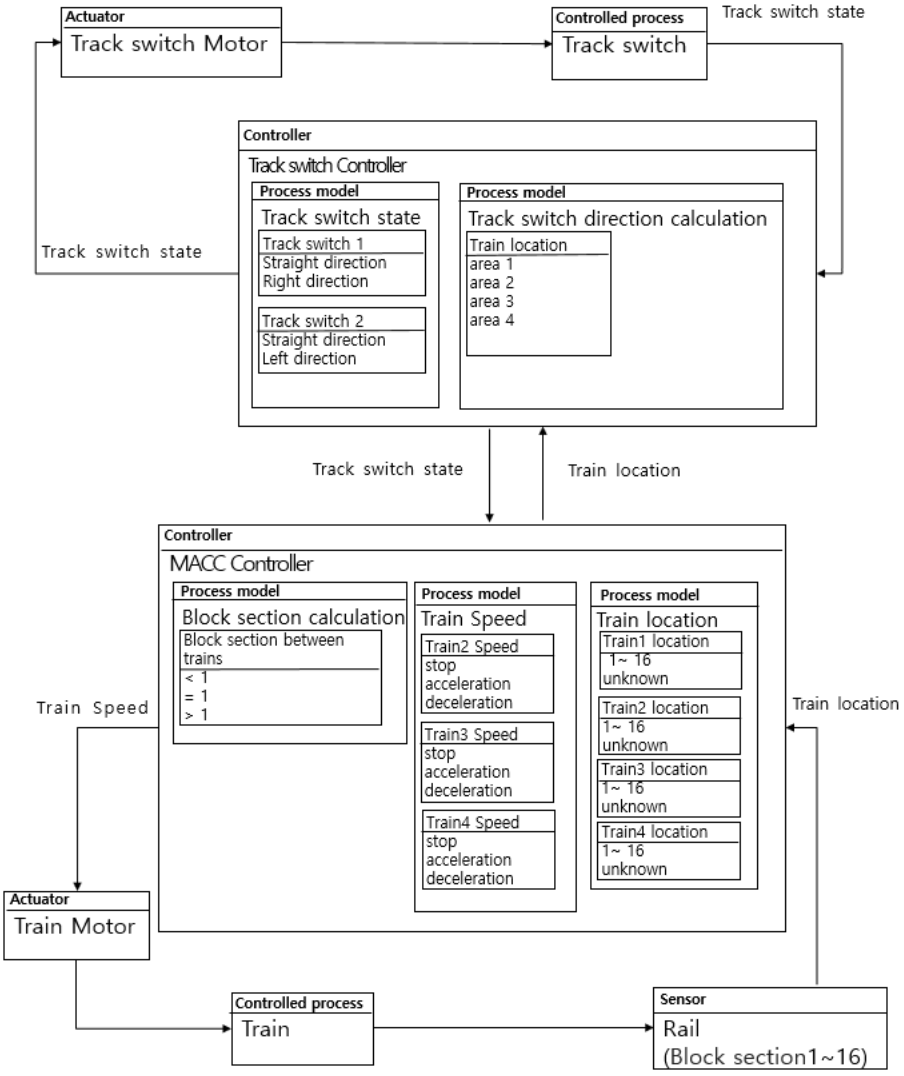


그림 9. 선로전환기가 추가된 MACC 시스템의 프로세스 모델이 표현된 제어구조도

5.2 STPA 적용

제안하는 문맥표 조합 방법을 적용하기 위하여 선로전환기가 추가된 MACC 시스템의 프로세스 모델을 식별한다. MACC 제어기(MACC Controller)는 열차위치, 열차 속도, 폐색 구간 계산 프로세스 모델을 가진다. 열차 위치 프로세스 모델은 선로로부터 열차 위치를 “1 ~ 16(총 16개의 폐색 구간)” 프로세스 값을 입력 프로세스 값으로 가진다. 이때 열차 위치 값으로 정의하지 않은 프로세스 값이 입력이 되었을 때는 안전장치(Fail-safe)를 작동시켜 모든 열차를 정지시켜 사고를 회피하기 위한 “알 수 없음”값을 입력 프로세스 값으로 갖도록 한다. 폐색 구간 계산 프로세스 모델은 열차 위치 프로세스 모델에 의해 파악된 각각의 열차 위치를 기반으로 선행 열차와 후행 열차의 거리가 “1(폐색구간 1)”로 유지할 수 있도록 제어한다. 열차 속도 프로세스 모델은 폐색 구간 프로세스 모델에 의해 계산된 선행 열차와 후행 열차 사이의 거리를 기반으로 각각의 열차에 “가속”, “감속”, “정지” 프로세스 값을 출력 프로세스 값으로 가진다.

선로 전환기 제어기는 MACC 제어기의 열차 위치 프로세스 모델에 의해 정리된 열차 위치를 입력 프로세스 값으로 받는다. 입력 프로세스 값을 바탕으로 선로전환기 방향을 정할 수 있도록 선로 전환기 방향 프로세스 모델을 가진다. 선로 전환기는 총 2대 운영되며 선로 전환기는 각각의 선로전환기에서 선로 전환기 방향을 계산하기 위해 열차 위치를 입력 프로세스 값으로 가진다. 이때 열차 위치 프로세스 값은 MACC 제어기에서의 프로세스 값을 기본으로 재정의하여 “지역 1”, 지역 2”, 지역 3”, 지역 4”을 제어 알고리즘 프로세스 값으로 가진다. 선로 전환기 방향 계산 프로세스 모델에 의해 각각의 선로전환기에 제어 명령을 전달하기 위해 전환기 1은 “직선 방향”, “오른쪽 방향” 그리고 선로전환기 2는 “직선 방향”, “왼쪽 방향”을 출력 프로세스 값으로 가진다. 식별된 프로세스 모델, 프로세스 변수, 프로세스 값을 모델링하여 제어 구조도에 표현하면 그림 9와 같다.

5.3 문맥표 조합 적용

이번 적용 사례에서의 분석 범위는 선로 전환기2 주변에서 최대 4대의 열차가 운행될 때의 소프트웨어 위험원을 분석하는 것으로 설정한다. 그림 10과 같이 총 16개로 나누어진 폐색 구간 중 지역1, 지역 2, 지역3, 지역 4에 열차가 총 4대까지 위치할 때의 위험원 분석을 실시한다. 열차가 추돌하지 않도록 선로 전환기2 그리고 선행열차와 후행열차의 간격을 유지시켜 주는 MACC 기능 정상적으로 동작하도록 소프트웨어를 제어하는 것이 목표이다.

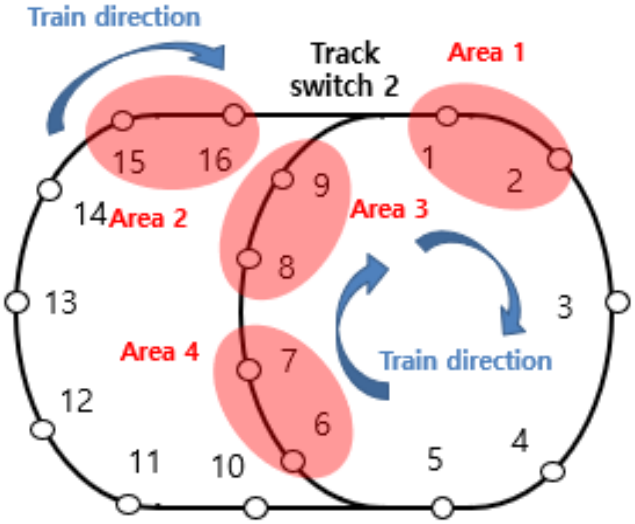


그림 10. 적용 사례의 분석 범위

제안하는 상황 분류 기반의 문맥표 조합 절차에 따라 선로전환기가 추가된 MACC의 소프트웨어 위험원 분석을 실시한다. 분석 시스템에서 제어기들의 구조가 계층적인지 동일 계층인지를 파악하고 문맥표 조합 가능 여부를 판단해야 한다. 이를 위하여 시스템 전체의 프로세스 모델을 분석한다. MACC가 정상적으로 동작하기 위해서는 선로 전환기 제어기에 의하여 선로 전환기가 제어가 된 후에, MACC가 동작해야 기능 요구사항들을 만족시킬 수 있다. 따라서 선로전환기는 상위 계층이고 MACC 제어기가 하위 계층인 계층적 구조로 모델링 한다. 이러한 계층적 구조에 의하여 선로 전환기의 적절한 제어를 분석하기 위한 문맥표를 전제 조건으로 분석이 진행되어야 한다. 이에 따라 선로 전환기의 문맥표가 MACC 시스템 전체의 분석 상황을 분류하는 상황 분류 문맥표로 작성된다. 상황 분류에 따라 가장 하위 계층인 기본 기능 MACC가 기본 기능 문맥표가 되며 상황 분류에 따라 확장 기능 문맥표를 작성하거나 기본 기능 문맥표를 재활용하여 조합이 가능할 것을 예상할 수 있다. 또한 프로세스 모델 분석 결과로 프로세스 변수 중 “열차 사이 폐색 구간” 보다는 선행 열차와 후행 열차의 “열차 속도” 프로세스 변수가 소프트웨어 사고 시나리오에 더 큰 영향을 준다는 것을 확인할 수 있다. 왜냐하면 선로전환기가 추가된 MACC 에서는 “열차 사이의 폐색 구간”이 이전 연구에서 와는 다르게 1로 줄었기 때문이다.

위와 같은 프로세스 모델 분석 결과를 바탕으로 규칙 기반 접근법을 적용하여 이후 작성될 문맥표들의 크기를 최적화할 규칙들을 정의한다. 규칙 정의는 표 4와 같이 정의하여 상태 변수로써 상황을 조합하는데 있어서 영향을 미치지 않는 프로세스 값들을 조합하여 이후 식별될 문맥표의 크기를 최적화하도록 한다.

표 4 선로전환기가 추가된 MACC 문맥표의 규칙 정의

Defined rule	Process variable
	Block section between trains
ANY	<1 = 1 >1

Defined rule	Process variable
	Train location
area 1	1 or 2
area 2	15 or 16
area 3	8 or 9
area 4	6 or 7

Defined rule	Process variable
	Train speed
Moving	acceleration deceleration

표 5 MACC 문맥표(기본 기능 문맥표)

Control Action	Controlled process	Control variable	State variable	State variable	Software state	Scenario list
	Trailing Train	Train speed	Block section between Trains	Train speed (Leading Train)		
CA 3.1.	Train2 stop command	stop	ANY	ANY	Safe	-
CA 3.2.	Train2 acceleration command	acceleration	ANY	acceleration	Safe	-
				deceleration	Hazardous	SS 2
CA 3.3.	Train2 deceleration command	deceleration	ANY	moving	Safe	-
CA 3.2 or CA 3.3	Train2 acceleration or deceleration command	moving	ANY	stop	Hazardous	SS 3

상황 분석에 따라 조합이 되는 하위 계층 제어기의 MACC 문맥표(기본 기능 문맥표)를 표 5와 같이 작성한다. 기본 기능 문맥표에서는 제어 대상의 제어 프로세스 값 3개로 생성되는 제어 명령에 대해서 상황을 조합하는 상태 프로세스 값이 3개씩 조합되어 총 크기 값이 27인 문맥표가 작성되어야 한다. 제안하는 문맥표 조합 방법에서는 앞서 설명한 규칙 기반 접근법을 적용하여 문맥표의 크기를 5까지 최적화할 수 있다. MACC 문맥표에서는 다음과 같은 2개의 사고 시나리오가 식별된다. “SS2: 선행 열차가 정지가 아닌(움직이고 있는) 상태에서 후행 열차가 가속하지 않아 폐색 구간 거리가 1 미만이 되어 열차가 추돌한다.”, “SS2: 선행 열차가 정지한 상태에서 후행 열차가 정지하지 않으면 폐색 구간 거리가 1 미만이 되어 열차가 추돌한다.”

다음 단계로 기본 기능 문맥표 또는 확장 기능 문맥표를 조합하기 위한 상황 분류 문맥표를 작성한다. MACC 문맥표(기본 기능 문맥표)를 조합하기 위해서는 상위 제어기인 선로전환기 제어기에 의해 선로전환기가 정상적으로 작동하는 것을 전제조건으로 하여 선행 열차와 후행 열차의 MACC 기능만을 고려하는 상황들을 분류해야 한다. 이를 위하여 표 6과 같이 선로 전환기 문맥표(상황 분류 문맥표)를 작성한다. 열차 위치 상태 변수들의 조합에 의하여 선로전환기 주위(지역1 ~ 지역4)에 위치한 열차들의 위치를 확인할 수 있다. 열차가 선행열차와 후행열차의 2대 사이 MACC기능만을 고려하면 되는 상황에서는 기본기능 MACC 문맥표가 조합된다. 선로전환기에 의해 3대 또는 4대의 열차가 위치하는 상황을 분석하기 위해서는 확장 기능 문맥표를 조합하여 문맥표의 크기를 최적화한다.

선로 전환기 문맥표에서는 다음과 같은 3개의 사고 시나리오가 식별된다. “SS 4: 선로 전환기가 직진 방향일 때 지역 3에서부터 열차가 접근하면 열차가 탈선한다.”, “SS 5: 선로 전환기가 왼쪽 방향일 때 지역 2에서부터 열차가 접근하면 열차가 탈선한다.”, “SS 6: 열차 위치 지역 1, 지역 2, 지역 3에서 동시에 열차 정보가 파악되면 열차가 탈선하거나 추돌한다.”

표 6. 선로 전환기 문맥표(상황 분류 문맥표)

Control Action	Controlled process	Control variable	State variable	Software state	Context table combination	Scenario list
	Track switch2	Track switch2	Train location			
CA 4.3.	Switch straight	straight direction	area1 & area3	Hazardous	-	SS 4
			area1 & area2	Safe	MACC context table (table 5)	-
			area1 & area2 & area3	Hazardous	3 trains MACC context table1 (table 7)	SS 6
			area1 & area2 & area3 & area4	Hazardous	4 trains MACC context table2 (table 8)	SS 6
CA 3.4.	Switch left	left direction	area1 & area3	Safe	MACC context table (table 5)	-
			area1 & area2	Hazardous	-	SS 5
			area1 & area2 & area3	Hazardous	3 trains MACC context table2	SS 6
			area1 & area2 & area3 & area4	Hazardous	4 trains MACC context table2	SS 6

표 7. 열차 3대 MACC 문맥표 1(확장 기능 문맥표)

Control Action	Controlled process	Precondition State variable	State variable	State variable	Software state	Context table combination	Scenario list
	Train2 (area 3) Track switch2	Track switch2	Train1 (area 1) speed	Train3 (area 2) Speed			
CA 3.1. CA 4.3.	Train2 stop command Switch straight	straight direction	ANY	stop	Normal	-	SS 7
				moving	Safe	MACC context table (table 5)	-
CA 3.2. CA 4.3.	Train2 acceleration command Switch straight		ANY	ANY	Hazardous	-	SS 4
CA 3.3. CA 4.3.	Train2 deceleration command Switch straight						

표 8. 열차 4대 MACC 문맥표 1(확장 기능 문맥표)

Control Action	Controlled process	Precondition State variable	State variable	State variable	Software state	Context table combination	Scenario list
	Train3 (area 2) Track switch2	Track switch 2	Train 2 (area 1) speed	Train 4 (area 4) speed			
CA 3.4 CA 4.3	Train3 stop command Switch straight	straight direction	ANY	stop	Normal	-	SS 7
				moving	Safe	MACC context table (table 5)	-
CA 3.5 CA 3.6 CA 4.3	Train2 acceleration command Train2 deceleration command Switch straight			stop	Normal	-	SS 7
				moving	Safe	3 trains MACC context table1 (table 7)	-

선로 전환기 문맥표(상황 분류 문맥표)와 조합되는 열차 3대 MACC 문맥표(확장 기능 문맥표)와 열차 4대 MACC 문맥표(확장 기능 문맥표)는 표 7과 표8과 같다. 상황 분류 문맥표에 조합되는 두개의 확장 기능 문맥표는 문맥표 작성에 있어 단순한 프로세스 변수들의 조합으로 분석이 불가능한 상황이 있으며 상황 분석을 기반으로 문맥표를 설계하고 조합하는 방법으로 문맥표의 크기를 최적화할 수 있다는 것을 확인할 수 있다.

열차 3대 문맥표1에서는 선로 전환기의 상태가 전제 조건 상황 변수로 추가되며 제어 대상 또한 선로 전환기가 추가된 것을 확인할 수 있다. 열차가 선행열차와 후행열차의 2대 사이 MACC기능만을 고려하면 되는 상황에서는 기본기능 MACC 문맥표를 조합하여 문맥표 크기를 최적화한다. 열차 3대 문맥표1에서는 다음과 같은 1개의 사고 시나리오가 추가적으로 식별된다. “SS 7: 지역 4와 지역 6을 점유한 상태에서 두 열차가 정지하면 해당 구역에서 열차가 계속 정차하게 되며 MACC 시스템이 동작할 수 없다.” 사고 시나리오 ”SS7”은 소프트웨어의 상태가 정상(Normal)으로 분석되었지만 사고 시나리오로 식별이 되었다. 사고를 일으키는 직접적인 위험(Hazardous) 상태는 아니지만 MACC 시스템의 운영이 정지되는 상황이다. 따라서 이후의 상황에 대한 상세한 분석이 필요하다.

열차 4대 문맥표1은 앞서 식별된 사고 시나리오 “SS7”에 대한 구체적인 분석을 가능하게 하는 확장 기능 문맥표이다. 추가적으로 식별되는 사고시나리오는 존재하지 않지만 열차가 3대일 때 식별되는 사고 시나리오 “SS7”이 여전히 해결되지 못하여 시스템이 동작할 수 없는 상태인 것을 확인할 수 있다.

6. 결론

본 연구에서는 소프트웨어의 복잡도가 증가에 따라 소프트웨어 위험원 분석을 위해 작성되는 문맥표의 크기는 최적화하고 최종 분석 결과는 기존과 동일하게 유지할 수 있는 방법에 대해 연구를 진행하였다. 제안하는 상황 분석 기반의 문맥표 조합 방법은 소프트웨어 위험원 분석을 실시하기 이전에 프로세스 모델 분석을 실시하여 기본 기능 문맥표의 조합 가능성을 분석하고 이를 적용하여 문맥표의 크기를 최적화하는 결과를 확인하였다. 문맥표 조합이 문맥표 전체 크기를 최적화하며 각각의 문맥표 작성 시에는 규칙 접근법을 적용하여 조합되는 문맥표의 크기를 최적화할 수 있다.

표 9. 문맥표 크기 비교

	Previous context table applied	Rule-based context table applied	Combination context table applied
MACC System context table size	27	5	-
Track switch added MACC System context table size	Unable to measure size	153	22

제안하는 상황 분류 기반의 문맥표 조합 방법은 기존 문맥표의 한계점들을 해결하며 문맥표의 크기를 최적화할 수 있는 것을 선로전환기가 추가된 MACC에 적용하여 확인할 수 있었다. 표9는 제안하는 방법이 문맥표의 크기를 최적화하여 위험원 분석의 비용과 노력 측면에서 효율성이 얼마나 증가하는지 비교하기 위한 표이다. 문맥표를 조합하기 이전에 작성되는 기본 기능 문맥표인 MACC시스템 문맥표의 크기가 규칙을 적용한 후에는 27에서 5로 최적화된 것을 확인할 수 있다. 선로전환기가 추가된 MACC의 복잡한 소프트웨어 위험원 분석에서는 측정할 수 없을 정도의 크기가 커지며 문맥표 작성에 어려움이 존재하였다. 제안 방법에서는 프로세스 분석을 실시하여 상황 분류 문맥표를 작성하고 기본 기능 문맥표와 확장 기능 문맥표를 조합하여 크기가 153인 시스템 전체의 문맥표 작성이 가능한 것을 확인하였다. 또한 조합되는 모든 문맥표에 규칙이 적용한 후에는 문맥표의 크기가 22로 최적화되는 것을 확인하였다.

참고 문헌

[1] J. Lee and D. Lee, "Safety Analysis Method for Software-based Safety-Critical System," Communications of the Korean Institute of Information Scientists and Engineers, Vol. 33, No. 7, pp. 41-46, 2015. (in Korean).

[2] J. Park, J Baik, and J. Shin, "A System Reliability Modeling Approach with Consideration of Hardware and Software Interactions," Proc. of the KIISE Conference, Vol.38, No.2B, pp. 147-150, 2011. (in Korean).

[3] N. Leveson, Engineering a Safer World: Systems Thinking Applied to Safety, The MIT Press, 2012 A.

[4] N. Leveson and J. Thomas, (2018). STPA HANDB OOK [Online]. Available: https://psas.scripts.mit.ed u/home/ (downloaded 2019, Mar. 5).

[5] N. Leveson, (2013). An STPA Primer [Online]. Av ailable: https://psas.scripts.mit. edu/home/ (downloaded 2019, Feb. 21).

[6] A. Abdulkhaleq, A System-Theoretic Safety Engineering Approach for Software-Intensive Systems, Ph.D. Dissertation, Universität Stuttgart, 2017.

[7] H. Yang and G. Kwon, "A Case Study of Railway Software Safety Analysis with STAMP/STPA," Proc. of the KIISE Korea Computer Congress, pp. 607- 609, 2017. (in Korean).

[8] H. Yang and G. Kwon, "Identifying Causes of an Accident in STPA Using the Scenario Table", Journal of KIISE: Vol.46 No. 8, 787-799, 2019 (in Korean).

[9] Ministry of Science and ICT, Korea Information and Communication Technology Association, (2018) Hazard Analysis Guide Using STPA [Online]. Av ailable: https://sw.tta.or.kr/notify/data_view.jsp?no=63(downloade d 2019, Oct. 5).

[10] J. Rising and N. Leveson, Systems-Theoretic Process Analysis of space launch vehicles, Journal of Space Safety Engineering, Volume 5, Issues 3-4, Pages 153-183, September-December 2018.

[11] Information-technology Promotion Agency, (2016). First Time STAMP/STPA (Foundation Level): New Safety Analysis Method based on System Thinking (in Japanese) [Online].Available:https://www.ipa.go.
jp/sec/reports/20160428.html (downloaded 2018, Dec. 5).

[12] Information-technology Promotion Agency, (2017). First Time STAMP/STPA (Practice Level): New Safety Analysis Method based on System Thinking (in Japanese) [Online].Available:https://www.ipa.go.
jp/sec/reports/20170324.html (downloaded 2019, Jan. 11)

[13] D. Gurgel, C. Hirata and M. Bezerra, A RULE-BASED APPROACH FOR SAFETY ANALYSIS USING STAMP/STPA, 34th Digital Avionics Systems Conference September 13-17, 2015.

결함 트리와 마코프 모델을 이용한 안전 기능의 정량적 검증

Ngoc-Tung La^{O}, 김소연, 권기현
경기대학교 컴퓨터공학과
{latung, sharonso, khkwon}@kgu.ac.kr

Quantitative Verification of Safety Function using Fault Tree and Markov Model

Ngoc-Tung La^{O}, Soyeon Kim, Gihwon Kwon
Department of Computer Engineering, Kyonggi University

요 약

안전 기능은 위험 사건이 발생되었을 때 시스템을 안전한 상태로 유지시켜주는 안전 보호 장치이다. 실행 요청을 받았을 때 안전 기능은 고장없이 동작해야 하며, IEC 61508 은 안전 기능의 고장율을 안전 무결성 등급(SIL)으로 정해놓고 있다. 본 논문에서는 결함 트리와 마코프 모델을 이용하여 SIL 검증을 수행한다. 결함 트리에서는, 안전 기능의 실행이 실패하는 원인을 하향식으로 모델링한후에, 거꾸로 단말 노드에서 루트로 올라가면서 고장율을 계산한다. 결함 트리는 단순한 반면에, 정적 모델이다. 마코프 모델은 상태 기반으로 안전 기능을 모델링한후에, 안정 상태 확률을 통해서 고장율을 계산한다. 결함 트리에 비해서 복잡하지만, 동적 모델이어서 시간에 따른 고장율을 분석에 용이하다. 두 방식의 정확성 확인을 위하여, IEC 61508 고장율 표준 공식과 비교한 결과, 거의 동일한 값을 얻었을 뿐만 아니라, 결함 트리와 같은 정적 모델로는 원인 분석을, 마코프 모델 같은 동적 모델로는 시간의 흐름에 따른 고장율 추이를 분석할 수 있었다.

1. 서론

IEC 61508 에 의하면, 안전 기능(safety function)이란 위험 사건이 발생되었을 때 시스템을 안전한 상태로 유지시켜주는 안전 보호 장치이다(IEC 61508, Part 4, p.19)[1]. 안전 기능은 위험 사건이 발생되었을 때 즉시 실행되어야 한다. 요청을 받았는데도, 고장으로 말미암아 안전 기능이 실행되지 않는다면 끔직한 사고가 발생할 수 있다. IEC 61508 은 안전 기능의 고장율을 네 등급으로 구분하여, 이를 안전 무결성 등급 (Safety Integrity Level, 줄여서 SIL) 이라고 부른다. SIL 은 위험 사건이 발생되었을 때, 안전 기능이 얼만큼 고장없이 잘 작동할 수 있는지를 나타낸다. 안전 기능의 고장율이 특정 SIL 구간에 있는지를 확인하는 작업을 SIL 검증(SIL verification)이라고 부른다[2].

본 논문에서는 두 가지 방법으로 SIL 검증 문제를 다룬다. 첫째, 결함 트리를 이용해서 안전 기능 수행을 요청 받았는데도 동작에 실패하는 원인을 하향식으로 모델링한 후에, 반대 상향식 방식으로 고장율을 계산하여, 전체 시스템의 고장율이 주어진 SIL 구간을 만족하는지를 검증한다. 둘째, 안전 기능의 동작 실패 방식을 마코프 모델로 모델링하고, 안정 상태 확률로 계산된 고장율이 SIL 구간을 만족하는지를 검증한다. 두 방법의 모델링 및 계산 결과의 정확성을 확인하기 위하여 사례 연구를 수행한다.

본 논문의 기여는 다음과 같다. 첫째, 이전에도 SMV, SPIN 같은 모델 검증 도구로 안전 기능을 검증하려는 연구가 있었다[3]. 그러나 이들 도구들은 정성적인 값만 다룰 뿐, SIL 검증에서 요구되는 정량적인 값은 다루지 못한다. 둘째, 현업에서 조차도 SIL 검증은 어려운 분야여서, 전문가에게 맡기는 것이 보통이다. 제안된 방법을 이용하면 비전문가도 SIL 검증을 수행할 수 있다.

2. 배경 지식

안전 기능의 고장율을 위한 척도로서 썸머빌의 소프트웨어 공학 교재에서는 POFOD(Probability of Failure on Demand)를 사용한다 [4]. 한편, IEC 61508 은 POFOD 와 유사한 PFD(Probability of dangerous Failure on Demand)를 사용한다. POFOD 가 일반적인 고장을 다루는데 비해서, PFD 는 위험 고장을 다룬다. 표 1 은 SIL 등급별 평균 위험 고장율을 보인다. SIL 검증이란, 안전 기능의 평균 고장율이 주어진 SIL 구간 내에 있음을 보이는 것이다.

표 1. IEC 61508 의 SIL 등급

SIL 등급	낮은 요청 모드 하에서 평균 고장율
SIL 4	$10^{-5} \le PFD < 10^{-4}$
SIL 3	$10^{-4} \le PFD < 10^{-3}$
SIL 2	$10^{-3} \le PFD < 10^{-2}$
SIL 1	$10^{-2} \le PFD < 10^{-1}$

3. 결함 트리를 이용한 SIL 검증

결함 트리는 배우기 쉽고 사용하기 단순하다는 이유로 위험원 평가, 근본 원인 분석 등 다양한 용도로 활용되고 있으며, SIL 검증에도 사용될 수 있다.

결함 트리를 이용하여 SIL 검증하는 과정을 요약하면 다음과 같다. 첫째, 하향식 방식으로 루트 노드에서 시작해서 밑으로 내려가면서 안전 기능의 동작 실패 원인을 모델링한다. 둘째, 모델링이 다 완료되면, 상향식 방식으로 단말 노드에서 시작하여 위로 올라가면서 고장율을 계산한다. 계산을 위해서는 AND 게이트, OR 게이트, 단말 노드, moon 구조, 컷셋 뿐만 아니라 공통 원인 고장인 CCF(Common Cause Failure)도 고려해야 한다. 본 논문에서는 CCF 를 다루기 위해 β 요소 기법을 사용한다[5]. 이렇게 계산하면 루트 노드의 값이 PFD 이다.

SIL 3 등급의 과압 보호(Overpressure Protection) 시스템을 사례로 다룬다[6]. 이 시스템은 센서를 갖는 입력부, 로직을 다루는 제어부, 밸브를 제어하는 출력부로 구성된다. 이 시스템의 위험 사건은 설정 압력 이상으로 과다한 압력이 발생한 경우이다. 제안한 방법을 이용하여 결함 트리로 모델링 하고, 단말 노드에서 루트 노드로 올라가면서 고장율을 계산한 결과, 루트 노드의 고장율이 1.80E-04 이어서, 목표한 대로 SIL 3 을 만족한다.

4. 마코프 모델을 이용한 SIL 검증

마코프 모델은 시스템을 상태 기반으로 모델링한 후에 시간이 흐름에 따라서 시스템의 상태 변화를 분석하는 기법이다[7]. 크게 시간 종속적인 분석과 안정 상태 분석이 있다. 안정 상태 분석에서는 시스템의 상태가 고장과 회복에 따라서 계속 변화해 가다가, 언젠가는 더 이상 변화지 않는 상태에 이르게 된다. 즉, 안정 상태에 도달했을 때 시스템이 각 상태에 머무를 확률이 안정 상태 확률이다. 특히 고장 상태의 안정 상태 확률이 PFD 값이다.

본 논문에서는 안전 기능을 CTMC(Continuous-time Markov Chain)로 모델링하였고, SIL 검증에 필요한 속성을 CSL (Continuous Stochastic Language) 로 명세 하였다. 그런 후에, PRISM 확률 모델 체킹 도구를 SIL 검증을 수행하였다[8]. 결함 트리에서 사용되었던 똑같은 데이터를 사용하여 과압 보호 시스템의 평균 고장율 마코프 모델로 계산한 결과 $10^{-4} \leq 1.79 * 10^{-4} < 10^{-3}$ 여서, SIL 3 를 만족한다.

5. 평가

결함 트리와 마코프 모델로 계산한 값을 어떻게 믿을 수 있겠는가? 이 질문에 대답하기 위해서, IEC 61508 표준에 있는 고장율 계산 공식과 비교 하였다.

표 2 에 있듯이, 본 논문에서 사용한 결함 트리와 마코프 모델을 이용한 계산 결과가 IEC 61508 표준에서 정의한 공식의 계산 결과와 일치하였고, 따라서 사례였던 과압 보호 시스템이 SIL 3 임을 입증하였다.

표 2. 세 방법의 계산 결과

	입력부	제어부	출력부	전체
결함 트리	1.50E-5	3.83E-5	1.27E-4	1.80E-4
마코프 모델	1.56E-5	3.88E-5	1.25E-4	1.79E-4
61508 공식	1.53E-5	3.83E-5	1.25E-4	1.79E-4

6. 결론

안전 기능은 위험한 사건으로부터 사람의 목숨, 재산, 환경을 지키는 안전 보호 시스템이기 때문에, 실행 요청을 받았을 때에는 가능한 고장없이 동작해야 한다. 본 논문에서는 결함 트리 및 마코프 모델을 이용하여 안전 기능의 SIL 검증을 정량적으로 수행하였다. 제안 방법의 정확성을 확인하기 위하여 과압을 방지하는 안전 보호 기능에 적용하였다. 그 결과 IEC 61508 표준 고장율 공식의 결과와 본 논문의 결과가 거의 일치함을 보였다.

본 논문에서는 IEC 61508 의 낮은 요청 모드만을 다루고 있다. 이러한 연구를 높은 요청 모드 및 ISO 26262 에서 사용되는 PMHF(Probabilistic Metrics for random Hardware Failure)로도 확장하고자 한다[9].

참고문헌

[1] IEC, “IEC 61508:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems,” 2010.

[2] E. B. Abrahamsen, "A new approach of uncertainty treatment in the verification of safety integrity level of safety instrumented system", Master Thesis, University of Stavanger, 2015.

[3 T. Grimm, D. Lettnin, and M. Hübner, "A Survey on Formal Verification Techniques for Safety- Critical Systems-on-Chip", Electronics, Vol. 7, No. 81, pp. 1-27, May 2018.

[4] Ian Sommerville, Software Engineering, 10th edition, Pearson, 2015.

[5] M. Rausand, "Reliability of Safety-Critical Systems: Theory and Applications", Wiley Publishing, 2014.

[6] Moore Industries, “White Paper: Overpressure Protection,” 2014

[7] IEC, “IEC 61165:2006 Application of Markov techniques,” 2006.

[8] M. Kwiatkowska, G. Norman and D. Parker, “PRISM 4.0: Verification of Probabilistic Real-time Systems,” In Proc. of Computer Aided Verification (CAV'11), pp 585-591, 2011.

[9] N. Das and W. Taylor, "Quantified fault tree techniques for calculating hardware fault metrics according to ISO 26262", in Proc. of Product Compliance Engineering, IEEE Xplore, pp. 1-8, May 2016.

준 지도 학습을 활용한 네트워크 패킷에서의 이상검출

어준선 ○, 장기영, 지효상, 양성봉*
연세대학교 컴퓨터과학과
{june2339, wall2010, jihyosang}@yonsei.ac.kr, sbyang@yonsei.ac.kr

Anomaly detection using semi-supervised learning in network packet data set

Joonsun Auh○, Kiyoung Jang, Hyosang Ji, Sungbong Yang*
Department of Computer Science, Yonsei University

요 약

기술이 발달함에 따라 많은 사람들이 언제 어디서든 쉽게 인터넷에 접근할 수 있다. 많은 사람이 접근하는 만큼 주고 받는 네트워크 패킷의 양도 방대해졌으며 보안을 위협하는 공격의 시도도 많아졌다. 많은 네트워크 패킷 중 공격 패킷을 잡아내기 위해 인공지능을 사용하기 위한 연구가 활발히 진행 중이다. 본 논문에서는 인공지능을 사용하여 공격 패킷을 검출하는 모델들의 성능을 향상시키고자 한다. 즉 기존에 개발된 여러가지 모델들과는 다른 방식을 적용하여 연구를 시도하였으며 그 결과와 향후 연구과제를 제시한다.

1. 서 론

인터넷이 발달하면서 많은 사람들이 인터넷에 쉽게 접근 할 수 있게 되었다. 많은 사람들이 접근하는 만큼 셀 수 없는 양의 네트워크 패킷을 주고 받는다. 하지만 모든 패킷이 안전하지는 않다. 사용자의 개인정보를 노리는 공격 패킷도 있고 회사나 단체의 활동을 멈추게 만드는 공격 패킷도 있다. 그러나 엄청난 양을 사람이 하나하나 검사하며 패킷을 확인하기는 어렵다. 따라서 인공지능을 결합하여 사람보다 많은 양의 네트워크 패킷을 정확하고 빠르게 구별해내려는 연구가 진행되고 있다.

이상 검출은 한 무더기의 데이터에서 보통 데이터들과는 다른 무리의 데이터를 검출해내는 기술이다. 이상 검출은 많은 분야에서 사용 될 수 있는데 보안 분야의 네트워크 패킷에 적용되면 많은 양의 네트워크 패킷에서 정상 패킷이 아닌 공격 패킷을 찾아 내는데 사용될 수 있다. 이미 Generative Adversaial Network (GAN)과 Autoencoder (AE)를 사용하여 많은 연구가 진행 되고 있다.

본 논문에서는 앞서 이루어진 연구를 바탕으로 새로운 방식을 사용하여 모델의 성능을 높이기 위한 연구를 진행하였다.

2. 배경 지식

2장에서는 본 연구의 이해를 돕기 위한 배경 지식을 설명한다.

2.1 Bidirectional-Generative Adversarial Network (Bi-GAN)

Bidirectional-Generative Adversarial Network (Bi-GAN)[1]은 GAN에서 파생된 모델이다. GAN은 연구[2]에서 처음으로 제안되었으며 생성자와 구별자로 구성된다. 생성자는 구별자를 속이기 위하여 랜덤 노이즈로부터 가짜 데이터를 만들어낸다. 구별자는 입력된 데이터가 진짜 데이터인지 아니면 생성자가 만들어낸 가짜 데이터인지를 구별한다. 생성자는 구별자를 속이기 위해 점점 정확도를 높이고 구별자는 속지 않기 위해 정확도를 높인다. 이렇게 서로 경쟁하며 성능이 향상 되면 구별자가 진짜 데이터와 가짜 데이터를 구분할 수 없는 데이터를 만들어내는 생성자를 얻을 수 있다.

Bi-GAN은 GAN에 인코더가 추가된 모델이다. Bi-GAN의 구조는 그림 1에서 볼 수 있는데 여기 추가된 인코더는 구별자의 역함수로 생각하면 된다. 이 인코더는 진짜 데이터를 인코딩하여 노이즈로 만들어서 진짜 데이터와 함께 구별자에 넣는다. 즉, 구별자에는 랜덤 노이즈와 가짜 데이터, 진짜 데이터와 인코딩 된 노이즈, 이렇게 두 쌍의 데이터가 들어간다. Bi-GAN의 생성자도 두 쌍의 데이터를 바탕으로 GAN의 생성자처럼 구별자를 속이기 위해 학습하여 발전한다.

2.2 K-means 클러스터링

K-means 클러스터링은 주어진 데이터를 K개의 클러스터로 묶어주는 클러스터링이다. K개의 초기 중심을 랜덤으로 선택하고 각각의 데이터들을 가장 가까운 초기 중심에 할당한다. 그리고 각 데이터들과

* 교신저자

중심과의 거리의 평균을 구한다. 구해진 평균 값을 다음 중심 값으로 설정하여 다시 각각의 데이터들을 가장 가까운 중심에 할당한 뒤, 데이터들과 중심과의 거리의 평균을 구한다. 이 과정을 반복하여 중심과 평균의 값이 차이가 없다면 데이터들이 속한 클러스터도 변화가 없으므로 클러스터 할당이 완료된다. 이 클러스터링은 비지도학습이며 정답 레이블이 없는 데이터들에게 레이블을 달아준다.

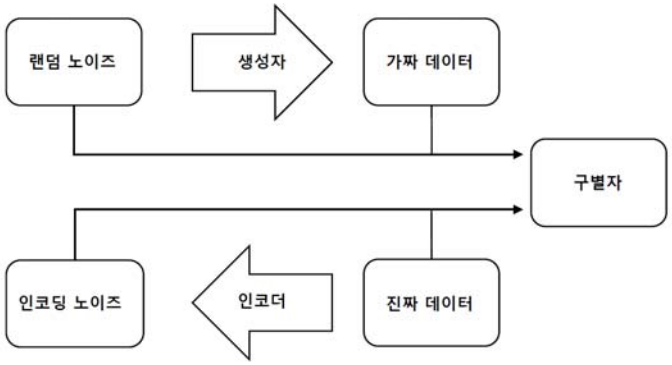


그림 1 Bi-GAN의 구조

3. 관련 연구

기존 연구에 대한 조사는 GAN이나 Autoencoder를 사용하여 이상 검출을 실시한 연구를 조사하였다. 연구[3]에서는 처음으로 GAN을 사용하여 이상 검출을 시도하였다. 의료분야에 쓰였으며 이상 검출 점수를 측정하였다. 정상인 사람을 정상 데이터로 보았으며 환자를 비정상 데이터로 봤다. 그리하여 이상 검출의 점수가 일정 수치를 넘어갈 경우 환자로 판단하였다. 연구[4]에서는 Bi-GAN을 사용하여 이상 검출을 연구하였으며 지도 학습을 사용하였다. 연구[5]에서는 Autoencoder를 사용하였다. 기존에 있던 Autoencoder의 연구와는 다른 방향으로 성능을 향상시킨 연구다. 비지도 학습을 사용하였으며 l2 정규화를 전처리 과정에서 사용하여 Autoencoder의 이상 검출 성능을 향상시켰다.

대부분의 연구가 지도 학습 아니면 비지도 학습을 사용했으며 l2 정규화를 적용시키거나 새로운 점수 측정 방식으로 성능을 향상시키려고 노력하였다. 하지만 아직 전처리 과정이나 학습방법, 특징벡터를 추출해내는 방식 등 많은 부분에서 성능의 발전이 가능할 것으로 본다.

4. 제안 모델

4장에서는 본 연구에서 사용된 모델의 설명과 실험과정을 설명한다. 본 연구에서는 Bi-GAN과 K-means 클러스터링을 섞어서 사용했다. GAN이 이상 검출을 하는 원리는 생성자가 구별자를 속일 정도로 세밀하게 가짜 데이터를 만들 수 있게 학습을 했을 때, 평소에 들어오던 입력들과 상이한 입력이 들어왔을 시 진짜 데이터와 비슷한 데이터를 만들지 못하고 확실하게 구별이 되는 가짜 데이터를 만든다는 점에서 이 모델이 이상 검출에 쓰인다. 여기서 Bi-GAN은 인코더를 사용해 진짜 데이터를 인코딩 노이즈로 바꾸는데 본 연구에서는 이 노이즈가 GAN에서 생성자가 계속 학습하여 발전하는 노이즈보다 데이터의 특징을 더 잘 나타낼 수 있다고 생각하여 Bi-GAN을 사용하게 되었다.

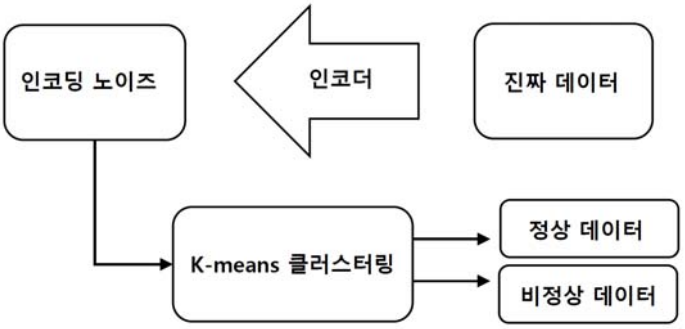


그림 2 제안 모델의 구조

정상 데이터를 학습한 Bi-GAN의 인코더에서 진짜 데이터로 만든 노이즈를 추출하여 K-means 클러스터링으로 클러스터링 한다. 그러면 이 노이즈들은 정상 데이터와 비정상 데이터로 분류된다. 본 연구에서는 구별자가 구별하는 것보다 클러스터링 방식이 정상 데이터와 비정상 데이터를 더 잘 구별할 것으로 예측하여 연구를 진행하였다. 그림 2에서 제안 모델의 구조를 확인할 수 있다.

5. 실험 및 결과

본 연구의 실험에서는 KDD 데이터셋[6]을 사용했다. KDD 데이터셋은 네트워크 보안 성능을 테스트하기 위해 만들어졌으며 최근 20년동안 사용된 데이터셋이다. 이 데이터셋은 정상 네트워크 패킷과 크게 4가지 공격 패킷으로 나뉘어있다. 또 41개의 차원을 가지고 있지만 이 실험에서 One-hot encoding을 통하여 121개의 차원을 가지게 되었으며 트레이닝셋과 테스트셋으로 나누었다. 트레이닝셋은 정상 데이터와 비정상 데이터가 섞인 전체 데이터의 절반에서 무작위로 뽑았다. 테스트셋은 남은 절반에서 정상 데이터와 비정상 데이터를 섞어서 무작위로 뽑았다.

본 실험에서는 준 지도 학습을 사용하였는데 그 이유는 매번 새로운 형태로 나타나는 네트워크 공격을 하나하나 미리 예측 할 수 없다. 하지만 미리 정상 데이터를 학습시켜 놓는다면 정상 데이터가 아닌 모든 데이터들은 비정상 데이터로 구분할 것이기에 새로운 형태의 공격 네트워크 패킷도 감지 할 수 있을 것이다.

정상 데이터인 보통의 네트워크 패킷은 크게 변하지 않을 것이라 생각했다. 따라서 본 연구에서는 일부 정상 데이터에만 레이블을 달고서 Bi-GAN을 준 지도 학습으로 KDD 데이터셋의 트레이닝셋으로 학습시켰다. 그리고 인코더에서 인코딩 된 노이즈로 K-means 클러스터링을 학습시켰다. 그리고 테스트셋을 넣어서 인코딩 노이즈를 뽑아낸 뒤, K-means 클러스터링에 넣어서 정상 데이터와 비정상 데이터로 나누었다.

성능 평가를 위해 재현도(Recall), 정확도(Precision)와 F1 점수를 사용하였다. K-means 클러스터링으로 나누어진 데이터들의 레이블 값을 원래 데이터들의 레이블 값과 비교하여 재현도, 정확도 F1점수를 계산하였다. 또한 K-means 클러스터링에서 K의 값을 2와 5로 나누어 진행하였으며 그 결과는 표 1에서 확인할 수 있다.

K	정확도(Precision)	재현도(Recall)	F1점수
2	81.2335	99.7865	89.5592
5	80.4221	99.6454	89.0076

표 1 실험결과

표1에서 확인 할 수 있듯이 K의 값이 2와 5일 때 큰 차이가 없었다. 또한 재현도는 상당히 높았으나 정확도가 낮았다.

6. 결론 및 향후 연구

기술들이 발전하고 편리해지면서 보안도 중요시된다. 특히 네트워크보안은 사용자의 개인정보가 연결되어있기 때문에 더더욱 중요하다. 네트워크의 공격은 매번 새로운 공격이 나타나기때문에 탐지하기 더욱 어렵다. 따라서 본 연구에서는 네트워크의 정상 데이터의 패턴이 변하지 않는 것에 초점을 두고 준 지도 학습을 사용하여 이상 검출의 성능을 향상시키기위한 연구를 하였다.

본 논문에서는 준 지도 학습 방법을 사용한 Bi-GAN과 K-means 클러스터링을 합쳐서 이상 검출을 시도하였으며 KDD 데이터셋을 사용하여 성능을 평가하였다. 그 결과 재현도는 상당히 높았지만 정확도는 다른 모델들에 비해 낮은 편이었다. 하지만 연구[5]에서 사용된 l2 정규화를 전처리과정에 적용시키거나 특징 벡터를 좀 더 잘 뽑아 낼 수 있다면 정확도도 더욱 향상 시킬 수 있을 것이다.

따라서 향후 연구는 제안한 모델을 기반으로 발전시켜서 정확도를 향상 시키는 방안을 연구할 계획이다.

[참고문헌]

[1] DONAHUE, Jeff; KRÄHENBÜHL, Philipp; DARRELL, Trevor. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[2] GOODFELLOW, Ian, et al. Generative adversarial nets. In: *Advances in neural information processing systems*. 2014. p. 2672-2680.

[3] SCHLEGL, Thomas, et al. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: *International Conference on Information Processing in Medical Imaging*. Springer, Cham, 2017. p. 146-157.

[4] ZENATI, Houssam, et al. Efficient gan-based anomaly detection. *arXiv preprint arXiv:1802.06222*, 2018.

[5] AYTEKIN, Caglar, et al. Clustering and unsupervised anomaly detection with l 2 normalized deep auto-encoder representations. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018. p. 1-6.

[6] Bache, K., Lichman, M.: UCI Machine Learning Repository (2013). http://archive.ics.uci.edu/ml

무인이동체 소프트웨어 시험 평가 기준 및 절차 연구

장정훈 O

(주)모아소프트

jhjang@moasoftware.co.kr

A Study on Software Test Evaluation Criteria and Procedure for Unmanned Aerial Vehicle

Jeong-Hoon Jang O

Moasoft Corp.

요 약

본 연구는 각 산업에서 요구하는 임무를 수행하는 무인이동체 시스템에서 운용되는 소프트웨어에 대한 시험 평가 기준 및 절차를 제시한다. 무인이동체 시스템의 소프트웨어는 크게 비행제어, 임무제어, 지상제어의 3가지 모듈로 구성되어 있다. 비행제어와 지상제어 소프트웨어는 이미 상당히 정형화된 수준으로 솔루션이 확보되어 있으나, 임무제어 소프트웨어는 각 산업에서 요구하는 다양한 임무를 충족시켜주기 위한 임무 관련된 요구사항의 비정형화 특성 때문에 개발된 소프트웨어에 대한 시험 및 평가가 더욱 중요하다. 본 연구는 이러한 무인이동체 소프트웨어 모듈의 특성에 따라 3등급으로 분류하고 각 등급에 대한 시험 평가 기준 및 절차를 제시한다.

1. 서 론

최근 무인이동체에 대한 기술발전이 급속도로 이루어지고 있다. 무인이동체 분야의 선도업체인 우버가 올해 미국, 호주 등 3개 도시에서 “우버에어(Uber Air)”를 시범 서비스한다고 하며, 우리나라도 2025년에 “플라잉카(Flying Car)”를 실용화한다고 선언한 상태이다. 이러한 무인이동체가 상용화하기 위해서는 아직도 법적 체계, 인프라, 제도 등 많은 장애가 산더미같이 쌓여 있지만, 무엇보다도 가장 중요한 점은 “안전성(Safety)” 확보이다[1][2][3][6][7].

또한 무인이동체는 각 산업에서 활용성이 점점 더 높아지고 있는데, 본 연구에서 대상으로 하는 무인이동체 시스템의 “임무(Mission)”는 다음과 같다.

1) 철도 시설물 점검
2) 하천 조사
3) 산불 대응
4) 다중이용시설 사고예방
5) 우편 배송

본 연구에서는 각 산업에서 요구하는 임무를 수행하는 무인이동체 시스템에서 운용되는 소프트웨어에 대한 시험 평가 절차를 제시한다.

무인이동체 시스템은 크게 비행체와 지상제어장비로 이루어 지며, 소프트웨어는 비행제어, 임무제어, 지상제어의 3가지 모듈로 구성되어 있다.

각 산업에서 무인이동체의 활용성을 더욱 향상시키기 위해서는 무인이동체 시스템의 소프트웨어에 대한 “안전성”과 “임무”를 중점적으로 검증하여야 하며, 이러한 소프트웨어의 시험 평가는 각각의 모듈 특성에 따라 소프트웨어 시험 평가 기준 및 절차를 적용하여야 한다[4][5].

2. 무인이동체 구성

무인이동체 시스템은 크게 비행체와 지상제어장비로 이루어 진다. 또한 비행체에는 임무를 수행하기 위한 광학 카메라 등 임무장비가 탑재되어 있다. 기본적인 무인이동체의 운용시나리오는 지상에서 비행체에 임무를 부여하고 필요한 장비를 탑재하며 지상제어장비에서 비행체와의 통신을 통해 비행제어 및 임무제어를 수행하게 된다. [표1]은 무인이동체의 기본적인 구성을 보여준다.

[표1] 무인이동체 구성

Level 1	Level 2	Level 3	소프트웨어
무인이동체	비행체	모터	
		프로펠러	
		배터리	
		프레임	
		FCC	O
		MCC	O
		GPS	
		통신장치	
		센서(LiDAR 등)	
	탑재장비	광학카메라	
		탐지장치	

		임무적재물	
	지상제어장비	모니터	
		조종간	
		GCS	O
		통신장치	
		저장장치	

(모터, GPS, 통신장치 등에 내장되어 있는 소프트웨어도 있으나 본 연구에서의 시험 평가 대상에서 제외함)

3. 무인이동체 소프트웨어 모듈의 특성

무인이동체 소프트웨어는 크게 3개 모듈로 비행제어컴퓨터(FCC, Flight Control Computer), 임무제어컴퓨터(MCC, Mission Control Computer), 그리고 지상제어시스템(GCS, Ground Control System)으로 구성된다. 무인이동체 소프트웨어 모듈의 주요 기능과 특성은 [표2]와 같다.

[표2] 무인이동체 소프트웨어 기능 및 특성

모듈	주요 기능	특징
FCC	모터 제어	모터 수에 따라 제어(3개 ~ 8개)
	배터리 관리	하이브리드 연료장치 가능
	자세 유지	
	비행경로 관리	
	장애물 회피	다양한 상황에 대한 알고리즘 구현
	장애 대응	장애 발생 시 안전 상태로 운행
	GPS	
	통신	RF, LTE 등
	센서(LiDAR 등) 처리	
MCC	광학카메라	
	임무장치 제어 (탐지장치 등)	임무에 따라 임무장치 다양
	임무적재물 제어	임무에 따라 적재물 다양
	데이터 관리	
	통신	RF, LTE 등
GCS	비행계획 관리	
	비행제어(수동조종)	비행체 비상 시
	비행정보 관리	비행체의 고도, 경로, 배터리 잔량 등 상태기록
	비행경로 Display	3차원 지도
	데이터 관리	
	통신	RF, LTE 등

본 연구에서 대상으로 하는 각 산업에서 요구하는 임무에 따라 MCC에 탑재되는 임무제어 소프트웨어 모듈의 주요 기능은 [표3]과 같다.

[표3] 무인이동체 임무제어 소프트웨어 주요 기능

임무	주요 기능	특징
공통임무	비행 계획	GCS 연동
	비행 경로 통신	FCC 연동
	실시간 영상 통신	GCS 연동
	영상 데이터 저장	
	임무 센서 정보 처리	
	임무 장비 상태 진단	
1) 철도 시설물 점검	전자광학(EO)장비 탑재: 근접 촬영	
	3D 디지털 맵 기반 자율 항행	
	2대 이상 다중 드론 제어	
2) 하천 조사	수심 LiDAR 탑재	
	AI/VI 활용 초분광 영상 실시간 분석	
3) 산불 대응	30Kg 소화탄 탑재 및 자동 투하	
	실시간 영상처리(산불 발생 위치 자동 분석)	
4) 다중이용시설 사고예방	(실내용) 위험물질 탐지 센서 탑재 및 감지 정보 전달	
	EO/IR 카메라, LiDAR 센서 운용	
	(실외용) 상황 인식 및 경고, 방송	
	Around View Monitoring System	
5) 우편 배송	우편물 적재함 탈부착	
	10kg 우편물 Loading/Unloading	
	다지점 배달	

전기전자, 항공, 철도, 자동차 등 대표적인 산업 분야에서 안전을 우선시 하는 시스템에서의 소프트웨어는 각 모듈의 “안전성” 등급(SIL: Safety Integrity Level, DAL: Design Assurance Level, ASIL: Automotive Safety Integrity Level)에 따라 달성해야 하는 목적이 차등적으로 요구되고 있다[6][7][8][9][10]. 각 산업에서의 소프트웨어 안전등급의 분류는 [표4]와 같다.

[표4] 산업 규격별 소프트웨어 안전등급 분류

안전등급	산업 규격			
	IEC 61508 (전기전자)	DO-178C (항공)	IEC 62279 (철도)	ISO 26262 (자동차)
등급 1	SIL-4	DAL A	SIL-4	ASIL-4
등급 2	SIL-3	DAL B	SIL-3	ASIL-3
등급 3	SIL-2	DAL C	SIL-2	ASIL-2
등급 4	SIL-1	DAL D	SIL-1	ASIL-1
등급 5	SIL-0	DAL E	SIL-0	ASIL-0

(등급 5: Non-Safety)

본 연구에서는 “등급 1”은 최상의 안전을 요구하는 소프트웨어 등급이며, “등급 5”는 안전과 관련 없는 소프트웨어에 대한 등급으로 정의한다. DO-178C에서의 안전등급 분류 기준은 [표5]와 같다[6].

[표5] 항공용 소프트웨어 안전등급 분류기준

안전등급	정의
DAL A	Catastrophic failure condition: multiple fatalities, usually with the loss of the airplane
DAL B	Hazardous failure condition: largely reduce the capability of the airplane or the ability of the flight crew
DAL C	Major failure condition: significantly reduce the capability of the airplane or the ability of the flight crew
DAL D	Minor failure condition: not significantly reduce airplane safety
DAL E	No effect: Non-Safety

각 산업 규격에서 요구하고 있는 소프트웨어의 안전등급에 따른 차등적인 소프트웨어 시험 및 평가 기준을 요약하면 [표6]과 같이 정리할 수 있다.

[표6] 소프트웨어 안전등급별 시험평가 기준

안전등급	시험평가 기준			
	Requirements Coverage	Software Structural Coverage		
		Statement Coverage	Branch Coverage	MC/DC
등급 1	O	O	O	O
등급 2	O	O	O	-
등급 3	O	O	-	-
등급 4	O	-	-	-
등급 5	-	-	-	-

(MC/DC: Modified Condition/Decision Coverage)

본 연구에서 소프트웨어 시험 평가 대상으로 하고 있는 무인이동체 시스템에서의 “임무제어 소프트웨어”에 대한 등급 분류는 “안전성” 등급과 비교하거나 유사하게 적용하기에는 무리가 있지만, 임무제어 소프트웨어의 장애에 따른 영향도에 근거하여 각 산업 규격에서 요구하고 있는 소프트웨어의 안전등급에 알맞게 부여한다.

[임무 제어 소프트웨어의 등급 정의]

등급 2: “임무” 장애로 인한 비행체의 손상은 없으나 운영자의 업무 부담이 증가함

본 연구에서는 무인이동체 소프트웨어 모듈의 특성은 안전성 등급과 임무 등급을 통합하여 다음과 같이 3등급으로 정의한다.

[무인이동체 소프트웨어의 등급 정의]

[등급 1] 비행체의 이륙, 착륙, 비행, 추락, 충돌 등과 관련된 기능을 포함하는 소프트웨어

[등급 2] 임무 수행과 관련된 기능을 포함하는 소프트웨어

[등급 3] 비행 데이터 저장 및 표시, 경로 추적, 상태 진단 등과 관련된 기능을 포함하는 소프트웨어

4. 무인이동체 소프트웨어 시험 평가 기준 및 절차

각 산업 규격에서 요구하고 있는 소프트웨어 등급 별 소프트웨어 검증 및 시험평가 기준은 [표7]과 같이 요약할 수 있다. 소프트웨어 개발 생명주기에서 수행하는 활동을 계획-요구사항-설계-구현-시험의 5단계로 구분하고 있으며, 소프트웨어 안전등급에 따라 각 단계의 활동 및 결과물에 대한 검증을 요구하고 있다[6].

[표7] 소프트웨어 개발 단계별 검증 요건

안전등급	계획	요구사항	설계	구현	시험
등급 1	필수	필수	필수	필수	[표7] 참조
등급 2	필수	필수	필수	필수	
등급 3	필수	필수	필수/일부선택 (비고1)	필수/일부선택 (비고2)	
등급 4	선택	선택	선택	선택	
등급 5	선택	선택	선택	선택	

(비고1: 설계 단계 검증 시 주요 필수항목은 요구사항 적합성/추적성, 설계 정확성, 설계 표준 준수성 등이며, 선택항목은 운영환경 호환성, 설계 검증가능성이다.)

(비고2: 구현 단계 검증 시 주요 필수항목은 설계 적합성/추적성, 소스코드 정확성, 통합 완전성 등이며, 선택항목은 소스코드 검증가능성이다.)

소프트웨어 개발 단계별 검증 요건은 구현된 소스코드에 대한 시험 평가 기준과 부합하므로, 본 연구에서 제시하는 각 등급별 소프트웨어 검증 및 시험평가 기준으로 무인이동체 소프트웨어에 대한 시험평가 기준은 다음과 같이 정의한다.

[기준 1] 소프트웨어 모듈의 기능에 대한 검증
- 기능에 대한 검증을 통해 Requirements Coverage를 달성한다.

[기준 2] 소프트웨어 모듈 간 인터페이스 검증
- 인터페이스에 대한 검증은 소프트웨어 설계 검증이며, 모듈 통합 및 하드웨어 또는 타 시스템과의 연동을 검증한다.

[기준 3] 소프트웨어 소스코드에 대한 구조적 검증
- 소스코드의 구조적 검증은 Statement Coverage, Branch Coverage, MC/DC를 달성한다([표7] 참고).

무인이동체 소프트웨어 모듈 및 기능의 등급에 따라 적용할 시험평가 기준은 [표8]과 같이 제시한다.

[표8] 무인이동체 소프트웨어 등급별 시험평가 기준

등급-기준	기준1	기준2	기준3
등급 1	O	O	O
등급 2	O	O	O(비고 1)
등급 3	O		O(비고 2)

(비고1: 등급2에 대하여 3가지 소스코드 구조적 검증 중에서 Statement Coverage와 Branch Coverage를 적용한다.)
(비고2: 등급3에 대하여 3가지 소스코드 구조적 검증 중에서 Statement Coverage를 적용한다.)

무인이동체 소프트웨어 시험평가 기준 별 절차는 다음과 같다.

[기준 1에 대한 시험평가 절차]

소프트웨어의 기능 요구사항에 기반하여 시험평가를 수행하여 소프트웨어 요구사항이 모두 구현되고 정상적으로 동작하는지를 검증함

- 기능 요구사항에 대한 정상/비정상 시험항목을 설계하고 시험 결과를 분석하여 Requirements Coverage를 달성한다.

[기준 2 에 대한 시험평가 절차]

소프트웨어 구조 설계에서 모듈 구조, 함수 호출 및 파라미터(매개변수) 전달관계와 관련된 시험을 수행함

- 소프트웨어 모듈 통합 시험, 하드웨어 통합 시험 및 외부 타 시스템과의 연동을 검증한다.

[기준 3 에 대한 시험평가 절차]

소스코드의 안전성을 확보하기 위하여 코딩 규칙 준수 및 소스코드의 구조적 검증을 수행함

- 코딩 규칙은 SW설계단계에 정의하고, 구현된 소스코드에 대하여 코딩 규칙 준수 여부를 검증함
- 소스코드의 구조적 검증은 Statement Coverage, Branch Coverage, MC/DC를 달성한다([표9] 참고).

5. 결론

본 연구에서는 무인이동체 소프트웨어에 대한 시험평가 기준 및 절차를 소프트웨어의 특성에 따라 등급을 부여하여 각 등급에 따라 제시하였다. 소프트웨어 설계 시에 모듈 구성이 매우 중요하며 소프트웨어의 구성에 따라 여러 개의 모듈로 나누어질 경우에는 각 모듈에 등급을 부여한다.

본 연구에서 분류한 3개의 무인이동체 소프트웨어에 대한 대표적인 등급은 “FCC-등급1, MCC-등급2, GCS-등급3”으로 제시할 수 있지만, [표2]에서 나열된 무인이동체 소프트웨어의 기능 및 특성에 대하여 소프트웨어의 세부 모듈을 구성하고 각 세부 모듈의 특성을 분석하여 등급을 부여하는 것은 바람직하다.

본 연구의 결과는 [표3]에서와 같이 본 연구에서 대상으로 하는 각 산업에서 요구하는 임무에 따라 FCC, MCC 및 GCS에 탑재되는 소프트웨어가 개발되고 있으며, 해당 소프트웨어 모듈에 대한 시험 평가의 기준으로 적용할 예정이며, 적용 결과는 후속 연구에서 제시한다.

6. 참고 문헌

[1] F2911-14e1, Standard Practice for Production Acceptance of Small Unmanned Aircraft System (sUAS), ASTM, 2014
[2] 항공안전법, 국토교통부, 2017
[3] KS W 9001, 무인 항공기 시스템 - 무인동력 비행장치의 설계, 산업통상자원부 국가기술표준원, 2018
[4] IEEE Std. 829 Software and System Test Documentation, IEEE, 2008
[5] IEEE Std. 1008 Software Unit Testing, IEEE, 1987
[6] DO-178C, Software Considerations in Airborne Systems and Equipment Certification, RTCA, 2011
[7] DO-278A, Software Integrity Assurance Considerations for Communication, Navigation, Surveillance and Air Traffic Management (CNS /ATM) Systems, RTCA, 2011
[8] IEC 61508-3, Functional safety of electrical /electronic/programmable electronic safety-related systems - Part 3: Software requirements, IEC, 2010
[9] IEC 62279, Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems, IEC, 2015
[10] ISO 26262-6, Road vehicles - Functional safety Part 6: Product development at the software level, ISO, 2011

(본 연구는 국토교통부 / 국토교통과학기술진흥원의 지원으로 수행되었음. (과제번호19DPIW-C153651-01, 과제명: 공공혁신조달 무인이동체 통합기술관리 및 시험평가체계 개발))

무인이동체 소프트웨어 요구사항 추적 기준 연구

장정훈 ○
(주)모아소프트
jhjang@moasoftware.co.kr

A Study on Software Requirements Trace Criteria for Unmanned Aerial Vehicle

Jeong-Hoon Jang○
Moasoft Corp.

요 약

본 연구는 각 산업의 임무를 수행하는 무인이동체 소프트웨어의 모듈 특성에 따른 요구사항 추적 기준을 제시한다. 무인이동체 시스템의 소프트웨어를 구성하는 비행제어, 임무제어, 지상제어 중에서 비행제어와 지상제어 소프트웨어는 거의 정형화된 수준의 솔루션이 확보되어 있다. 하지만 임무제어 소프트웨어는 각 산업에서 요구하는 다양한 임무를 충족시키기 위한 관련 요구사항의 비정형화 특성 때문에 개발된 각 소프트웨어에 대한 추적성 확보가 더욱 중요하다. 본 연구는 무인이동체 소프트웨어 모듈 특성에 따라 3가지의 등급으로 분류하고 각 등급에 대한 요구사항 추적 기준을 제시한다.

1. 서 론

최근 각 산업분야에서 무인비행장치(소형 드론) 등 무인이동체에 대한 관심이 높아지면서, 이에 따라 기술발전이 급속도로 이루어 지고 있다. 이러한 무인이동체의 상용화를 위해서는 아직도 법적 체계, 인프라, 제도 등 많은 과제가 쌓여 있지만, 가장 중요한 점은“안전성(Safety)” 확보이다[1][2][3][7][8].

무인이동체는 각 산업의 요구되는 임무에 따라 활용성이 점점 더 높아지고 있다. 본 연구에서의 무인이동체 시스템의 “임무(Mission)”는 다음과 같다.

1) 철도 시설물 점검
2) 하천 조사
3) 산불 대응
4) 다중이용시설 사고예방
5) 우편 배송

본 연구에서는 각 산업에서 요구하는 임무를 수행하는 무인이동체 시스템에서 운용되는 소프트웨어에 대한 요구사항 추적 기준을 제시한다.

무인이동체 시스템은 크게 비행체와 지상제어장비로 이루어 지며, 소프트웨어는 비행제어, 임무제어, 지상제어 모듈로 구성된다.

각 산업에서 무인이동체의 활용성을 더욱 향상시키기 위해서는 무인이동체 소프트웨어에 대한 “안전성”과 “임무”에 대한 요구사항을 중점적으로 추적하여야 하며, 이러한 “안전성” 및 “임무” 관련 요구사항을 추적하여 구현된 소프트웨어에서 확인하여야 한다. 이러한 소프트웨어의 추적성을 확보하기 위해서는 각각의 모듈 특성에 따라 소프트웨어 요구사항에 대한 추적 기준을 적용하여야 한다[4][5][6].

2. 무인이동체 구성

무인이동체 시스템의 구성 중 하나인 비행체에는 임무를 수행하기 위한 탐지장치 등의 임무장비가 탑재된다. 무인이동체의 기본적인 운용시나리오는 지상에서 비행체에 임무를 부여하고, 필요한 장비를 탑재한다. 그리고 지상제어장비에서 비행체와의 통신으로 비행제어 및 임무제어를 수행한다. [표1]은 무인이동체의 기본적인 구성을 보여준다.

[표1] 무인이동체 구성

Level 1	Level 2	Level 3	소프트웨어
무인이동체	비행체	모터	
		프로펠러	
		배터리	
		프레임	
		FCC	O
		MCC	O
		GPS	
		통신장치	
		센서(LiDAR 등)	
	탑재장비	광학카메라	
		탐지장치	
		임무적재물	
	지상제어장비	모니터	
		조종간	
		GCS	O
		통신장치	

(모터, GPS, 통신장치 등에 내장되어 있는 소프트웨어도 있으나 본 연구에서의 요구사항 추적 대상에서 제외함)

3. 무인이동체 소프트웨어 모듈의 특성

무인이동체 소프트웨어는 크게 3개 모듈로 비행제어컴퓨터(FCC, Flight Control Computer), 임무제어컴퓨터(MCC, Mission Control Computer), 그리고 지상제어시스템(GCS, Ground Control System)으로 구성된다. 무인이동체 소프트웨어 모듈의 주요 기능과 특성은 [표2]와 같다.

[표2] 무인이동체 소프트웨어 기능 및 특성

모듈	주요 기능	특징
FCC	모터 제어	모터 수에 따라 제어(3개 ~ 8개)
	배터리 관리	하이브리드 연료장치 가능
	자세 유지	
	비행경로 관리	
	장애물 회피	다양한 상황에 대한 알고리즘 구현
	장애 대응	장애 발생 시 안전 상태로 운행
	GPS	
	통신	RF, LTE 등
	센서(LiDAR 등) 처리	
MCC	광학카메라	
	임무장치 제어 (탐지장치 등)	임무에 따라 임무장치 다양
	임무적재물 제어	임무에 따라 적재물 다양
	데이터 관리	
	통신	RF, LTE 등
GCS	비행계획 관리	
	비행제어(수동조종)	비행체 비상 시
	비행정보 관리	비행체의 고도, 경로, 배터리 잔량 등 상태기록
	비행경로 Display	3차원 지도
	데이터 관리	
	통신	RF, LTE 등

본 연구에서 대상으로 하는 각 산업에서 요구하는 임무에 따라 MCC에 탑재되는 임무제어 소프트웨어 모듈의 주요 기능은 [표3]과 같다.

[표3] 무인이동체 임무제어 소프트웨어 주요 기능

임무	주요 기능	특징
공통임무	비행 계획	GCS 연동
	비행 경로 통신	FCC 연동
	실시간 영상 통신	GCS 연동
	영상 데이터 저장	
	임무 센서 정보 처리	
	임무 장비 상태 진단	
1) 철도 시설물 점검	전자광학(EO)장비 탑재: 근접 촬영	
	3D 디지털 맵 기반 자율 항행	
	2대 이상 다중 드론 제어	
2) 하천 조사	수심 LiDAR 탑재	
	AI/VI 활용 초분광 영상 실시간 분석	
3) 산불 대응	30Kg 소화탄 탑재 및 자동 투하	
	실시간 영상처리(산불 발생 위치 자동 분석)	
4) 다중이용시설 사고예방	(실내용) 위험물질 탐지 센서 탑재 및 감지 정보 전달	
	EO/IR 카메라, LiDAR 센서 운용	
	(실외용) 상황 인식 및 경고, 방송	
	Around View Monitoring System	
5) 우편 배송	우편물 적재함 탈부착	
	10kg 우편물 Loading/Unloading	
	다지점 배달	

전기전자, 항공, 철도, 자동차 등 대표적인 산업 분야에서 안전을 우선 시 하는 시스템에서의 소프트웨어는 각 모듈의 "안전성" 등급(SIL: Safety Integrity Level, DAL: Design Assurance Level, ASIL: Automotive Safety Integrity Level)에 따라 달성해야 하는 목적이 차등적으로 요구되고 있다[7][8][9][10][11]. 각 산업에서의 소프트웨어 안전등급의 분류는 [표4]와 같다.

[표4] 산업 규격별 소프트웨어 안전등급 분류

안전등급	산업 규격			
	IEC 61508 (전기전자)	DO-178C (항공)	IEC 62279 (철도)	ISO 26262 (자동차)
등급 1	SIL-4	DAL A	SIL-4	ASIL-4
등급 2	SIL-3	DAL B	SIL-3	ASIL-3
등급 3	SIL-2	DAL C	SIL-2	ASIL-2
등급 4	SIL-1	DAL D	SIL-1	ASIL-1
등급 5	SIL-0	DAL E	SIL-0	ASIL-0

(등급 5: Non-Safety)

본 연구에서는 "등급 1"은 최상의 안전을 요구하는 소프트웨어 등급이며, "등급 5"는 안전과 관련 없는 소프트웨어에 대한 등급으로 정의한다. DO-178C에서의 안전등급 분류 기준은 [표5]와 같다[7].

[표5] 항공용 소프트웨어 안전등급 분류기준

안전등급	정의
DAL A	Catastrophic failure condition: multiple fatalities, usually

	with the loss of the airplane
DAL B	Hazardous failure condition: largely reduce the capability of the airplane or the ability of the flight crew
DAL C	Major failure condition: significantly reduce the capability of the airplane or the ability of the flight crew
DAL D	Minor failure condition: not significantly reduce airplane safety
DAL E	No effect: Non-Safety

각 산업 규격에서 요구하고 있는 소프트웨어의 안전등급에 따른 차등적인 소프트웨어 요구사항 추적 기준을 요약하면 [표6]과 같이 정리할 수 있다.

[표6] 소프트웨어 안전등급별 요구사항 추적 기준

안전등급	요구사항 추적 단계			
	요구사항	설계	구현	시험
등급 1	O	O	O	O
등급 2	O	O	O	O
등급 3	O	O	O	O
등급 4	O	-	-	O
등급 5	-	-	-	-

본 연구에서 소프트웨어 요구사항 추적 대상으로 하고 있는 무인이동체 시스템에서의 “임무제어 소프트웨어”에 대한 등급 분류는 “안전성” 등급과 비교하거나 유사하게 적용하기에는 무리가 있지만, 임무제어 소프트웨어의 장애에 따른 영향도에 근거하여 각 산업 규격에서 요구하고 있는 소프트웨어의 안전등급에 알맞게 부여한다.

[임무 제어 소프트웨어의 등급 정의]

등급 2: “임무” 장애로 인한 비행체의 손상은 없으나 운영자의 업무 부담이 증가함

본 연구에서는 무인이동체 소프트웨어 모듈의 특성은 안전성 등급과 임무 등급을 통합하여 다음과 같이 3 등급으로 정의한다.

[무인이동체 소프트웨어의 등급 정의]

[등급 1] 비행체의 이륙, 착륙, 비행, 추락, 충돌 등과 관련된 기능을 포함하는 소프트웨어

[등급 2] 임무 수행과 관련된 기능을 포함하는 소프트웨어

[등급 3] 비행 데이터 저장 및 표시, 경로 추적, 상태 진단 등과 관련된 기능을 포함하는 소프트웨어

4. 무인이동체 소프트웨어 추적 기준

소프트웨어 개발 생명주기의 각 단계 말에 수행하는 소프트웨어 검증 활동에서 소프트웨어 요구사항에 대한 추적성을 확인한다. 요구사항 단계에서는 시스템 요구사항으로부터 할당 받은 소프트웨어 요구사항을 정의하고, 설계 단계에서는 소프트웨어 요구사항에 적합하게 소프트웨어를 설계하였는지를 추적성을 확인하여 검증한다. 각 단계의 산출물과 소프트웨어 요구사항에 대한 추적 대상은 다음과 같다.

[소프트웨어 요구사항 추적 대상]

[대상 1] 소프트웨어 요구사항 명세서, 시스템 요구사항과 소프트웨어 요구사항에 대한 추적

[대상 2] 소프트웨어 설계 명세서, 소프트웨어 요구사항과 설계항목 간의 추적

[대상 3] 소스코드, 소프트웨어 설계항목과 소스코드 간의 추적

[대상 4] 소프트웨어 시험 명세서, 소프트웨어 요구사항, 설계항목, 소스코드와 소프트웨어 시험항목(Test Case) 간의 추적

본 연구에서 제시하는 무인이동체 소프트웨어 모듈의 등급에 따라 적용할 요구사항 추적 대상은 [표7]과 같다.

[표7] 소프트웨어 등급별 요구사항 추적 대상

등급-기준	대상1	대상2	대상3	대상4
등급 1	O	O	O	O
등급 2	O	O		O
등급 3	O			O

무인이동체 소프트웨어 추적 대상에 따라 추적 기준은 다음과 같다.

[등급 1에 대한 요구사항 추적 기준]

- 시스템 요구사항과 소프트웨어 요구사항 간의 추적
- 소프트웨어 요구사항과 설계항목 간의 추적
- 소프트웨어 요구사항과 소스코드 간의 추적
- 소프트웨어 요구사항과 시험항목 간의 추적
- 소프트웨어 설계항목과 소스코드 간의 추적
- 소프트웨어 설계항목과 시험항목 간의 추적

[등급 2에 대한 요구사항 추적 기준]

- 시스템 요구사항과 소프트웨어 요구사항 간의 추적
- 소프트웨어 요구사항과 설계항목 간의 추적
- 소프트웨어 요구사항과 시험항목 간의 추적
- 소프트웨어 설계항목과 시험항목 간의 추적

[등급 3에 대한 요구사항 추적 기준]

- 시스템 요구사항과 소프트웨어 요구사항 간의 추적
- 소프트웨어 요구사항과 시험항목 간의 추적

5. 결론

본 연구에서는 무인이동체 소프트웨어의 특성에 따라 등급을 부여하는 기준을 제시하고, 각 등급에 따른 소프트웨어 요구사항 추적 기준을 제시하였다. 소프트웨어 설계 시에 모듈 구성이 매우 중요하며 소프트웨어의 구성에 따라 여러 개의 모듈로 나누어질 경우에는 각 모듈에 등급을 부여한다.

본 연구에서 분류한 3개의 무인이동체 소프트웨어에 대한 대표적인 등급은 “FCC-등급1, MCC-등급2, GCS-등급3”으로 제시할 수 있지만, [표2]에서 나열된 무인이동체 소프트웨어의 기능 및 특성에 대하여 소프트웨어의 세부 모듈을 구성하고 각 세부 모듈의 특성을 분석하여 등급을 부여하는 것이 바람직하다.

본 연구의 결과는 [표3]에서와 같이 본 연구에서 대상으로 하는 각 산업에서 요구하는 임무에 따라 FCC, MCC 및 GCS에 탑재되는 소프트웨어가 개발되고 있으며, 해당 소프트웨어 모듈에 대한 요구사항 추적 기준으로 적용할 예정이며, 적용 결과는 후속 연구에서 제시한다.

6. 참고 문헌

[1] ASTM F2911-14e1, Standard Practice for Production Acceptance of Small Unmanned Aircraft System (sUAS), ASTM, 2014

[2] 항공안전법, 국토교통부, 2017

[3] KS W 9001, 무인 항공기 시스템 - 무인동력 비행장치의 설계, 산업통상자원부 국가기술표준원, 2018

[4] IEEE Std. 830 Software Requirements Specification, IEEE, 1998

[5] IEEE Std. 1012 Software Verification and Validation, IEEE, 2004

[6] IEEE Std. 1016 Software Design Description, IEEE, 1998

[7] DO-178C, Software Considerations in Airborne Systems and Equipment Certification, RTCA, 2011

[8] DO-278A, Software Integrity Assurance Considerations for Communication, Navigation, Surveillance and Air Traffic Management (CNS /ATM) Systems, RTCA, 2011

[9] IEC 61508-3, Functional safety of electrical /electronic/programmable electronic safety-related systems - Part 3: Software requirements, IEC, 2010

[10] IEC 62279, Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems, IEC, 2015

[11] ISO 26262-6, Road vehicles - Functional safety Part 6: Product development at the software level, ISO, 2011

(본 연구는 국토교통부 / 국토교통과학기술진흥원의 지원으로 수행되었음 (과제번호19DPIW-C153651-01, 과제명:공공혁신조달 무인이동체 통합기술관리 및 시험평가체계 개발)).

Triplet 추출 기반 범용 온톨로지 구축 설계

김아령 ○, 이상백, 이규철†
충남대학교 컴퓨터공학과
aryoung9622@gmail.com, {roy881020, kciee}@cnu.ac.kr

Design of Universal Ontology Construction based on Triplet Extraction

A-Ryoung Kim○, Sang-Baek Lee, Kyu-Chul Lee†
Department of Computer Engineering, Chungnam National University

요 약

기존의 온톨로지는 일반적이거나 특정 영역에 제한되어 구축되기 때문에 정보 제공 범위의 한계가 있었으며, 대부분 수작업으로 구축되어 시간 및 인적 제약 때문에 실용적인 온톨로지를 구축하기 어려웠다. 본 논문에서는 정보 제공 범위의 한계를 해결하기 위해 기본적인 의미를 갖는 상위 개념과 하위 개념을 제공하며, 시간 및 인적 제약 문제를 해결하기 위해 데이터로부터 개념을 자동 추출하는 범용 온톨로지 구축 설계 방법을 제안한다. 범용 온톨로지가 상위, 하위 간의 의미 상호운용성을 가질 수 있도록 지원하기 위해 상위 온톨로지를 적용하며, 자연어 처리 분석 기술을 사용하여 비/반정형 데이터로부터 Triplet을 자동 추출하여 하위 온톨로지로 확장시킨다. 본 논문에서 제안한 방법으로 테스트 범용 온톨로지를 구축하였고, 이를 통해 실제로 구축했을 때의 활용가능성에 대해 살펴보았다.

1. 서론

웹 사용자의 수가 증가하면서 웹으로 쏟아져 들어오는 지식들이 매년 급속도로 증가하고 있다. 이러한 지식들 사이에서 의미 있는 정보를 찾는 것은 쉽지 않았으며, 이러한 문제를 해결하기 위한 노력들이 계속되어 왔다. 시맨틱 웹(Semantic web)은 컴퓨터가 쉽게 해석할 수 있는 잘 정의된 의미를 부여함으로써 의미 있는 정보를 찾을 수 있도록 하였다[1]. 오늘날 시맨틱 웹 기술이 발전됨에 따라 대량의 지식들을 효율적으로 관리하고 사용하기 위해 계속해서 다양한 기술들이 개발되고 있다. 대량의 지식들은 정형, 비/반정형 데이터들로 구성되어 있으며, 이들 간의 복잡한 관계를 효율적으로 표현하기 위한 대표적인 기술로 온톨로지가 있다. 온톨로지의 특성을 잘 표현하고 있는 [2]에 의하면 온톨로지는 개념에 대한 정형화되고 명시적인 명세를 정의하는 기술로 정의하고 있으며, 개념간의 관계(Relationships), 개념들의 속성을 나타내는 술어(Predicate), 타입(Types) 등을 표현할 수 있다. 온톨로지는 계층 구조를 형성하는 상속 관계로 개념들은 관계를 통해 다른 개념들과 연결된다. 개념은 클래스로도 표현이 되며 같은 단어들의 집합을 표현할 수 있는 대표 단어이다. 예를 들어, Animal, Zebra, Snake, Tiger라는 단어들이 있을 때, Animal은 상위 개념 또는 상위 클래스, Zebra, Snake, Tiger는 하위 개념 또는 하위 클래스로 분류된다. 이러한 계층 구조를 갖는 온톨로지는 정보 추출, 의료정보와 바이오정보, e-비지니스, 자연 언어 처리, 기업 모델링 등 다양한 기술 분야에 적용되고 있다. 온톨로지는 구축 범위에 따라 크게 일반 온톨로지와 도메인 온톨로지로 나뉠 수 있다. 일반 온톨로지는 시간, 공간, 상태, 사건 등과 같은 사물의 기본적 관념과 개념을 제공하는 세상에 대한 일반 지식을 포함, 도메인 온톨로지는 전자, 의학, 디지털 영역과 같이 특정 영역에만 통용되는 지식을 표현한 온톨로지이다[3].

현재 구축 및 연구된 온톨로지는 특정 영역에 제한되어 구축되고 적용되는 도메인 온톨로지가 대부분이며, 이는 다양한 분야에서 적용가능한 범용적인 온톨로지는 매우 부족한 상황임을 의미한다. 또한 일반 온톨로지에서는 가장 기본적인 상위 개념은 존재하지만, 그 개념에 속해있는 특징적인 하위 개념은 존재하지 않는다. 예를 들어, Animal 이라는 상위 개념은 존재하지만, Zebra, Snake, Tiger와 같은 Animal의 하위 개념에 대해서는 제공하지 않는다. 이는 온톨로지로부터 얻을 수 있는 정보 제공 범위의 한계가 존재한다는 것을 의미한다. 이러

† 교신저자

* 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임
(No. NRF-2019R1A2C1011567).

한 문제를 해결하기 위해서는 상위 개념뿐만 아니라 그 하위 개념에 대해서도 제공하고 있는 범용적인 온톨로지가 필요하다. 게다가 기존의 온톨로지들은 대부분 수작업으로 구축되고 있지만, 시간 및 인적 제약 때문에 실용적인 온톨로지를 구축하기 어렵다[4]. 때문에 수작업 소요 비용 절감 및 효율성을 향상시키기 위해 데이터로부터 개념을 자동 추출하는 방법이 필요하다.

본 논문에서는 온톨로지 정보 제공 범위의 한계점과 수작업으로 구축되는 문제점들을 해결하기 위해 일반적인 온톨로지나 특정 영역의 도메인 온톨로지가 아닌, 기본적인 상위 개념 안에 속해있는 하위 개념에 대해서도 제공하며, 입력된 데이터로부터 개념을 자동 추출하는 범용 온톨로지를 구축한다. 이를 통해 다양한 분야에서 상황을 예측하고 분석하여 추론할 수 있는 온톨로지를 구축하고자 한다.

본 논문에서는 입력된 데이터를 기계가 읽을 수 있는 형태로 변환하기 위해 가장 널리 사용되고 있는 OWL(Web Ontology Language)을 사용하여 구축하였다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구와 관련된 연구들을 소개하며, 3장에서는 본 논문에서 제안하고자 하는 시스템에 대하여 구체적으로 서술한다. 4장에서는 실험을 통해 본 논문에서 제안하는 시스템이 실제로 구축했을 때의 활용 가능성 대해 살펴보며, 마지막으로 5장에서는 연구의 결론과 향후 연구 방향에 대해서 논의한다.

2. 관련연구

온톨로지는 구축 범위에 따라 구분될 수 있으며, 일반화 정도에 따라서 구분될 수 있다. Guarino의 연구에서는[5] 온톨로지를 상위(Top-level) 온톨로지, 도메인(Domain) 온톨로지, 과업(Task) 온톨로지, 응용(Application) 온톨로지로 구분하였다. 상위 온톨로지는 시간, 공간, 물질, 대상, 사건 등 구체적인 문제나 특정 도메인과 무관한, 매우 일반적인 개념을 묘사하며, 도메인 온톨로지와 과업 온톨로지에서는 상위 온톨로지에 도입된 용어를 전문화하여 각각 특정 영역 또는 과업 수행하기 위한 개념을 설명한다. 응용 온톨로지는 특정 영역과 과업 온톨로지에 따라 종속되는 개념을 설명한다.

범용 온톨로지를 구축하기 위해서는 크게 상위 온톨로지와 하위(Lower-level) 온톨로지로 나눌 수 있다. 온톨로지를 구성하는 경우에 여러 개의 하위 온톨로지를 만들 수 있는데, 법령[6], 식품[7], 국방[8] 등 여러 개의 도메인 온톨로지로 만들 수 있다. 이를 다시 합치기 위해서는 상위 온톨로지가 필요하다[9]. 또한 상위 온톨로지는 지식의 공유와 재사용을 위하여 서로 다른 목적에서 같은 대상을 서로 다르게 분류하기 때문에 더 일반적인 수준의 상위 온톨로지가 필요하다[10].

표준으로 인정받는 여러가지 상위 온톨로지가 있다. SUMO[11], OpenCyc[12], DOLCE[13]등은 상위 온톨로지를 이용하여 도메인 온톨로지들과의 계층화를 통해 상호운용성을 향상시키기 위한 연구이다. 그러나 이러한 연구는 대부분 의미 어휘 목록 집합을 이용하여 도메인 온톨로지들과 계층화를 이루는 연구들이기 때문에 사용자로부터 다양한 정보를 제공하는데 한계가 있다.

이러한 문제를 해결하기 위해 본 논문에서는 일정한 형식이나 틀이 있는 정형 데이터의 기본적인 개념을 정의 및 생성하여 상위 온톨로지를 구축한 다음, 비/반정형 데이터로부터 자동 추출하여 얻은 데이터를 하위 온톨로지로 확장시켜 상호 연결성을 갖는 범용적인 온톨로지를 구축 설계한다.

3. 범용 온톨로지 구축 설계

본 논문에서 제안하는 그림 1과 같이 일반적이거나 한정된 특정 영역의 온톨로지가 아닌 범용적인 온톨로지를 구축하기 위해 상위 온톨로지를 OWL 언어를 사용하여 기 구축함으로써 데이터를 구조화한다. 구조화된 온톨로지에 비/반정형 데이터로부터 추출된 triplet을 하위 계층으로 분류하여 온톨로지를 구축한다.

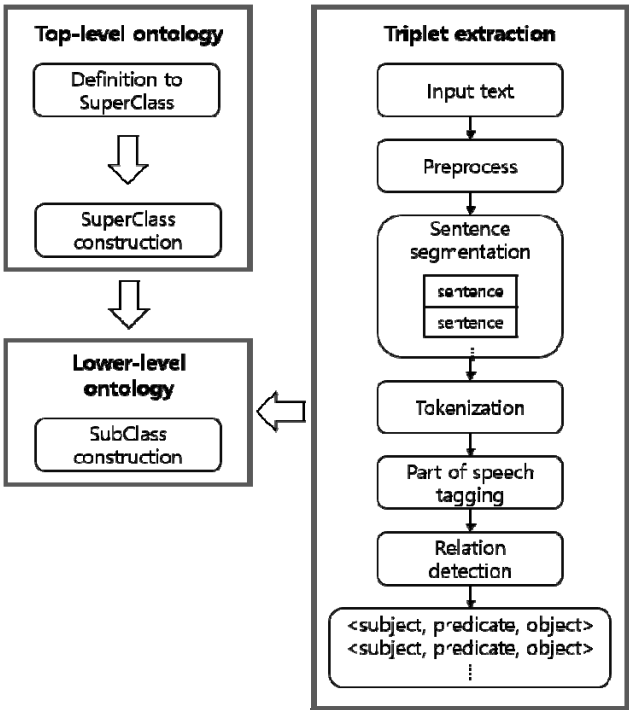


그림 1. 범용 온톨로지 시스템 구성도

3.1 상위 온톨로지

데이터를 적절하게 관리하기 위해서는 데이터가 분류되어야 한다. 데이터를 분류하기 위해서는 데이터들이 상위, 하위 간의 의미 상호운용성을 갖는 데이터이어야 한다. 상위 온톨로지의 가장 중요한 기능은 모든 영역을 지원하는 온톨로지 자체를 제공하는 것이 아니라, 많은 온톨로지들이 상위 온톨로지 아래에서 매우 포괄적인

의미 상호운용성을 가질 수 있도록 지원하기 위한 것이다[14].

상위 온톨로지를 구축하기 위해 일정한 형식이나 틀이 있는 정형 데이터로부터 기본적인 의미를 갖는 상위 개념을 상위 클래스(SuperClass)로 정의하여 생성한다. 상위 클래스를 정의할 때에는 다양한 분야의 개념들을 포함하고 있는 의미여야 하며, 정의하고자 하는 상위 클래스에 하위 클래스들이 속할 수 있는지 고려해야 한다. 상위 온톨로지를 구축할 때, 추가적으로 입력된 데이터에 상위 개념의 하위 개념들이 존재할 경우 상위 개념을 상위 클래스, 하위 개념을 상위 클래스 하위의 중간 클래스(MidClass)로 정의하여 생성한다.

3.2 Triplet 추출

W3C에서 권고하고 있는 OWL은 RDF, RDFS에 형식적인 의미를 가진 어휘(Vocabulary)를 추가하여 XML, RDF, RDFS에서 지원하는 것 보다 풍부한 의미 표현을 가능하게 한다. RDF, RDFS는 Triplet 구조로 되어 있으며, Triplet이란 세 가지 Tuple의 조합. 즉, <Subject, Predicate, Object>로 구성된 형태를 의미한다. 비/반정형 데이터인 자연어를 OWL 표현 언어로 표현하기 위해 Triplet 구조로 추출한다. 자연어는 다양한 Subject, Object와 그 사이의 관계를 나타내는 Verb 또는 Predicate로 구성할 수 있기 때문이다[15]. 효과적으로 Triplet을 추출하기 위해 자연어 처리 분석 기술을 사용하여 자연어 데이터로부터 Triplet을 자동 추출한다. 자연어 분석 기술로는 NLTK, OpenNLP, Stanford Parser, Link Parser 등이 존재하며, 입력된 문장으로부터 Sentence segmentation, Tokenize, Parts of speech 등을 수행할 수 있다.

Triplet 추출은 Preprocess, Sentence segmentation, Tokenization, Parts of speech tagging, Relation detection의 5가지 과정으로 나눌 수 있다. 먼저 Sentence segmentation을 통해 입력된 텍스트를 문장으로 분리한다. 분리된 긴 문장들은 Tokenization을 통해 문자열 단위(token)로 나누는 작업을 거치게 된다. Parts of speech tagging은 문자열 단위로 분리된 데이터에 각각 품사 정보를 부착하여 문장의 구문 구조를 형성하게 되는데, 이를 통해 문장의 의미 관계를 분석하여 Triplet을 추출한다. 추출된 Triplet을 상위 온톨로지의 하위 온톨로지로 정의하여 구축한다.

4. 실험

앞서 설명한 범용 온톨로지를 구축하는 것이 가장 이상적이지만, 범용 온톨로지를 연구하기 전에 하나의 도메인을 선정하여 다양한 분야에서 상황을 예측하고 분석하여 추론할 수 있는 테스트 범용 온톨로지를 구축하고자 한다.

본 논문에서는 특정 영역의 데이터 셋이 아닌, 다양하고 일상적인 용어들로 구성되어 있는 데이터들을 고려하여 선정하였다. 테스트 구축에 사용된 데이터는 YouTube-8M Large-Scale Video Understanding Competition에서 제공하고 있는 YouTube 8M[16] 데이터 셋을 사용하였다. YouTube 8M은 비디오가 무엇에 관한 것인지 잘 설명할 수 있는 핵심 주제(e.g sports, animals, foods)를 결정하는 작업을 하기 위해 만들어진 데이터 셋이다. YouTube 8M 데이서 셋은 약 3,800여 개 클래스를 갖고 있으며, 시각적으로 관찰 가능하고 충분히 일상생활에서 빈번하게 사용하고 있는 용어들로 구성되어 있다. 각 클래스들은 24개의 Top-level Category로 분류되어 WikiURI, WikiDescription 등 9개의 항목을 갖는다.

상위 온톨로지를 구축하기 위해 24개의 Top-level Category를 상위 클래스, 3,800여개의 클래스를 중간 클래스로 정의하여 구축하였으며, 하위 온톨로지를 구축하기 위해 WikiDescription을 사용하였다. WikiDescription으로부터 전처리 과정을 통해 데이터를 가공시켜 사용할 문장들을 추출한다. 추출된 데이터는 대표적인 자연어 처리 분석 기술인 Stanford Parser를 사용하여 Triplet을 추출하였다. Sentence segmentation, Tokenize, Parts of speech, Relation detection을 통해 Triplet을 추출하고 앞서 구축한 중간 클래스의 하위 클래스로 구축하였다.

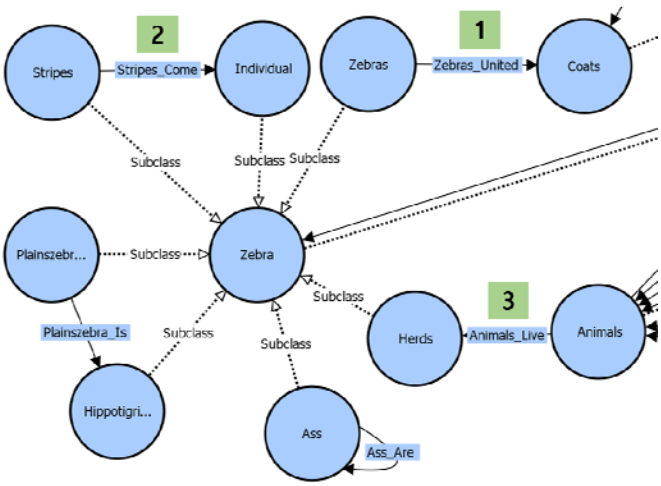


그림 2. 테스트 범용 온톨로지를 시각화한 예시

본 논문에서 제안한 테스트 범용 온톨로지는 입력 정보를 기반으로 다양한 정보를 얻음으로써 상황을 예측하고 분석하여 추론할 수 있다. 예를 들어, “Zebra”라는 입력 정보가 들어오게 되면, 테스트 범용 온톨로지 그림 2로부터 “얼룩말은 결합된 털을 갖고 있다.” “ 줄무늬가 독특하게 되어있다.” “동물들은 무리지어 산다.” 등의 정보를 얻을 수 있게 되는데, 이를 통해 “얼룩말은 독특한 줄무늬 모양의 털을 갖고있다.” “얼룩말들은 무리지어 산다.” 등을 추론할 수 있으며, 영상이나 이미지 데이터를 생성하는데 사용할 수 있다.

5. 결론 및 향후 연구

다양한 분야에서 상황을 예측하고 분석하여 추론할 수 있는 온톨로지를 구축하는 것은 매우 중요하다. 온톨로지의 중요성을 인식하면서 다양한 연구들이 활발하게 진행되어 왔다. 기존의 온톨로지는 정보 제공 범위의 한계점과 수작업으로 구축되면서 많은 시간과 비용, 인적 자원을 필요로 하는 문제를 갖고 있었다. 이를 해결하기 위해 본 논문에서는 기본적인 의미를 갖는 상위 개념과 하위 개념을 제공하며, 시간 및 인적 제약 문제를 해결하기 위해 데이터로부터 개념을 자동 추출하는 범용 온톨로지 구축 설계 방법을 제안하였다. 범용 온톨로지가 상위, 하위 간의 의미 상호운용성을 가질 수 있도록 지원하기 위해 상위 온톨로지를 적용하였고, 다양한 분야 적용 및 수작업 소요 비용 절감과 효율성을 향상시키기 위해 비/반정형 데이터인 자연어로부터 자연어 처리 도구를 이용하여 Triplet을 자동 추출하였다. 추출된 데이터를 하위 온톨로지로 분류하여 테스트 구축함으로써 범용 온톨로지의 활용 가능성에 대해 살펴보았다.

본 논문에서 제안한 범용 온톨로지는 다양한 분야에서 활용 가능한 지식들로 표현되어 있기 때문에 기존 온톨로지보다 검색 성능을 향상시킬 수 있다. 이를 통해 여러 상황들을 예측하고 분석하여 추론함으로써 향상된 사용자 중심의 서비스를 제공할 수 있을 것이다. 또한 다양한 온톨로지를 수작업으로 구축하는데 소요되는 시간 및 비용을 획기적으로 줄일 수 있을 것으로 기대된다.

향후 연구에서는 테스트 범용 온톨로지보다 확장된 온톨로지를 구축하기 위해 Triplet 추출 방법 중 정확한 정보 추출 및 관계 추출을 하기 위해 구문 관계, 의미 관계를 분석하며, 다양한 분야의 데이터를 수집하여 적용함으로써 온톨로지의 규모를 확장시키기 위한 연구가 진행 되어야 할 것이다.

6. 참고 문헌

[1] 이재호, 양정진. 시맨틱 웹:차세대 지능형 웹 기술, TTA Journal, vol.81, 2000.

[2] T. Gruber, “A Translation Approach to Portable Ontology Specifications”, Knowledge Acquisition, vol. 5, no.2, pp.199-220. 1993.

[3] 심경(아이리스닷넷), 온톨로지(Ontology), 도서관문화, vol.50, no.10, pp.49, 2009.10.

[4] 최기선, 류법모(KAIST), 온톨로지의 구축과 학습: 상하위 관계, 한국정보과학회지, vol.24, no.4, pp.24-30, 2006.4.

[5] Guarino N. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In: Pazienza MT, editor. Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology. Springer Verlag. pp.139-170, 1997.

[6] 조대웅, 김명호. 법령 온톨로지 구축에 관한 연구. 한국컴퓨터정보학회논문지, vol.19, no.11, pp. 105-113, 2014.

[7] 김은경, 김용기, 온톨로지를 이용한 식품첨가물 정보 지식의 구축, 한국지능시스템학회논문지, vol.27, no.1, pp.42-49, 2017.

[8] 장우혁, 무기체계 부품국산화 정보의 온톨로지 구축방안 연구, 멀티미디어학회논문지, vol.18, no.7, pp.873-885, 2015.

[9] 김동성, 언어지능:데이터사이언스총서, 커뮤니케이션북스, pp53~p64, 2017.

[10] 이수경, 채영문, 강건욱, 박경모, 박찬, 김곤희, 효율적인 암정보 제공을 위한 온톨로지 기반 대화시스템 개발, 한국보건정보통계학회지, vol.30, no.2, pp.31-39, 2005.

[11] I. Niles and A. Pease. Towards a standard upper ontology. In Chris Welty and Barry Smith, editors, Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS2001), 2001.

[12] OpenCyc Website, http://www.opencyc.org/.

[13] A Descriptive Ontology for Linguistic and Cognitive Engineering Website., http://www.loa-cnr.it/DOLCE.html

[14] 한국전산원, 웹 온톨로지 개발지침 연구, pp.1-4, 2004. 12.

[15] Anuja Jaiswal, Vinai George, A modified approach for extraction and association of triplets, International Conference on Computing, Communication & Automation (ICCCA 2015), p.36~40.

[16] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A largescale video classification benchmark. arXiv preprint arXiv:1609.08675, 2016.

HCAII : 고융합 AI 인프라스트럭쳐(infrastructure)에 대한 가성비 및 확장성 분석

전민규O 최규휘 김지원 성병학 정재웅 강양욱 기양석

아토 리서치, 삼성전자

mingyu.jeon@atto-research.com, kyuhwi.choi@atto-research.com, jiwon.kim@atto-research.com, byeonghag.seong@atto-research.com, jaewoong.chung@atto-research.com, yangwook.k@samsung.com, yangseok.ki@samsung.com

HCAII : Analysis on cost-effectiveness and scalability of Hyper Converged AI Infrastructure

Min-Gyu JeonO, Kyu-Hwi Choi, Jiwon Kim, Byeong-Hag Seong, Jae-Woong Chung, Yang-Wook Kang, Yang-Seok Ki

Atto Research, Samsung Electronics

요 약

빠르게 발전하는 AI 기술은 학습과 추론 시의 데이터처리 연산 요구량을 크게 증가시키고 있다. 급격히 증가하고 있는 GPU 시스템 구축비를 감축하기 위해서는 고사양 중심의 스케일 업(scale-up) 아키텍쳐가 아닌 가성비 높은 GPU의 클러스터링을 통해 전체 시스템의 성능을 높이는 스케일 아웃(scale-out) 구조여야 한다. 본 논문에서는 1) 스케일 아웃 중심의 GPU HCI(Hyper Converged Infrastructure) 아키텍쳐인 HCAII(Hyper Converged AI Infrastructure)를 제안하고, 2) 이를 스케일 업 중심의 고사양 NVIDIA GPU 및 All Flash 스토리지 서버 아키텍쳐와 비교한다. HCAII 는 높은 TFlops/$를 제공하는 AMD GPU와 SSD 안의 임베디드(embedded) ARM 코어를 이용해 비정형 데이터의 처리를 가속하는 KV-SSD를 적용하여 높은 가성비를 얻고, 하나의 서버에서 연산과 데이터 저장 기능을 모두 제공하는 HCI 형태로 시스템을 구성하여 시스템이 쉽게 스케일 아웃 되도록 한다. 마지막 실험에서는 HCAII 프로토타입이 스케일 아웃을 통해 스케일 업 아키텍쳐 대비 높은 가성비를 가지고 있음을 보였다.

1. 서 론

AI 기술의 급격한 발전은 AI 시스템이 학습과 추론 시에 요구하는 데이터처리 연산량을 급격히 증가시키고 있다. [그림 1] 은 AI 연산 요구량이 매년 10x 이상 증가하고 있음을 보여준다. 이러한 급격한 연산량의 증가는 GPU 기반의 AI 인프라 플랫폼 아키텍쳐에도 큰 도전을 안긴다.

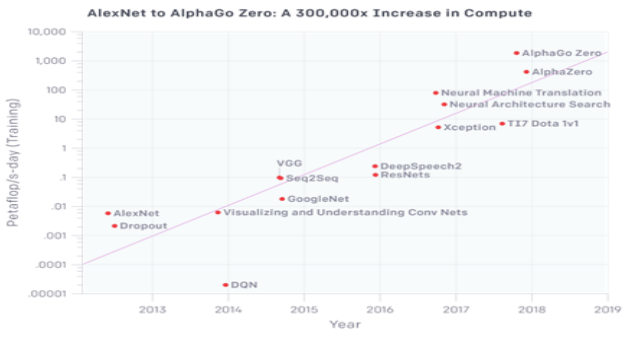


그림 1. AI 연산량의 급격한 증가[1]

본 논문에서는 스케일 아웃(scale-out) 기반의 GPU HCI[2] 아키텍쳐인 HCAII(Hyper Converged AI Infrastructure) 를 제안한다. HCAII는 가성비 높은 AMD GPU 의 클러스터링을 통해 높은 시스템 성능을 확보하며, 하드웨어 구성을 위해 AMD GPU 카드의 PCB 보드 및 쿨링시스템을 재설계하여, 단위 서버에 집적도 높게 장착이 가능하도록 한다. 또한 TensorFlow, Pytorch, Caffe 등의 주요 소프트웨어 플랫폼 지원을 위해 ROCm(Radeon Open Computing platform) 기반의 소프트웨어 스택을 적용하였다.

스케일 아웃 용의성을 갖춘 서버 구조를 위해 HCAII는 클라우드 시장에서 널리 쓰이는 구조인 HCI를 적용한다. GPU 서버에 HCI를 적용시의 문제점은 AI 플랫폼이 요구하는 CPU 자원과(예: 파라미터 서버 구동) 스토리지 시스템이 요구하는 CPU 자원이(예: Ceph[3]의 OSD) 이 하나의 서버상에서 충돌을 일으켜 단위 서버의 성능을 저하 시킬 수 있다는 점이다. 이를 해결하기 위해 HCAII 는 KV(Key Value)-SSD[4]를 적용하여 스토리지 시스템의 CPU 요구량을 줄인다.

HCAII 프로토타입 서버는 재설계된 4장의 AMD RX580 카드와 1TB KV-SSD로 구성된다. 클러스터 시스템의 자원관리를 위해 OpenStack 기반의 HCI 관리 시스템이 이용되었으며 스토리지 시스템은 Ceph가 적용되었다. AMD GPU상의 AI 소프트웨어 구동을 위하여 ROCm 컨테이너 이미지가 적용되었다. 비교 대상인 스케일 업(scale-up)

시스템은 2개의 NVIDIA v100 GPU와 All Flash 스토리지(storage)로 구성되며, 나머지는 동일한 조건이다.

실험은 TensorFlow 상에서 Alexnet과 ResNet 네트워크를 통한 가성비 비교를 진행하고, KV-SSD 기반의 스토리지 시스템과 일반 SSD의 CPU 자원 소모량을 비교하여 GPU HCI 구성을 통한 AI 시스템의 가성비 및 확장성을 검증한다.

2. 스케일 업(scale-up) vs 스케일 아웃(scale-out)

점차 증가하는 데이터를 처리하기 위해 더 높은 데이터 처리 능력을 가진 AI 인프라스트럭쳐(infrastructure)가 필요하게 되었다. 이를 위해서 일반적으로 스케일 업(scale-up) 방식과 스케일 아웃(방식이 고려되고 있다.

스케일 업 AI 인프라스트럭쳐는 고사양 GPU 카드들로 구성되며, 전용 머신을 구매하여 사용 하는 경우와 기존 서버 및 데스크탑에 GPU 카드를 추가하여 사용하는 경우가 있다. NVIDIA의 DGX서버가 전자의 예이다. 대표적인 고사양 GPU 카드로는 NVIDIA Tesla V100이 있다. 현재 AI를 위한 GPU 카드 중 최고 사양으로 16GB 장비 기준 $6,600 정도의 가격을 가지고 있다[5]. 이 경우 GPU 카드 1장당 처리 성능은 높지만 가격 역시 비싸서 처리 데이터량 증가에 따른 장비 추가에 대한 비용 부담이 발생한다.

이와 같은 문제를 해결하기 위해서는 가성비 좋은 GPU 카드들을 이용한 스케일 아웃 형태의 AI 인프라스트럭쳐가 고려되고 있다. HCAII와 같이 스케일 아웃 방식을 통해 저렴한 비용의 GPU 다수를 확보하여 전체 처리 성능을 스케일 업 방식보다 높게 가져갈 수 있다.

3. HCAII 구조

3.1 AMD GPU

스케일 아웃에 적합한 GPU 클러스터링 구축을 위해서는 가성비 높은 GPU 카드 사용과 이에 적합한 소프트웨어 스택 적용이 중요하다. [표 1]은 NVIDIA와 AMD사의 고사양 및 중저가 GPU카드의 가격대 성능비를 보여준다. HCAII에서는 가성비가 가장 높은 RX 580을 사용하여 확장성을 얻고자 한다.

표 1. GPU 카드 가격대 성능 정보[5,6,7]

	NVIDIA GPU		AMD GPU	
	2080 Ti	V100	Rx 580	Vega 64
FP 32	11.75	15	6.175	12.5
가격	$ 1,300	$ 6,599	$ 190	$ 500
GFlops/$	9.04	2.27	32.5	25.2

오늘날 GPU 가속에 이용되는 소프트웨어는 대개 NVIDIA의 GPU에서만 동작하는 CUDA 라이브러리를 이용한다. 이에 대한 대응으로 AMD는 ROCm을 공개 하였다. ROCm은 이론적으로 언어 및 하드웨어로부터 독립적인 GPU용 미들웨어 계층을 구현한다. [그림 2]은 ROCm의 구조를 나타낸다. ROCm 구성 요소 중 하나인 HIP은 엔비디아의 CUDA 라이브러리를 사용하는 CUDA 코드에 대해 C++코드로 변환시켜주는 HIPify와 새로 작성된 C++코드를 NVIDIA 혹은 AMD 환경에서 동작 가능하도록 컴파일 해주는 HIPCC로 구성이 되어 있다. 이 특징을 이용하여 신규 코딩 없이 기존 GPU 가속 소프트웨어를 AMD 환경에서 동작할 수 있게 한다. HCAII에서는 ML 사용자의 부담을 덜기 위해 소프트웨어 스택에 ROCm을 적용하여 시스템을 구축하였다.

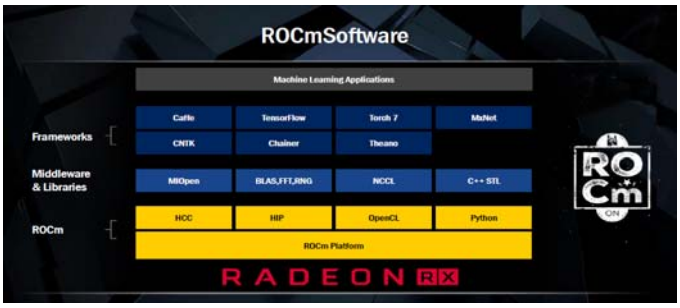


그림 2. ROCm의 소프트웨어 구조[8]

3.2 KV-SSD 기반 HCI 구조

HCAII는 확장성이 높은 AI 인프라를 위해서 스토리지와 서버를 하나로 합친 형태의 HCI 구조를 사용한다. HCAII는 AI 연산에 필요한 대용량 데이터를 분산 스토리지인 Ceph에 저장한다.

Ceph는 내부 스토리지 엔진으로 XFS 기반의[3] FileStore 대신 블록 스토리지에 직접 접근하는 BlueStore로 대체하면서 I/O 성능을 크게 개선하였다[9]. 하지만 오브젝트 형태의 데이터를 블록 스토리지에 저장하기 위해서는 데이터 변환 작업을 처리해야 하는데, 이는 CPU 오버헤드를 유발한다. 특히 HCI 환경에서 파라미터 서버와 같이 CPU를 사용하는 AI 프로세스들과도 경쟁을 유발시켜서 전체 AI 성능을 저하시킬 수 있다.

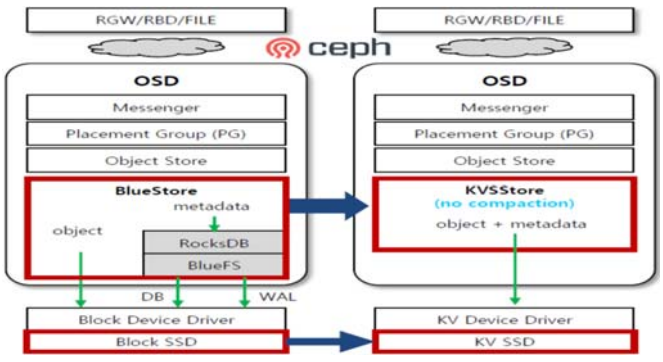


그림 3. SSD와 KV-SSD 기반의 Ceph 구조 비교[9]

위의 문제를 해결하기 위해서 HCAII는 KV-SSD를 저장장치로 적용하였다. KV-SSD는 CPU가 처리하던 블록 연산을 SSD 내부 컨트롤러에서 처리하도록 지원하는 저장장치이다. 기존의 블록 인터페이스 대신 Key-Value 인터페이스를 지원하기 때문에 Ceph의 BlueStore는 사용자에게서 입력된 오브젝트 데이터를 형식의 변환없이

저장할 수 있다. 또한 메타데이터 저장소로 사용되는 RocksDB를 BlueFS 기반에서 Key-Value 인터페이스로 대체함으로써 메타데이터 I/O 오버헤드 또한 크게 감소시킬 수 있다[9]. 기존 하드웨어에서 펌웨어 업데이트만으로 KV-SSD 변환이 가능하기 때문에 비용이 크지 않아 가격 경쟁력을 통한 노드 확장성뿐만 아니라 노드 내의 저장장치 확장성에도 크게 기여할 수 있다.

3.3 HCAII 프로토타입

HCAII는 가성비 높은 AMD GPU 의 클러스터링과 HCI를 위한 KV-SSD 구성을 통해 하드웨어를 구축하였다. 프로토타입 서버 하드웨어는 재설계된 4장의 AMD RX580 카드와 삼성전자의 1TB KV-SSD로 구성된다. 클러스터 시스템의 자원관리를 위해 OpenStack 기반의 HCI 관리 시스템이 이용되었으며 스토리지 시스템은 Ceph가 적용되었다. AMD GPU상에서 구동되는 AI 소프트웨어 플랫폼의 배포 자동화를 위해 ROCm 컨테이너 이미지가 적용되었다[10].

4. 실 험

4.1. AMD vs NVIDIA

본 실험은 동일 환경의 HCAII 서버에서 AMD RX580 및 NVIDIA Tesla V100 GPU를 가격 대비 성능 관점에서 비교하였다. 비교 실험에 사용된 환경은 [표 2]에 명시되어있다.

표 2. 환경 정보

	모델 명칭
CPU	Intel Xeon Silver 4110 (x2)
Main Board	ASUS WS C621E SAGE
RAM	DDR4 16G (x4)
Storage	Samsung 860 EVO 500G (x2)
Cloud OS	OpenStack (Queens)
GPU Library	ROCm (v2.9.6), CUDA (v10.2)

실험은 대표적인 머신 러닝 플랫폼인 TensorFlow 에서 자주 사용되는 CNN (Convolutional Neural Network)들의 벤치마킹을 수행하였다. 본 실험의 상세한 테스트 정보는 [표 3]에 명시하였다. OpenStack을 통해 생성된 VM은 GPU passthrough가 적용되었으며, 4개의 vCPU를 할당하였다.

표 3. 테스트 정보

	선정 항목
Train Data	CIFAR10, ImageNet (synthetic)
Network	ResNet, Alexnet, Inception
Platform	TensorFlow (v1.13)
Bench Tool	tf_cnn_benchmarks.py[11]

테스트는 GPU의 가격 대비 성능을 측정하기 위해 [수식 1]을 정의하였다. 벤치마킹 툴을 통해 GPU의 이미지 처리 성능(images/sec) 을 측정하였고, 그 다음 [5,12] 기준의 GPU 제품 가격을 나누어 단위당 이미지 처리 성능을 얻었다. 아래 정의된 PCE 값이 높을 수록 가격 대비 성능이 높다는 의미를 가지고 있다.

수식 1. 가격 대비 성능 측정

$$Performance\ Cost\ Effectiveness\ (PCE) = \frac{Performance\ (\frac{images}{second})}{Price\ (\$)}$$

표 4. 테스트 결과 (단위: PCE, Batch Size: 64)

	Tesla V100	RX 580
Alexnet	2.0	25.5
ResNet 56	0.48	4.5
Inception3	0.03	0.12

테스트 결과, 동일 GPU 구매 비용으로 AMD GPU에서 Alexnet의 경우 약 12배, ResNet의 경우 약 9배, 더 많은 이미지를 처리할 수 있다는 결과가 나왔다. 단, Inception3에서와 같이 초당 이미지 처리 성능이 낮아질수록 그 차이는 점차 줄어드는 현상을 보였다. [그림 4]는 실험 결과인 [표 4]를 PCE 값의 상대적 비율을 차트로 나타내었다.

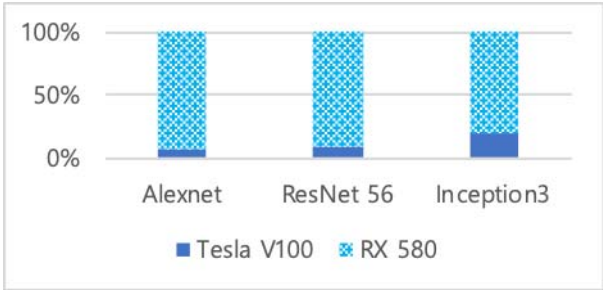


그림 4. 비교 테스트 결과 (PCE 비율)

4.2. SSD vs KV-SSD

HCAII에서 KV-SSD를 도입 시, 스토리지 시스템의 CPU 점유로 인한 AI 학습 성능 영향도 파악을 위해 실험을 진행하였다. 첫번째로 기존의 실험을 참고하여 SSD를 증가시켰을 때의 스토리지 시스템의 CPU 점유율을 확인하였다[13]. 두번째로는 CPU 확보로 인한 성능 개선치를 측정하기 위해 CPU 파라미터 서버를 사용하는 AI training 성능을 측정하였다.

[그림 5]는 스토리지(storage) 스택에서 SSD 개수에 따른 CPU 점유율을 나타낸다. 그래프와 같이 KV-SSD의 경우 SSD를 증가시켜도 점유율이 10%내외로 CPU 자원 확보를 용이하게 한다.

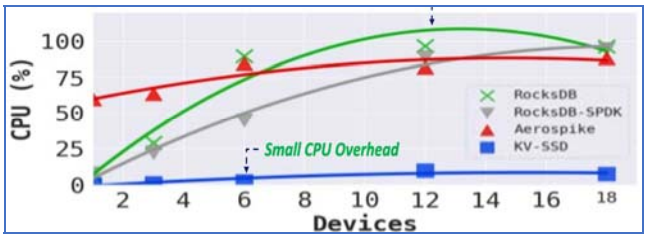


그림 5. SSD 추가에 따른 CPU 점유량[13]

[표 5]은 CPU를 증가시켰을 때의 AI 학습 성능 측정 결과이다. [표 2]의 환경에서 CPU 파라미터 서버를 사용하고, VM당 vCPU 수를 표와 같이 제한한 상태에서 측정하였다.

표 5. AI CPU 제한 테스트 (단위: Img/sec)

	ResNet 56
1 CPU	224
2 CPU	425
4 CPU	802
8 CPU	1157
16 CPU	1155

블록 기반 SSD는 개수가 6개 이상이 되었을 때 CPU 점유율이 약 90~80%에 도달하게 되어 CPU 파라미터 서버를 사용하는 AI 학습 성능에 영향을 줄 수 있다. 그와 달리 KV-SSD는 약 10%~20%로 일정한 CPU 사용률을 보여주기 때문에 스토리지 개수와 상관없이 일관된 성능을 보여주게 된다. 저장장치가 소수인 일반 환경에서는 문제가 되지 않지만, 스토리지 확장성이 요구되는 클라우드 등의 환경에서는 스토리지 시스템의 CPU 점유로 인해 AI 훈련 성능을 하락시킬 수 있다.

HCAII에서는 KV-SSD를 적용하여, 스토리지 시스템의 CPU 점유로 인한 AI 연산의 병목을 완화시켜 GPU를 효율적으로 사용할 수 있게 하였다.

5. 결론

본 논문에서는 스케일 아웃(scale-out) 중심의 GPU HCI 아키텍쳐인 HCAII를 제안하고, 이를 스케일 업(scale-up) 중심의 고사양 NVIDIA 서버 아키텍쳐와 비교하였다.

HCAII는 AMD GPU 및 KV-SSD를 적용하여 높은 가성비를 얻었으며, HCAII 프로토타입을 통한 TensorFlow GPU 실험 결과는 HCAII가 스케일 업(scale-up) 아키텍쳐 대비 Alexnet에서는 12배, ResNet에서 9배 높은 가격 대비 성능을 보임을 확인하였다.

또한 KV-SSD를 적용하여 클라우드와 같이 스토리지 확장성이 요구되는 환경에서도 CPU의 가용성을 보장하여 안정적인 AI 학습 성능을 확보할 수 있다는 결론을 얻었다.

6. 참고 문헌

[1] Information Processing, Blogspot, https://infoproc.blogspot.com/2018/05/exponential-growth-in-compute-used-for.html

[2] Shaikh Abdul Azeem & Satyendra Sharma, Study of Converged Infrastructure & Hyper Converge Infrastructure As Future of Data Centre, International Journal of Advanced Research in Computer Science, 8권, 5호, 901쪽, 2017

[3] Sage A. Weil, CEPH: Reliable, scalable, and high-performance distributed storage, University of California, 9쪽, 2017

[4] Katelin A. Bailey & Peter Hornyack & Luis Ceze & Steven D. Gribble & Henry M. Levy, Exploring Storage Class Memory with Key Value Stores, University of Washington, 7쪽, 2013

[5] NVIDIA Telsa V100 Volta GPU Accelerator 32GB Graphics Card, Amazon, https://www.amazon.com/NVIDIA-Tesla-Volta-Accelerator-Graphics/dp/B07JVNHFFX

[6] Deep Learning GPU Benchmarks - Tesla V100 vs RTX 2080 Ti vs GTX 1080 Ti vs Titan V, Lambdalabs, https://lambdalabs.com/blog/best-gpu-tensorflow-2080-ti-vs-v100-vs-titan-v-vs-1080-ti-benchmark/

[7] Graphics Card Pricing Update: December 2018, Techspot, https://www.techspot.com/article/1772-graphics-card-pricing-dec-2018/

[8] AMD Announces Radeon Instinct: GPU Accelerators for Deep Learning, Coming In 2017, AnandTech, https://www.anandtech.com/show/10905/amd-announces-radeon-instinct-deep-learning-2017/3

[9] Research and Business Opportunities with Key Value SSD, Github, https://github.com/OpenMPDK/KVSSD/wiki/presentation/kvssd_seminar_2018/kvssd_seminar_2018_research_and_business_opportunity.pdf

[10] rocm, DockerHub, https://hub.docker.com/u/rocm/

[11] TensorFlow benchmarks, Github, https://github.com/tensorflow/benchmarks

[12] Sapphire Technology Radeon 11265-05-20G Pulse RX580 8GB GDDR5 Dual HDMI/ DVI-D/ Dual DPOC with Backplate (UEFI) PCI-E Graphics Card, Amazon, https://www.amazon.com/Sapphire-11265-05-20G-Backplate-Graphics-Graphic/dp/B06ZZ6FMF8/ref=sr_1_2?keywords=rx580&qid=1576478274&sr=8-2

[13] Towards Building a High-Performance, Scale-In Key-Value Storage System, Systor, https://www.systor.org/2019/slides/S8P1%20Towards%20Building%20a%20High-performance,%20Scale-in%20Key-value%20Storage%20System.pdf

BERT 기반의 SNS 메시지 불안 분류기 및 사회적 불안 분포 시각화 시스템

송지수○, 최용석
한양대학교 컴퓨터소프트웨어학과
autorsong@hanyang.ac.kr, cys@hanyang.ac.kr

BERT-based Anxiety Classifier for Social Media Messages and Visualization System for Social Anxiety Distribution

Jisoo Song○, Yong Suk Choi
Department of Computer Science, Hanyang University

요 약

본 연구에서는 한국 사회의 불안 감정 분포를 시각화하는 시스템을 구축하기 위해 세계적으로 널리 사용되는 대표적인 SNS 서비스인 트위터(Twitter) 로부터 수집된 대량의 메시지에 대해 각 메시지 별로 불안이라는 감정을 내포하고 있는 지의 여부를 판별하는 감정 분류기를 개발하고, 분류 결과를 시공간 별로 메타데이터화 하여 시각화 할 수 있는 시스템을 제안한다. 자연어 처리 분야에서 매우 좋은 성능을 보이는 언어 모델인 BERT를 기반으로 불안 분류기를 개발하였고, 실험 결과 기타 딥 러닝 기반 언어 모델 기반 분류기 대비 분류의 성능이 우수함을 확인하였다. 또한 시각화 시스템을 성공적으로 구축하여 한국 내 불안 분포를 한 눈에 확인할 수 있도록 했다.

1. 서 론

한국 사회에서 나타나는 높은 자살률, 실업률 및 낮은 고용률 등 다양한 사회적 문제들은 불안으로부터 비롯된 부정적인 심리에 기인한다 [1-3]. 이러한 이유로 한국 사회의 구성원들이 느끼는 불안의 정도 및 추이와 시공간적 분포를 분석하는 것은 사회 문제의 해결과 사회적 복리 향상에 도움이 될 수 있다.

본 논문에서는 사회구성원 개개인의 감정과 생각을 자유롭게 드러낼 수 있는 게시판 역할을 하는 SNS를 이용하여 한국 사회의 불안 감정 분포를 시각화 할 수 있도록 SNS에 게시된 텍스트 메시지의 불안 감정 내포 여부를 판별할 수 있는 감정 분류기 및 메타데이터 시각화 시스템의 구조를 제시한다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 설명한다. 3장에서는 BERT[4]를 기반으로 한 SNS 메시지 대상 감정 분류기 모델의 설계 동기와 구조를 제시한다. 4장에서는 SNS로부터 얻은 빅데이터를 3장의 감정 분류기로 분류하고, 그 결과를 메타데이터화 하는 방법 및 메타데이터를 시각화 할 수 있는 시스템의 설계 동기 및 구조를 제시한다. 마지막으로 5장에서는 결론과 향후 연구 방향을 제시한다.

2. 관련 연구

많은 연구들이 소셜 미디어 데이터에 대한 감성 분석을 수행하기 위해 다양한 분류 방법을 사용해왔다. 딥 러닝 기반 언어 모델이 활발히 사용되기 이전에 규칙 기반 혹은 Naïve Bayes 기반의 언어 모델이 감성 분석에 활용되었다 [5-6]. 딥 러닝 기반의 언어 모델이 좋은 성능을 내기 시작한 이후, 여러 딥 러닝 기반 언어 모델이 감성 분석에도 사용되기 시작하였다. 구글 플레이 서비스에 게시된 리뷰에 대한 감성 분석을 LSTM 기반의 모델을 통해 SVM 혹은 Naïve Bayes 기반의 전통적 인공지능 기법에 기반한 언어 모델보다 18%p 가량의 분류 성능이 있었다는 연구 결과 [7]가 있다.

3. BERT 기반의 SNS 메시지 감정 분류기

이 장에서는 BERT 기반 SNS 메시지 감정 분류기의 구조를 설명하고 실제로 SNS 로부터 얻어진 다량의 텍스트 메시지에 대한 판별 결과를 제시한다.

3.1 SNS 메시지 빅데이터 수집

본 연구에 쓰일 SNS 메시지 빅데이터를 얻기 위하여 국내에서 많은 이용자를 보유하고 있는 트위터(Twitter) 로부터 데이터를 수집하였다. 데이터 크롤링에는 MIT 라이선스 하에 배포되는 Python 용 트위터 크롤러인 'Tweepy' API 를 이용하였다. 2018 년 9 월부터 수집을 시작하여 2020 년 01 월 현재 3,235,935 건의 트위터 메시지를 수집하였다.

트위터 메시지를 수집할 때는 트위터 사에서 부여한 행정구역 (시군구) 별 place_id 값이 존재하는 메시지에 대해서만 수집이 이루어졌다. 따라서 분류 결과에 따라 place_id 값을 활용해 시군구 및 광역시도 별로 지역별 불안 지수를 산출하고 시각화를 진행하였다.

3.2 BERT 를 이용한 텍스트 분류

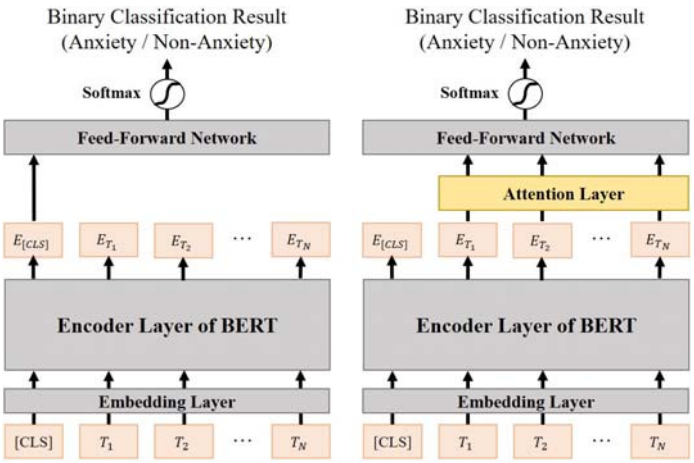
BERT 모델은 Attention Mechanism [8]를 기초로 하는 Transformer 모델을 기반으로 만들어진 언어 모델이다. Transformer 모델은 인코더와 디코더로 이루어지는데 BERT 는 다량의 텍스트 코퍼스를 이용하여 학습을 한 후 인코더 부분을 다층으로 쌓아 세부 태스크와 학습 데이터에 맞게 적은 횟수의 Fine-tuning 을 거쳐 사용할 수 있다.

BERT 는 인코더의 적층 개수에 따라 BERT-base 모델과 BERT-large 모델로 나뉘어지는데, BERT-base 모델은 총 12 개의 인코더를 쌓은 모델이다. 본 연구에서는 한국어 코퍼스로 학습한 BERT-base 모델을 이용하였다.

BERT 모델은 각 토큰마다 개별적으로 벡터 형태의 은닉 표현(Hidden Representation)을 생성하는데, 입력된 시퀀스의 가장 첫 부분에 '[CLS]' 라는 특수 토큰을 삽입한다. '[CLS]' 토큰의 은닉 표현은 문장 전체의 맥락적 의미를 함축하도록 학습이 된다. BERT 를 이용한 텍스트 분류는 이 '[CLS]' 토큰의 최종 은닉 표현 벡터를 일반적인 FFN(Feed-Forward Network)에 통과시켜 수행한다.

3.3 분류 레이어에서의 Attention Mechanism

본 연구에서는 '[CLS]' 토큰의 은닉 표현을 통한 분류 대신, BERT 의 인코더 층을 통과한 후 Attention Layer 를 추가하여 분류를 수행하는 방식으로 언어 모델을 튜닝하였다. 제안하는 모델 구조의 도식은 [그림 1]의 우측과 같다.



[그림 1] 기존 BERT의 구조 (좌), Attention Layer를 추가한 BERT의 구조(우)

'[CLS]' 토큰 대신 문장에 쓰인 전체 토큰의 은닉 표현을 Attention Layer에 통과시켜 각 토큰에 대한 Alpha 값을 얻는다. Alpha 값은 각 토큰의 은닉 표현이 분류를 위해 FFN으로 들어갈 때 특정 토큰의 은닉 표현 값에 얼마만큼의 가중치를 줄 것인지 결정하는 역할을 한다. 즉 각 토큰마다 Alpha 값은 모두 다르며 문장 속에서 i번째 토큰의 Alpha 값 α_i는 다음과 같이 계산된다:

$$\alpha_i = \frac{\exp(S_i \cdot W)}{\sum_{j=1}^{n} \exp(S_j \cdot W)}$$

이 Alpha 값이 큰 토큰일수록 해당 토큰의 은닉 표현이 더욱 큰 가중치를 가진 채로 FFN에 들어가며, 따라서 이 Alpha 값은 곧 해당 토큰이 감정 분류에 영향을 끼친 정도를 의미하게 된다.

Alpha 값을 계산한 후, 각 토큰의 Alpha 값은 은닉 표현 벡터 값과 reduce sum을 거쳐 최종적으로 Attention Layer의 Output을 결정하게 된다. 해당 Output은 FFN을 거쳐 이진 분류를 수행하게 된다.

3.4 분류 실험 결과

	Anxiety Class			Macro F1-score	Accuracy
	Precision	Recall	F1-score		
Naïve Bayes	**0.4619**	0.1841	0.2633	0.5878	0.8432
Bi-RNN [9]	0.1746	**0.8070**	0.2817	0.5702	0.7568
Multi-Channel CNN [10]	0.3031	0.4859	0.3716	0.6587	0.9003
BERT (1)	0.2017	0.6958	0.3128	0.6028	0.8145
BERT (2)	0.4114	0.3817	**0.3869**	**0.6740**	**0.9266**

[표 1] 여러 언어 모델 기반의 분류기를 통한 분류 실험 결과

본 연구에서는 BERT 이외에도 통계적인 알고리즘 및 다양한 딥러닝 알고리즘에 기반한 언어 모델을 이용해 SNS 메시지 감정 분류기를 훈련시켜 성능을 비교하였다. 표 1의 BERT (1) 항목은 3.2절에서 언급한 '[CLS]' 토큰만을 이용하여 분류를 시행한 결과이며, 표 1의 BERT (2)는 3.3절에서 제시하고 있는 분류 과정에서 BERT 모델의 인코더를 거친 후 Attention Layer를 추가한 방식으로 분류를 시행한 결과이다.

테스트 데이터 중 불안으로 분류된 트윗에 관하여 Precision이 제일 높은 분류기는 Naïve Bayes 기반, Recall이 제일 높은 분류기는 Bi-RNN 기반의 분류기였다. 그러나 Precision과 Recall의 값이 동시에 얼마나 높은지를 가리는 평가 지표인 F1-score에서는 BERT (2) 항목이 제일 높았으며 정확도 또한 제일 높은 성능을 기록하였다.

크롤링 된 전체 트위터 메시지 중 78,022 건을 무작위로 추출해 그 중 70%를 훈련 데이터로, 15%를 검증 데이터로, 15%를 테스트 데이터로 지정해 불안 분류기의 훈련에 사용했다. 훈련 데이터의 불안/비불안 태깅은 불안의 정의에

따라 수동으로 이루어졌는데, 오직 4,729 건 (약 6.5%)의 메시지만이 불안 감정을 포함하고 있다고 태깅되었다. 비불안과 불안 태깅의 비율이 약 14.4:1에 이르는 극심한 데이터 불균형으로 인해 이진 분류 모델임에도 분류 성능 평가 기준값이 전체적으로 0.4 부근에서 형성되어 있다 이는 통상적인 성공적 이진 분류 모델의 분류 성능 [11]을 달성하기 어렵게 한다.

4. 감정 분포 시각화 시스템

이 장에서는 BERT 기반의 SNS 메시지 감정 분류기를 이용하여 각 SNS 메시지에 대한 불안 감정 내포 여부를 기간별 및 행정구역별의 시공간 축으로 메타데이터화하는 방법 및 산출된 메타데이터를 시각화하는 시스템의 구조와 구현 결과를 제시한다. 본 시스템은 다음의 웹 사이트 링크 (http://166.104.143.105:62000/main)를 통해 접속 가능하다.

4.1 불안지수의 산출

본 연구에서는 SNS 메시지에 대한 불안 감정 내포 여부 판별 결과를 시공간 축으로 메타데이터화하기 위하여 판별 결과를 바탕으로 불안지수를 정의하였다. 본 연구에서 사용된 불안지수로는 절대불안지수와 상대불안지수 두 종류가 있다. 절대불안지수 $f_{absolute}$ 의 값은 간단히 특정 지역에서 특정 기간 내에 게시된 모든 트윗의 개수 대비 불안 감정을 내포하고 있다고 분류된 트윗의 비율로 계산한다. 상대불안지수 $f_{relative}$ 의 값은 동일 기간 중 절대불안지수의 전국 평균 수치로 정규화 한 값으로 다음과 같이 계산된다:

$$f_{relative} = \frac{f_{absolute} - \bar{f}}{\sigma}$$

$\bar{f}$ 는 전국 절대불안지수의 평균값이며 σ 는 전국 절대불안지수의 표준편차이다. 두 가지 불안지수 모두 행정구역(광역시도별, 시군구별)에 따라 매주 산출된다.

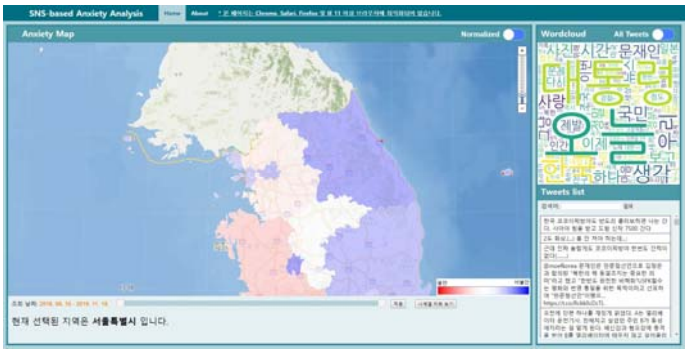
4.2 빈출 단어 추출

SNS 메시지를 분석함에 있어 감정 분류기를 통해 얻어진 불안 감정에 대한 정보 이외에도 SNS 상에서의 트렌드 혹은 화제가 되는 사회적 이슈 등에 대한 정보를 얻기 위해 SNS 메시지 텍스트로부터 빈출 단어를 추출하였다.

한 글자로 이루어진 단어와 명사 이외의 품사인 단어들은 제외하고 빈출 단어를 추출하며, 불안지수와 마찬가지로 각 행정구역별로 매주 산출된다.

4.3 감정 분포 시각화 시스템

본 연구에서는 상기에 언급한 불안지수와 빈출 단어 등의 메타데이터를 한 눈에 볼 수 있도록 시각화 시스템을 설계 및 개발했다. 본 시스템은 크게 불안 지도, WordCloud, 메시지 리스트의 세 가지 기능을 포함하고 있다. 이 세 가지 기능은 서로 상호작용을 하며 선택된 기간과 지역에 따라 그에 맞는 정보를 제시한다.



[그림 2] 감정 분포 시각화 시스템 메인 페이지의 모습

4.3.1 불안 지도

[그림 3 (좌)] 상대불안지수 표시 모드인 불안 지도 예시
[그림 4 (우)] 절대불안지수 표시 모드인 불안 지도 예시

불안 지도 기능은 사용자가 선택한 기간 및 지역에 따라 절대불안지수 혹은 상대불안지수를 지도상에 표시해 주는 기능이다. 두 종류의 불안지수 중 어떤 것을 표시할 지는 상단의 토글 버튼을 통해 변경할 수 있으며, 축척 조절을 통해 광역시도 및 시군구 간의 행정구역 표시 범위도 변경할 수 있다.

상대불안지수 표시 모드에서는 상대불안지수의 값에 따라 개별 지역의 색상이 빨간색부터 파란색의 범위 안에서 정해지게 된다. 특정 지역의 상대불안지수가 양수이면 빨강 계열의 색으로, 음수이면 파랑 계열의 색으로 표시가 된다.

절대불안지수 표시 모드에서는 절대불안지수의 값에 따라 흰색부터 빨간색의 범위 안에서 표현이 된다. 특정 지역의 절대불안지수가 높을수록 빨간색에 가깝게 표시가 된다.

4.3.2 WordCloud

WordCloud는 선택된 기간과 지역에서 게시된 SNS 메시지 중 자주 사용된 단어를 시각화하여 보여주는 기능으로 [그림 2의 우상단에 위치하고 있다. 자주 사용된 단어일수록 크기가

크게 나타난다. 상단의 토글 버튼을 클릭하면 빈출 단어를 추출할 대상이 되는 SNS 메시지의 범위를 선택할 수 있다. 전체 메시지 모드는 수집된 모든 메시지로부터 빈출 단어를 추출하여 보여주는 모드이며, 불안 메시지 모드는 불안 감정을 내포하고 있는 것으로 분류된 메시지로부터 빈출 단어를 추출하여 보여주는 모드이다.

4.3.3 메시지 리스트

[그림 5(좌)] 전체 메시지 모드인 메시지 리스트 예시
[그림 6(우)] 불안 메시지 모드인 메시지 리스트 예시

메시지 리스트 기능은 상기에 언급된 불안지수 및 빈출 단어 등의 메타데이터들을 산출하는 데 기초 근거가 되는 원본 SNS 메시지를 확인할 수 있는 기능이다.

이 곳에 표시되는 메시지는 WordCloud의 빈출 단어 추출 대상 범위에 따라 달라진다. WordCloud가 전체 메시지 모드라면 메시지 리스트 또한 선택된 기간과 지역에서 게시된 모든 메시지를 보여주며, 불안 메시지 모드라면 불안 감정을 내포하고 있는 것으로 분류된 메시지만을 보여준다.

상기 3.3절에서 언급한 Alpha 값은 분류에 큰 영향을 미치는 토큰일수록 높은 수치를 갖는다는 점에서 착안하여, 불안 감정을 내포하고 있다고 분류된 SNS 메시지를 이루고 있는 각 토큰의 Alpha 값을 이용해 어떤 토큰이 불안으로의 분류에 영향을 끼쳤는지 한 눈에 확인할 수 있는 기능을 탑재했다. 불안 메시지 모드일 경우 Alpha 값이 높은 토큰에 빨강 계열의 색으로 표시가 되며, Alpha 값이 높을수록 빨간색에 가까운 색으로 표시가 되도록 설계하였다.

5. 결론 및 향후 연구계획

본 연구에서는 BERT 기반 SNS 메시지 감정 분류기를 개발하여 타 언어 모델 대비 분류의 성능을 향상시켰으며, Attention Mechanism의 응용을 통해 불안 감정 내포 여부 분류에 영향을 끼친 단어를 시각화할 수 있는 근거를 제공했다. 또한 분류 결과 및 기타 데이터들을 한 눈에 확인할 수 있는 시각화 시스템을 개발하였다. 추후 분류기의 성능을 더욱 높이기 위하여 언어 모델의 튜닝, 한국어의 특성에 맞는 전처리 기법, 데이터 불균형 해결 기법 등을 연구할 계획이다.

Acknowledgements

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.2018R1A5A7059549).

6. 참고 문헌

[1] Nasreen, Hashima E., et al. Low birth weight in offspring of women with depressive and anxiety symptoms during pregnancy. results from a population based study in Bangladesh. BMC public health, 10.1, 515, 2010.

[2] Montgomery, Scott M., et al. Unemployment pre-dates symptoms of depression and anxiety resulting in medical consultation in young men. International Journal of Epidemiology, 28.1, 95-100, 1999.

[3] Brader, Ted, George E. Marcus, and Kristyn L. Miller. Emotion and public opinion. The Oxford Handbook of American Public Opinion and the Media, 2011.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805, 2018.

[5] Hutto, Clayton J., and Eric Gilbert. "Vader: A parsimonious rule-based model for sentiment analysis of social media text." Eighth international AAAI conference on weblogs and social media. 2014.

[6] Kharde, Vishal, and Prof Sonawane. "Sentiment analysis of twitter data: a survey of techniques." arXiv preprint arXiv:1601.06971 (2016).

[7] Day, Min-Yuh, and Yue-Da Lin. "Deep learning for sentiment analysis on google play consumer review." 2017 IEEE international conference on information reuse and integration (IRI). IEEE, 2017.

[8] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In International Conference on Learning Representations, 2017.

[9] Mike Schuster, Kuldip K. Paliwal. Bidirectional Recurrent Neural Networks. IEEE Transactions on Signal Processing, 45(11), 1997.

[10] 김민, 변증현, 이충희, 이연수. Multi-channel CNN을 이용한 한국어 감성분석. 제30회 한글 및 한국어 정보처리 학술대회 논문집, 2018.

[11] Zimmermann, Thomas, et al. "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process." Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. 2009.

그룹 리더 선출을 통한 비트코인 병렬 트랜잭션 처리

이도현[0], 송아람, 이선재, 박우람, 박수용
서강대학교
cailhuiris@gmail.com[0], songaram0106@gmail.com, sunjae042@sogang.ac.kr,
dnfka3020@gmail.com, sypark@sogang.ac.kr

Group Leaders Election and Bitcoin Parallel Transaction Processing

Do-Hyeon Lee[0], A-Ram Song, Sun-Jae Lee, Woo-Ram Park, Soo-Yong Park
Sogang University

요 약

비트코인은 탈중앙화 시스템인 블록체인을 기반으로 중앙은행의 개입 없이도 디지털 송금을 가능하게 한 전자화폐이다. 작업증명 기반의 합의 알고리즘을 사용하는 비트코인에서는 전체 네트워크의 노드들이 해시-퍼즐 문제를 풀며 가장 먼저 문제를 푼 단 하나의 노드를 선출하고 선출된 노드가 블록을 생성하여 네트워크로 전파한다. 따라서 전체 네트워크에서 단 하나의 노드만 데이터를 생성한다는 점에서 성능을 포기하고 비잔틴 노드의 네트워크 점유를 막는다. 하지만, 이 합의 알고리즘은 전체 시스템의 거래 신뢰도는 높이지만 거래 처리 능력을 급격하게 저하시키는 한계가 있다. 본 논문에서는 블록체인 네트워크에서 발생하는 트랜잭션을 병렬로 처리하기 위해 네트워크의 노드를 일정 단위로 그룹화하여 다수의 리더를 선출하되 시스템의 신뢰성은 포기하지 않는 모델을 제시한다.

1. 서 론

비트코인은 블록체인을 활용한 디지털 화폐로서 모든 거래 트랜잭션이 피어-투-피어 네트워크 기술을 통해 공통 장부에 기록하여 다수가 유지한다. 이를 통해 서로 신뢰할 수 없는 네트워크 상황에서 무결성 있는 데이터를 유지가 가능해지게 되었다.[1]

2018년부터 비트코인의 거래량이 폭발적으로 증가하게 되면서 비트코인 프로토콜의 확장성 문제가 주목받기 시작했다. 비트코인 프로토콜에서 사용하고 있는 작업증명 합의 알고리즘은 네트워크를 유지하고 있는 다수의 노드 중 단 하나의 노드를 해시 퍼즐을 푸는 방식으로 선출한다. 선출된 노드만이 거래를 원장에 기록할 수 있기 때문에 노드의 수가 증가하는 것과 무관하게 일정한 처리 속도를 유지한다.

2019년 현재 비트코인을 유지하는 전체 네트워크는 약 9000여 개 이지만 이 중 단 하나의 노드만 장부에 거래를 기록한다. 이러한 방식을 통해 데이터의 무결성을 유지하고 높은 신뢰성을 제공할 수 있지만 이러한 방식은 극단적으로 시스템의 성능을 저하시키는 한계를 내재한다.

이를 해결하기 위해 블록당 처리량(Throughput)과 블록 생성 시간(Interval)을 줄이는 방법이 소개되고 있지만, 이 두 해결방안은 프로토콜의 신뢰성을 저하 시킨다는 단점이 있다.

본 논문에서는 블록을 생성하기 위한 노드를 다중으로 선출

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음"
(IITP-2019-2017-0-01628*)

하여 블록을 병렬로 생성함으로써 전체 네트워크의 성능을 높이되 블록체인의 신뢰도는 기존과 동일하게 유지하는 방법을 제안한다.

2. 배경지식 및 관련 연구

가. 블록체인

블록체인을 구현하는 기술인 비트코인은 2008년 Satoshi Nakamoto에 의해 처음으로 소개되었다. 별도의 중앙기관이 필요 없는 Peer-to-Peer(P2P) 네트워크로 구성되어 있어 누구나 네트워크에 참여 가능한 개방성을 가지고 있다. 네트워크로 참여한 모든 구성원은 합의 과정에 참여하여 분산원장인 블록체인에 거래를 기록하고 모든 참여자에게 전파하여 모든 노드가 동일한 거래 기록을 가지고 있음으로써 부인방지 및 이중-지불 문제를 해결하였다.

나. 작업증명 알고리즘

작업증명(Proof-of-work) 알고리즘은 블록체인 네트워크의 각 참여자가 동일한 난이도의 해시-퍼즐 문제를 풀면서 가장 먼저 해답을 찾아낸 참여자의 블록이 블록체인 데이터베이스에 기록되는 방식으로 동작한다.

각 참여자는 동일한 해시 함수 H를 이용하고 이전 블록 헤더, 타임스탬프 등의 값 y와 nonce로 불리는 무작위 값 x가 특정 target보다 작을 때까지 x의 값을 조정한다. 아래 식을 가장 먼저 만족하는 x를 찾은 노드가 블록의 생성 권한을 갖는다.

$$H(y|x) < \text{some target value}$$

비트코인은 작업증명 알고리즘을 통해 하나의 노드가 블록체인 공유 원장을 점유하는 것을 방지한다.

다. UTXO

비트코인은 트랜잭션을 발생할 때마다 그 결과로 UTXO를 생성한다. UTXO는 Unspent Transaction Output으로 송금되지 않은 트랜잭션 객체를 의미한다. 이 UTXO는 다음 트랜잭션의 Input으로 들어가고 개인 키 서명을 통해 송금한다. 그리고 이 트랜잭션의 결과로 또 다른 UTXO가 생성된다. 아래 그림 1. 은 UTXO 모델에서 송금이 어떻게 발생하는지 도식화한 그림이다.

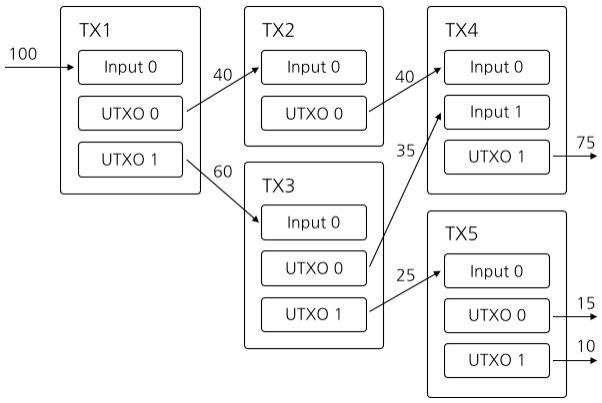


그림 1. UTXO 모델

한 트랜잭션(TX)은 M개의 Input과 N개의 UTXO로 이루어져 있다. UTXO 모델은 비트코인에서 디지털 화폐를 추상화한 객체 단위이다. 각 UTXO는 소유권을 증명하기 위한 서명 값이 포함되어 있으며 UTXO의 소유권자는 UTXO의 소유권을 다른 이의 공개키로 서명 함으로써 전송할 수 있다. 비트코인은 디지털 화폐의 송금 과정을 소유권의 이전으로 해석한다.

소유하고 있는 디지털 화폐를 송금하려는 사람은 자신의 비밀 키로 서명된 UTXO 셋을 트랜잭션의 Input으로 입력하고 받는 이의 공개 키로 서명한 새로운 UTXO를 생성한다. 이때 새로 생성된 UTXO의 총합과 Input의 총합이 서로 같아야 정당한 트랜잭션이 된다.

비트코인에서는 동일한 UTXO가 서로 다른 분기에서 생성된 블록이 동시에 거래 기록으로 인정되는 상황을 이중-지불 문제라고 부른다. 동일한 UTXO가 트랜잭션에 포함되어 있다면 사용자는 동일한 디지털 화폐를 두 번 사용한 것이 된다.

본 논문에서는 특정 시간 t에서 생성된 두 개의 블록이 포함하는 UTXO가 서로 중복되지 않으면 이중-지불이 발생하지 않으므로 두 블록을 모두 기록한다는 개념으로 출발한다. 자세한 내용은 [3. 제안 모델]에서 다룬다.

라. 확장성 문제

블록체인 프로토콜은 그 잠재력에도 불구하고 확장성 문제에 직면해있다. 블록체인 프로토콜은 두 가지 파라미터를 조정함으로써 네트워크의 성능을 높일 수 있는데 하나는 블록 크기이고 다른 하나는 블록 생성 주기이다. 비트코인은 한 블록에 저장될 수 있는 트랜잭션을 제한하기 위해 최대 크기를 지정하는데 이를 블록 크기라고 부른다. 이 블록 크기를 늘리면 한 블록에 포함되는 거래량을 높일 수 있는 대신 블록이 다른 네트워크로 전파되는 시간을 지연시키기 때문에 분기가 발생할 가능성이 커진다[2].

블록 크기를 그대로 유지하는 대신 블록 생성 주기를 줄이는 방안이 있지만, 생성 주기를 줄이기 위해서는 작업증명-합의에서 이용하는 해시-퍼즐의 난이도를 줄여야 한다. 이는 해시 파워가 높은 이용자가 블록 생성 권한을 독점할 위험을 높여준다.

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음"
(IITP-2019-2017-0-01628*)

마. 비트코인-NG

Ittay Eyal 등이 제안한 비트코인-NG에서는 비트코인의 블록을 리더 선출을 위한 Key Block과 거래내역 기록을 위한 Micro Block으로 분리하여 확장성 문제를 해결하였다[3]. Key Block은 기존 작업증명-합의 방식과 동일하게 해시-퍼즐 먼저 해결한 노드가 생성하며 리더가 되고 리더로 선출된 노드가 다음 리더가 나타날 때까지 Micro Block을 생성함으로써 블록체인의 성능을 높이는 방안을 제시하였다. 하지만 이 모델에서 최대 성능은 노드 한 대가 낼 수 있는 성능에 수렴하기 때문에 여전히 전체 네트워크의 기대 성능보다 낮은 성능을 낸다는 문제가 있다.

바. Sharding

이더리움을 기반으로 네트워크를 샤드라는 단위로 그룹화하고 샤드별로 독립된 블록체인을 유지하는 방법으로 성능을 높이는 Sharding에 관한 연구가 많이 진행되고 있다[4][5]. 하지만, 이 방법에는 샤드별로 인센티브를 어떻게 분배할 것인지의 문제가 존재하고 기존 Proof-of-Work 합의 알고리즘의 대안 없이 인위적으로 네트워크를 분할하면 각 샤드별로 해시-퍼즐의 난이도를 낮추어야 블록이 일정 주기로 생성되기 때문에 샤드별로 보안 문제가 발생한다.

3. 제안 모델

본 논문에서는 비트코인-NG의 모델을 확장하여 네트워크 안에서 리더 그룹을 조직화하고 다수의 리더를 선출하여 트랜잭션을 병렬로 처리하는 방식으로 확장성을 높이는 모델을 제안한다.

3-1. Key Block

Key Block은 리더를 선출하기 위해서 생성하는 블록으로 모든 참여자는 해시-퍼즐 문제를 통해 리더로 선출되기 위한 경쟁을 한다. Key Block에는 블록 해시, 이전 블록 헤더, 타임스탬프 그리고 리더 그룹의 Public Key 집합이 저장된다. [표1] 은 Key Block에 포함되는 값을 나열한다.

파라미터	정의
블록 높이	현재 블록이 생성된 높이
블록 해시	블록에 포함된 모든 값을 SHA256으로 해시한 값
이전 블록 헤더	이전에 생성된 모든 마이크로 블록의 헤더 집합
타임스탬프	블록이 생성된 시간 (초 단위)
리더 그룹 공개키	리더 그룹의 공개키 집합
서명	해당 블록을 생성한 리더의 개인 키로 서명한 값
nonce	해시 퍼즐 문제의 정답
난이도	해시 퍼즐 문제의 난이도 값

[표1] Key Block의 블록 구조

리더 그룹의 공개키 집합에는 K개의 공개 키만 저장될 수 있으며 모든 노드는 연결된 노드 중 무작위로 K개의 노드를 선별하여 리더 그룹 공개키 집합에 입력한다. 리더 그룹의 크기를 결정하는 K의 값은 블록체인의 네트워크의 규모에 따라 다르게 결정되어야 한다. K의 값이 크면 클수록 Parallel Micro Block이 많이 생김에 따라 네트워크 통신이 많아지기 때문에 부하가 발생한다.

3-2. Parallel Micro Block

Key Block이 생성되면 해당 블록 안에 포함된 리더 그룹은 Parallel Micro Block(마이크로 블록)이라고 불리는 작은 단위의 블록을 다음 Key Block이 발견될 때까지 생성하고 전파할 수 있는 권한을 가진다. 이때 선출된 리더는 새로운 Key Block을 생성하기 위해 해시-퍼즐을 풀거나 트랜잭션 풀에서 정보를 가져와 거래내역을 마이크로 블록에 기록하고 전파하는 두 가지 선택 중 하나를 수행할 수 있다.

마이크로 블록은 K개의 노드가 동시에 생성하기 때문에 블록에 포함되는 트랜잭션의 UTXO는 중복이 되면 안 된다. 또한 마이크로 블록은 전파를 빠르게 하기 위해 블록 크기를 작게 설정해야 한다. 만약 마이크로 블록이 전체 네트워크에 전파되기 전에 새로운 리더가 선출된다면 해당 블록은 포함되지 않을 수 있다.

[표2]는 마이크로 블록에 포함되는 데이터를 나열한다.

파라미터	정의
블록 높이	현재 마이크로 블록이 생성된 높이
블록 인덱스	다른 마이크로 블록과 구별하기 위한 인덱스 값
이전 블록 헤더	이전에 생성된 블록의 해시 값
블록 해시	현재 블록의 모든 값을 해시한 값
트랜잭션	마이크로 블록에 포함되는 트랜잭션 집합
서명	선출된 리더의 개인 키로 서명한 값

[표2] Parallel Micro Block의 블록 구조

그림 2. 는 Key Block과 마이크로 블록을 이용한 블록체인 데이터 구조를 나타낸다.

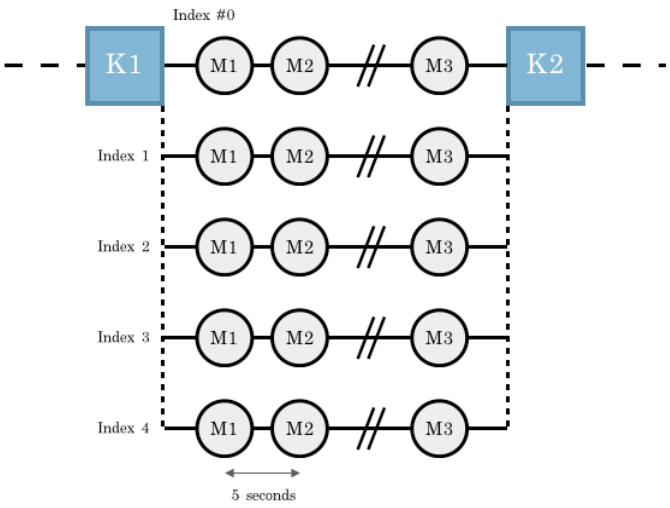


그림 2. 제안 모델의 블록체인 구조

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음" (IITP-2019-2017-0-01628*)

그림 2. 에서 K1과 K2는 리더 그룹을 선출하기 위한 Key Block을 나타낸다. Key Block 사이에서 생성된 { M1, M2 … M3 } 집합은 마이크로 블록을 나타내며 선출된 리더들이 자신이 권한을 가진 Index Slot에 마이크로 블록을 다음 Key Block인 K2가 생성될 때까지 생성한다.

위 그림에서는 K1 블록에서 선출된 5개의 리더가 병렬로 마이크로 블록을 생성하는 구조를 나타내고 있다. Key Block에 포함된 리더 그룹 공개키 집합 중 자신의 공개 키의 순서에 따라 Index Slot이 결정되며 결정된 Index Slot에만 마이크로 블록을 생성할 수 있다.

자신의 Index Slot은 그룹 공개키 집합을 오름차순으로 정렬한 결과에 자신의 공개키가 위치한 번호로 결정된다.

3-3. Transaction Choice Rule

그림 1. 과 같이 마이크로 블록을 동시에 생성하려면 각 Index Slot에 들어가는 트랜잭션의 UTXO가 중복되면 안 된다. 리더로 선출된 노드는 트랜잭션 풀 안에 존재하는 트랜잭션 중 일부분을 꺼내어 마이크로 블록에 포함한 뒤 전파해야 하는데 이 때 다른 리더와 동일한 트랜잭션을 블록에 기록하지 않기 위해 트랜잭션 선택 규칙을 따라야 한다.

본 논문에서는 트랜잭션 해시의 마지막 10개의 비트를 리더 그룹 크기로 나머지 연산하여 나온 값과 권한을 가진 Index Slot의 Index 값과 일치한 경우에 해당 트랜잭션을 마이크로 블록에 포함하는 트랜잭션 선택 규칙을 따른다.

3-4. Incentives

본 논문에서 제안하는 모델의 블록체인은 블록 생성 보상은 Key Block을 생성한 단 하나의 노드에게 제공한다. 따라서 마이크로 블록을 생성한 리더에게 보상을 제공해야 하는데 만약 보상이 주어지지 않는다면 피어-투-피어 네트워크 특성상 아무도 마이크로 블록을 생성하지 않을 것이다.

따라서, 본 논문에서는 트랜잭션 송금 수수료를 부과하여 마이크로 블록을 생성할 때 포함된 트랜잭션의 수수료는 해당 블록을 생성한 리더에게 전송하는 것을 제안한다. 이렇게 함으로써 선출된 리더는 확률이 낮은 새로운 Key Block을 생성하는데 컴퓨팅 파워를 사용하기보다 확실한 보상이 주어지는 마이크로 블록 생성을 하는데 동기가 더 클 것이다.

4. 결론

본 논문에서는 작업증명-합의 알고리즘을 리더 선출과 거래기록 생성이라는 두 가지 관점으로 분리하여 성능을 높인 비트코인-NG을 확장하여 Key Block에서 다수의 리더를 선출하여 블록을 병렬로 생성할 수 있는 방안을 제시하였다.

다만, 비교적 병렬로 처리할 수 있는 트랜잭션들을 식별하기 쉬운 ‘송금’트랜잭션으로 주제를 제한하였는데 스마트 계약이 존재하는 이더리움에서는 UTXO모델을 사용하지 않고 이전 사건에 종속되는 트랜잭션이 다수 존재하기 때문에 이를 효과적으로 식별할 수 있는 알고리즘이 필요하다. 후속 연구에서는 이를 스마트 계약 기능이 포함된 블록체인 플랫폼으로 확장하여 연구를 진행할 것이다.

5. 참고 문헌

[1] Satoshi Nakamoto. 비트코인: A peer-to-peer electronic cash. system. http://www.비트코인.org/비트코인.pdf. 2008.

[2] Yonatan Sompolinsky. AND Aviv Zohar. Accelerating 비트코인’s. Transaction Processing Fast Money Grows on Trees, Not Chains. In Financial Cryptography. 2015.

[3] Ittay Eyal, Adem Efe Gencer, Emin Gun Sirer and Robbert van. Renesse. 비트코인-NG: A Scalable Blockchain Protocol. (NSDI’16). ISBN 978-1-931971-29-4. 2016.

[4] Vitalik Buterin. A next generation smart contract & decentralized application platform. https://www.ethereum.org/pdfs/EthereumWhitePaper.pdf/. 2015.

[5] Mahdi Zamani, Mahnush Movahedi, Mariana Raykova. RapidChain: Scaling Blockchain via Full Sharading. (CCS’18, 2018 ACM SIGSAC Conference on Computer and Communications Security). ISBN 978-1-4503-5693-0. 2018.

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음" (IITP-2019-2017-0-01628*)

계약 관련 음성 녹취 정보의 신뢰성 있는 저장과 체계적 관리를 위한 블록체인 기반 시스템 구성 방식

김인근 ○, 서중원, Alshiri Saad, 장민오, 이유정, 박수용
서강대학교 컴퓨터공학과
Ingeun92@naver.com, jungwonrs@gmail.com, saaad77.sa@gmail.com,
minoh2842@gmail.com, lkitty0302@gmail.com, sypark@sogang.ac.kr

Blockchain-based system configuration for reliable storage and systematic management of contract-related voice recording information

Ingeun Kim○, Jung Won Seo, Alshiri Saad, Minoh Jang, You Jeong Lee, Sooyong Park
Department of Computer Science and Engineering, Sogang University

요 약

최근 스마트폰과 같은 개인용 휴대전화의 광범위한 보급에 힘입어 텔레마케팅 시장은 계속 확대되고 있다. 외부적인 산업 발전에 더불어 소비자의 권익도 보호하기 위해 현재 많은 텔레마케팅 과정에서 텔레마케터와 소비자 간의 대화 및 약관 설명 등의 내용을 녹취하여 저장해 둔다. 그리고 이 저장된 녹취 정보로 분쟁이 발생했을 때 확인을 해볼 수 있고 때때로 감사(audit)를 통해 텔레마케팅이 제대로 이루어지고 있는지 또한 확인할 수 있다. 그러나 이렇게 저장된 녹취 정보에 악의적인 수정이나 편집이 있다면 소비자의 권익이 침해당할 여지가 충분하다. 또한 감사 과정에서 소비되는 비용과 시간 때문에 이런 점을 예방하기 위한 활동에도 한계점이 존재하게 된다. 본 논문에서는 기존의 텔레마케팅 산업에서 가장 비효율적이었던 계약 시 음성 정보의 저장과 조회 및 감사 시 음성 정보를 사용해야 한다는 문제점을 해결하고 이 때의 해당 정보의 신뢰성 제고를 위한 방법을 제시한다. 텔레마케팅 계약 시 나오는 모든 음성 정보를 텍스트 정보로 바꾸어 저장하고 조회 및 감사 등 내용의 확인이 필요할 때 텍스트 내용으로만 가능하도록 하고 이 때 나올 수 있는 텍스트 정보에 대한 신뢰성 문제는 블록체인 상에 텍스트 내용과 음성에서 텍스트로 변환 시 정보들(Parameters)을 함께 저장하여 해결한다.

1. 서 론

텔레마케팅(TM)은 통신 기술의 발전과 휴대전화 보급 확대로 과거 상품에 대한 직접 판매나 방문 판매에 국한되어 있던 판매 전략에서 벗어나 더 쉽고 빠르게 상품을 판매할 수 있게 만들어 주었다. 이러한 장점 때문에 TM은 기업 사이에서 빠르게 퍼져 나갔고 이로 인해 약관이나 보험계약 시 중요한 내용을 고지하지 않는 등의 불완전 판매 피해 사례도 늘어나 TM의 소비자 피해 중 상당부분을 차지하게 되었다.[1] 위와 같은 피해를 막기 위해 정부에서 TM 시 오고 가는 음성 내용을 녹취하여 저장하게 하고 매월 판매가 완료된 계약의 20%는 녹취 파일을 확인하여 “표준상품설명대본”에 따라 상품에 대해 설명했는지 감사를 하게 된다.[2] 이를 통해 불완전판매 여부를 확인하게 된다. 그러나 중앙 서버 DB에 저장되는 기존 TM의 음성 정보 저장 방식에는 조작 가능성과 더불어 녹취 내용 확인에 많은 시간과 비용이 들어간다는 문제점이 존재한다. 이러한 문제점을 본 논문에서는 블록체인에 기반한 분산 원장에 녹취 정보를 저장하여 블록체인의 무결성을 바탕으로 해당 정보에 대한 보안성을 높이는 방법을 소개할 것이다. 그리고 블록체인 내에 저장을 할 때 음성인식 기술을 이용하여 음성 정보를 텍스트 정보로 바꾸어 저장해서 효율적으로 저장 및 열람을 할 수 있는 방법 또한 소개할 것이다.

본 논문 2장에서는 앞서 정의한 한계점을 해결하기 위한 해결 방안을 찾기 위해 관련조사를 한 내용을 담고 있다. 3장에서는 관련 연구를 기반으로 해서 TM 내부의 음성 녹취 정보를 텍스트 정보로 바꾸는 방법에 대한 내용과 블록체인 내에 음성 녹취 정보 및 녹취 텍스트 정보를 저장하는 방법에 대한 내용을 다룰 것이다. 마지막 4장에서는 본 눈문에서 말한 저장 방식이 앞서 언급된 TM 녹취 정보 저장에 대한 문제점들을 해결할 수 있음을 결론 내린다.

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음" (IITP-2019-2017-0-01628*)

2. 관련 연구

2.1 블록체인

블록체인은 Satoshi Nakamoto가 제안한 Bitcoin에서 나온 개념으로 데이터들이 Peer-to-Peer 방식을 기반으로 생성된 체인 형태의 연결고리 기반 분산 데이터 저장환경에 저장되어 누구라도 임의로 수정할 수 없고 누구나 변경의 결과를 열람할 수 있는 분산 컴퓨팅 기술 기반의 데이터 대변 방지 기술이다.[3] 트랜잭션이 발생하게 되면 블록체인 내의 모든 노드들에게 해당 트랜잭션이 전파되게 되고 이 트랜잭션이 유효한 트랜잭션인지에 대해 검증이 이루어지게 된다. 이렇게 검증된 트랜잭션들이 쌓이면 블록을 생성하게 되고 생성된 블록에 대해 블록체인 내의 모든 노드들에게 전파하게 된다. 그 후 작업 증명(Proof-of-Work, PoW) 과정을 통해 블록을 검증하고 검증이 완료된 블록은 이전 블록과 연결되어 블록들의 체인을 이루게 된다.

블록체인 기술의 가장 큰 핵심은 탈중앙화이다. 블록체인의 노드들은 블록체인의 부분 또는 전체를 가지고 있다. 이것은 중앙 집중형 데이터베이스에 모든 정보를 저장할 필요가 없게 한다. 이것은 또한 데이터의 임의 조작을 불가능하게 만들어 준다. 어느 데이터에 대해 수정을 하려면 그 데이터가 속한 블록의 정보를 가지고 있는 모든 노드들에 찾아가 데이터를 다 바꾸어 주어야 하기 때문이다. 이를 통해 데이터의 무결성이 유지될 수 있다. 그리고 블록체인에는 데이터가 해시 알고리즘에 의해 암호화되기 때문에 보안성이 보장되며 누구나 정보에 접근 가능하므로 투명성 또한 보장된다.

본 논문에서는 이러한 블록체인의 데이터 무결성과 투명성 그리고 보안성을 활용해 앞서 정의한 문제점을 해결하려 한다.

2.2 DHTs: Distributed Hash Tables

DHTs란 해시 테이블을 분산하여 관리하는 기술이다. 어떤 항목을 찾아갈 때 해시 테이블을 이용하는데, 중앙 시스템이 아닌 각 노드들이 이름을 값으로 매핑하는 기능을 하는 방식이다. 부하가 집중되지 않고 분산된다는 큰 장점이 있어, 극단적으로 큰 규모의 노드들도 관리할 수 있다.

DHT는 순수 P2P라도 네트워크의 부하를 억제할 수 있으며 네트워크 상의 콘텐츠를 빠르고 정확히 검색할 수 있는 것이 가능하다. 종래의 순수 P2P에서 채용되었던 방식에서는 수십만 노드 정도가 한계였으나, DHT의 사용으로 수십억 개의 노드를 검색범위로 할 수 있게 되었다.

DHT를 활용한 대표적인 시스템으로 비트토렌트(DHT를 확장하여 사용), eDonkey 등이 있다.

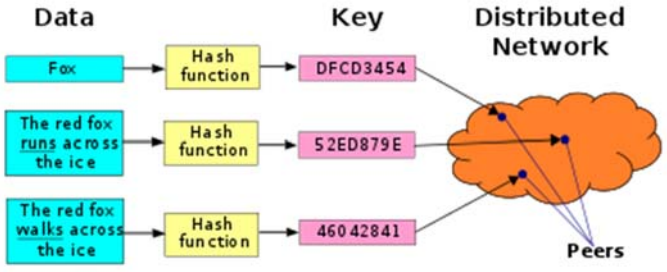


그림 1. DHT의 개념도

2.3 Merkle Tree

해시 트리는 모든 비-리프(non-leaf) 노드의 이름이 자식 노드들 이름의 해시로 구성된 트리 구조를 가리킨다. 발명자 랄프 머클의 이름을 따 머클 트리(Merkle tree)라고도 불린다.

해시 트리는 트리 구조의 일종으로, 잎 노드는 파일 등의 데이터를 가리킨다. 상위 노드는 각각 자식 노드들의 해시 값이 된다. 예를 들어 위 그림에서 해시 0은 해시 0-0과 해시 0-1을 연결한 문자열을 다시 해시 함수로 계산한 것이다. 해시 함수는 위 그림처럼 이진 트리를 사용할 수도 있지만, 임의의 차수를 가진 트리에서도 사용 가능하다.

해시 함수는 어떤 것이든 사용할 수 있지만, 보통 SHA-1, Tiger, Whirlpool 등의 암호화 해시 함수가 사용된다. 그러나 해시 트리를 사용하는 목적이 악의적인 공격자의 데이터 변조를 막으려는 것이 아니라, 단순히 오류를 찾기 위한 경우, CRC 등의 안전하지 않은 함수를 사용할 수도 있다.

데이터를 검증하고자 하는 사용자는 루트 노드의 해시 값(루트 해시 또는 마스터 해시라고 부른다)만 알면 데이터가 옳은 데이터인지 검증할 수 있다.

또한 데이터 전체가 아닌 일부만 검증하고자 할 때에도 자식 노드 가운데 하나의 해시 값을 알면 그 노드의 모든 자식 노드에 대해 데이터를 검증할 수 있는 특징이 있다. 이런 특징 때문에 만약 일부 데이터가 손상될 경우 어떤 데이터가 손상되었는지를 쉽게 찾아내어 손상된 데이터를 다시 전송받을 수 있는 장점이 있다. 예를 들어 위 그림에서 데이터 2번이 손상되었다면, 해시 0-1과 해시 0, 그리고 루트 해시가 달라지고 다른 값들은 달라지지 않을 것이다. 따라서 대량의 데이터가 있을 경우에도 손상된 데이터를 빠르게 찾아낼 수 있다.

해시 트리는 여러 블록으로 나뉜 데이터를 전송할 때 데이터가 변조되지 않았음을 보장하는 용도로 사용된다. 특히 P2P 망에서 전송받은 데이터에 오류가 있거나 악의적인 데이터 변조가 있는지를 검증하는 용도로 사용된다. 썬 마이크로시스템즈에서 개발한 ZFS 파일 시스템에서 해시 트리를 사용한다. 또한 구글 웨이브 프로토콜이나 깃 버전 관리 시스템, 비트코인 암호 통화

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음" (IITP-2019-2017-0-01628*)

시스템, 비트토렌트 프로토콜 등에서도 사용된다.
발명자 랄프 머클은 여러 개의 램포트 서명을 효율적으로 다루기 위해 해시 트리를 개발했다. 램포트 서명은 양자 컴퓨터가 실용화되어도 안전할 것으로 예상되는 디지털 서명 알고리즘이지만, 한개의 메시지마다 새로운 키를 생성해야 하는 단점이 있다. 여러 개의 램포트 키를 해시 트리로 묶으면 보다 효율적으로 다룰 수 있다. 이런 방식을 머클 서명이라고 부른다.

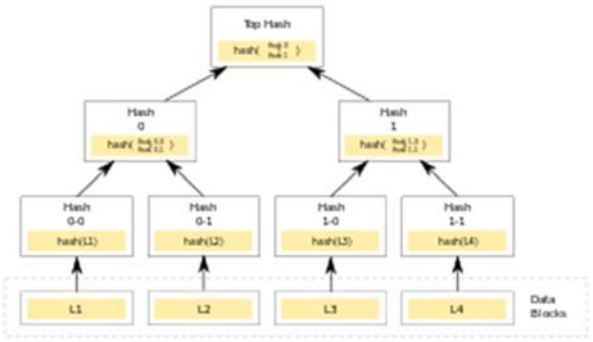


그림 2. Merkle Tree

2.4 InterPlanetary Name System

InterPlanetary Name System (IPNS)란 IPFS[4]의 name system으로 파일들의 해시값으로 이루어진 파일이름을 기준으로 Merkle DAG를 형성하여 모든 파일이 각각 영구적이고 변경할 수 없는 이름을 가지게 해주는 system이다. 이를 위해서는 Self-certified FileSystems(SFS)가 필요한데 SFS는 주소를 /sfs/(Location):(HostID)의 형식으로 표현하며, 여기서 Location은 서버의 주소이다. 또한 HostID는 hash(서버가 제공한 public key + Location)이다. 따라서 이용자는 서버가 제공한 public key를 통해 그 서버가 '주소와 일치하는 서버'임을 확인할 수 있다.

2.5 Speech to Text[5]

2.4.1. Pre-processing: 변환 및 특징 추출 청각 시스템과 같이 음성 신호로부터 시간 및 주파수 영역의 특징을 추출해 내는 과정이다. 청각 시스템의 와우각(달팽이관) 기능을 하며 음성 신호의 주기성과 동기성의 정보를 끄집어 낸다.
2.4.2. Acoustic Model: 특징으로부터 결과값 산출 음성 신호의 전처리를 통해 얻어낸 특징을 바탕으로, 문장을 구성하는데 필요한 원소인 음소, 음절, 단어를 인식해내는 역할을 하고 있다. (이를 위해 음성학, 음운학, 음운 배열론, 시형론 요구) 템플릿(사전) 기반의 다양한 알고리즘 사용되는데, 동일한 문제를 각기 다른 방식으로 접근한다. 알고리즘 별 접근 방식은 다음과 같다. (DTW: 동적 프로그래밍을 통한 접근/HMM: 확률추정을 통한 접근/Knowledge Base: 인공지능을 이용한 추론을 통한 접근/Neural Network: 패턴분류를 통한 접근)
2.4.3. Linguistic Model: 언어처리(문장 복원) 패턴 인식후의 결과인 음소, 음절, 단어를 재구성해서 문장을 복원해낸다. 이를 위해 구문론, 의미론, 어형론이 이용된다. 문장을 구성하기 위해 규칙, 통계 기반 모델을 이용한다.
- 구문규칙 모델(syntactic): 매 단어 다음에 올 수 있는 단어의 종류를 제한해 문장을 구성한다.
- 통계적 모델(statistical): 매 단어에 대해 이전의 N개의 단어가 발생할 확률을 고려해 문장을 인식(N-gram으로 표현)한다.

3. 본 론

3.1 시스템 구성

본 논문에서 제안한 음성 인식 기술을 이용한 블록체인 기반의 텔레마케팅 음성 정보 저장 시스템의 구성은 아래 그림 1과 같이 총 4가지의 Layer가 있다.

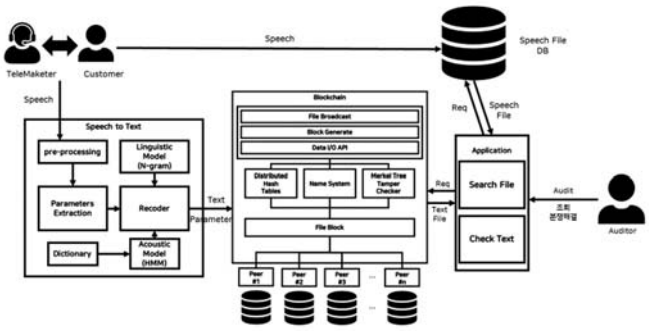


그림 3. 제안하는 시스템 개념도

3.1.1 Speech-to-Text Layer

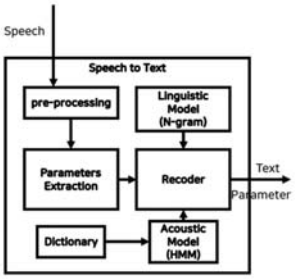


그림 4. Speech-to-Text Layer 구조도

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음" (IITP-2019-2017-0-01628*)

먼저, 위의 그림 2와 같이 TM시 발생하게 되는 저장이 필요한 음성 정보를 텍스트 정보로 바꾸어 주는 Layer이다. 여기서는 Speech 정보가 Text 정보로 바뀌는 Third-party를 나타낸 것으로 내부의 pre-processing과 Acoustic Model 그리고 Linguistic Model의 과정을 거쳐 input 정보인 Speech가 Text로 변환되고 변환되는 과정에서 필요한 Parameter들이 output으로 넘어가게 된다. output 정보는 블록체인에 저장되게 된다.

3.1.2 Blockchain Layer

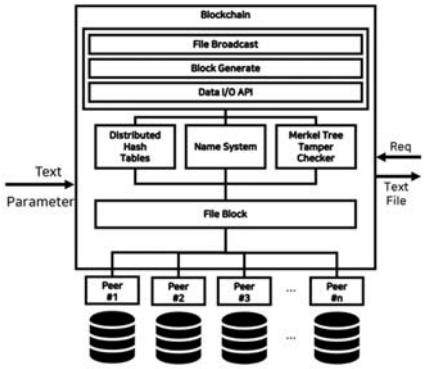


그림 5. Blockchain Layer 구조도

그 다음은 Speech-to-Text Layer에서 넘어온 Text 정보와 Parameter 정보가 저장되는 블록체인 시스템이다. Input으로 Text와 Parameter가 들어오면 해당 파일을 여러 개의 File Block으로 나누어 저장하고 이 저장본은 각각의 Peer node에 저장되게 된다. 이렇게 저장하는 까닭은 모두 같은 File을 모든 Peer node에 저장하여 위변조를 확인할 수 있으면 좋지만 저장 공간의 효율성을 높이고자 선택한 방법이다. 이렇게 했을 시 위변조의 확인은 Merkle Tree를 이용하여 각각의 파일들을 Merkle Tree의 leaf node에 위치하고 Merkle Tree의 root node만 확인하여 위변조를 확인할 수 있다. 이 일련의 과정들은 Distributed Hash Table(DHT)에 의해 관리되며 여기서 각 file들의 고유한 id 부여 및 원본 파일이 어떤 파일로 쪼개졌는지에 대한 정보가 담겨있다. 그리고 저장되는 파일들의 체계적인 관리와 쉬운 접근을 위해 Name System을 이용하며 이것을 통해 파일이 어떠한 Peer에 있던지 간에 인터넷의 DNS처럼 주소만 가지고 관리자는 파일들을 체계적으로 관리할 수 있고 사용자는 찾고자 하는 파일에 쉽게 접근할 수 있게 된다. 기본적으로 File Broadcast나 Block Generate 기능 또한 존재하고 있으며 Application Layer와 연결되는 부분에서 Request를 받고 output으로 Text File을 전달하기 위한 Data I/O API도 존재한다.

3.1.3 Application Layer

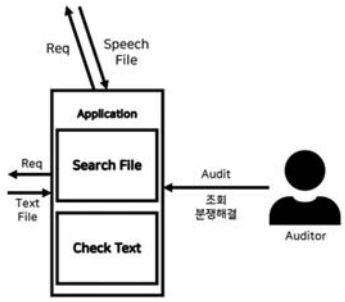


그림 6. Application Layer 구조도

사용자가 직접 사용하게 되는 Application으로 사용자가 감사, 조회, 분쟁해결을 위해 이 app을 사용하면 app은 필요한 정보를 Blockchain이나 Speech File DB에 Request하여 Text File을 가지고 오거나 원본 음성 정보를 가지고 오게 된다. 이렇게 가지고 온 정보를 토대로 작업을 수행하고 Text File로는 매크로나 프로그램을 돌릴 수 있는 환경을 제공해 준다.

3.1.4 Speech File DB Layer

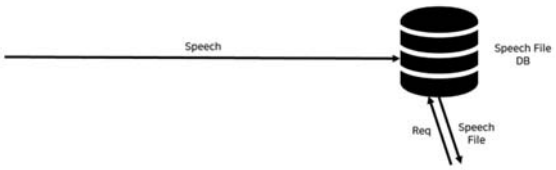


그림 7. Speech File DB Layer 구조도

만일의 상황을 대비하여 원본 Speech File을 저장하는 DB이다. 전체 시스템의 목적 자체는 Text File을 통해 모든 작업을 수행하는 것이지만 최후의 상황에 원본 Speech 정보가 필요할 때 이 정보를 제공하기 위해 단순히 Speech 정보를 DB에 저장하는 기능을 담당한다. Application에서 Request가 올 경우 Speech File을 Application에 전달해 준다.

4. 결 론

본 논문의 해결방안을 통해 텔레마케팅 계약 시의 정보에 대한 조회 및 감사 업무 그리고 분쟁 조정

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음" (IITP-2019-2017-0-01628*)

업무를 오로지 텍스트 정보를 사용하여 음성 정보를 사용할 때 소모되던 불필요한 시간과 비용을 절약할 수 있음을 보였다. 또한, 블록체인 상에 저장되는 정보 저장 체계를 통해 speech to text 과정의 신뢰를 보장하고 저장된 텍스트 정보는 위변조에 대해 면역을 지니게 되어 텔레마케팅 회사와 소비자 사이의 분쟁 관계를 최소화할 수 있음도 보였다. 추후 본 논문을 통해 이 시스템을 직접 구현하고 기존 방식과의 성능 비교 및 수치 검증을 통해 본 눈문에서 제안한 시스템을 검증할 예정이다.

참 고 문 헌

[1] 박명희, Hwang Jin-Ja. 불완전판매로 인한 보험계약 취소요건 비교 연구. 정책연구보고서, 1-4. 2008
[2] 금융감독원. TM채널의 불합리한 관행 개선 추진 보도자료, 붙임. 2018
[3] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008
[4] InterPlanetary File System (IPFS). https://ipfs.io
[5] 음성인식으로 시작하는 딥러닝. https://wikidocs.net/30649
[6] 음성 인식 기술을 이용한 블록체인 기반의 텔레마케팅 음성 정보 저장 방법. 김인근. 2019
[7] 계약 관련 음성 녹취 정보의 신뢰성 있는 저장을 위한 블록체인 기반 시스템 구성 방식. 김인근. 2019

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음" (IITP-2019-2017-0-01628*)

조치 가능한 버그 예측을 위한 유사 패치 추천

신지호 ○ 남재창
한동대학교
{21931006, jcnam}@handong.edu

Similar Patch Recommendation for Actionable Defect Prediction

Jiho Shin and Jaechang Nam
Handong Global University

요 약

소프트웨어 품질 보증 과정에서 발생되는 비용을 줄이기 위해 버그 검출/예측, 버그 위치 추정 (Fault Localization), 자동 패치 생성 (Automated Program Repair) 등 다양한 자동화 연구들이 진행되고 있다. 하지만 버그 예측과 같은 연구는 결과에 대한 설명이 어렵고 (explainability) 즉시 조치하기 어려운 문제가 (actionability) 있기 때문에 새로운 방법인 유사 커밋 검색을 통한 유사 패치 추천 기법을 제안하고자 한다. 개발자가 새로 커밋한 코드가 기존의 Bug Introducing Commit(BIC)과 유사하다면 버그 검출/예측 시 즉시 조치할 수 있는 정보를 제공하여 품질 보증에 기여할 수 있을 것으로 기대한다.

1. 서 론

소프트웨어의 복잡도가 지속적으로 늘고 있음에 따라 프로젝트들의 유지/보수 비용도 함께 증가하고 있다. 이와 같은 문제를 해결하기 위해 버그 예측과 같은 품질보증 프로세스를 자동화하는 연구들이 활발히 일어나고 있다. 하지만 기존 버그 예측 분야에서는 예측의 입상(granularity)이 크기 때문에 예측된 결과를 설명하기 어렵고 즉시 조치하지 못한다는 문제점이 있다.

이를 해결하기 위하여 유사 커밋 검색과 패치 커밋 추천 기법을 제안한다. 소프트웨어 프로젝트의 버그 리포트에는 버그에 대한 정보와 이를 수정한 패치 커밋들의 정보들이 축적되어 있다. 패치 정보들을 가지고 SZZ 알고리즘[1]을 적용하면 어느 커밋에 버그가 유입되었는지 알 수 있기 때문에 기존 소프트웨어 저장소에 있는 BIC 들과 현재 개발자가 커밋한 정보가 유사하다면 버그로 인식함과 동시에 패치를 제안할 수 있게 된다.

현재 이와 연관된 연구들로는 코드 검색, 코드 클론 (clone) 검출 및 검색, 커밋 군집, 자동 패치 생성 등이 있지만 코드 검색 같은 경우 아주 짧은 코드를 입력 받거나 자연어를 입력 받는 경우만 작동이 가능하고 코드 클론 검출/검색은 코드의 변화를 나타내는 커밋과는 형태의 차이가 분명하다. 또한, 자동 패치 생성과 같은 경우 처리할 수 있는 버그의 형태가 한정적이고 생성된 패치들이 테스트 케이스에 과다 적합 (overfitting) 되는 성향이 있다.

따라서 본 논문은 코드의 수정 정보를 갖고 있는 커밋을 소프트웨어 저장소에 버그라고 레이블 되어 있는 커밋들과 유사도를 비교하고 비슷한 커밋 일 경우, 버그 커밋을 고친 패치 커밋을 추천하는 아이디어를 제안한다.

2. 본 론

유사한 코드 커밋 검색의 가능성을 검증하기 위해 1) 수정된 코드를 벡터화 시켜 비교하는 실험과 2) 문자열로써 비교하는 실험을 실행하였고, 실험 결과를 바탕으로 3) 딥러닝으로 추출한 의미적 특징으로 유사도를 측정하는 방법을 제안한다.

3.1. 수정-벡터 수집 (Change Vector Collector)

세밀한 단위로 코드의 수정 내역을 이해하기 위해 Gumtree [2]라는 트리 비교 알고리즘을 사용했다. Gumtree 를 이용하면 수정된 양의 단위를 AST 노드 단위로 줄이고 추가된 코드와 제거된 코드 뿐만이 아닌 수정된 코드와 이동한 코드도 찾을 수 있다 [2]. 따라서 네 가지 (추가, 삭제, 수정, 이동) 작업에 대한 AST 노드의 수로 벡터를 추출하였다.

이 추출 방법으로 깃 허브에 있는 버그 수정에 대한 커밋들의 수정-벡터를 추출하고 확률 밀도 함수에서 널리 쓰이는 알고리즘 중 피어슨 상관 계수와 코사인 유사도를 사용했고 변수들의 순위 사이의 유사도를 검사하는 스피어만 상관 계수와 켄달 상관 계수를 적용하여 유사도를 측정하였다. 마지막으로 벡터간의 유사도를 집합관계에서 보여주는 자카드 지수를 사용하여 계산하였다.

그림 1.1 과 1.2 는 수정-벡터를 뽑은 후 자카드 지수를 취했을 때에 0.92 가 나온 예시이다. 그림 1.3 은 해당 커밋들에 대한 수정-벡터이다. 그림 1 의 예제와 같이 ImportDeclaration 을 추가한 인덱스 번호가 146 에 해당되고 MethodInvocation 은 159 에 해당된다. 이와 같이 수정된 AST 노드를 카운트하는 식으로 벡터를 이루었다.

하지만 이러한 코드들의 수정이 다른 메서드나 매우 다른 문맥에서 나오더라도 같은 AST 노드가 바뀌었다면 같은 카운트로 표기가 된다. 따라서 다른 코드의 위치로 인해 의미가 전혀 다르더라도 벡터가 비슷하게 나오고 그에 따른 상관계수도 높게 나오는 현상이 있었다.

```
+ import
org.apache.isis.viewer.wicket.ui.components.property.PropertyEditFor
mExecutor;
+         scalarModel.setFormExecutor(new
PropertyEditFormExecutor(scalarModel));
```

그림 1.1 수정-벡터 Isis 인스턴스 결과 예시

```
+ import
org.apache.ignite.internal.processors.rest.handlers.redis.key.GridRe
disExpireCommandHandler;
+         addCommandHandler(new GridRedisExpireCommandHandler(log,
hnd));
```

그림 1.2 수정-벡터 Ignite 인스턴스 결과 예시

{127 1,141 1,146 1,159 1,177 1,181 4,182 1,211 1,212 1}
{127 1,141 1,146 1,159 1,177 1,181 4,182 1,211 0,212 1}
그림 1.3 수정-벡터 예시

※ 본 연구는 2019 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2018R1C1B6001919).

3.2. 문자열 비교

커밋 코드들의 순서/문맥 정보를 보존하기 위해 코드 커밋을 문자열의 형태로 표현하고 문자열들의 유사도를 계산하는 알고리즘을 적용하였다.

먼저 커밋의 문자열들을 추출하기 위해서 검트리를 사용하지 않고 수정된 라인 전체를 취하는 기존의 비교 방법을 (git diff) 사용하였다. 검트리를 사용하면 바뀐 부분의 AST 노드 만을 취하기 때문에 연속된 코드의 정보를 얻지 못하기 때문이다.

유사도를 측정하기 전에 노이즈 정보를 최소화 하기 위해 전처리 과정을 다음과 같이 실시하였다. 먼저 같은 메서드 범위에서 제거된 라인과 (버그) 추가된 라인이 (패치) 모두 있어야 버그 수정과 관련된 수정이라 판단하고 제거나 추가만 있는 라인들은 모두 무시한다. 또한 코스메틱 수정에 불과한 공백들도 모두 제거한다.

문자열들의 유사도 검증 알고리즘은 벡터 계산보다 시간이 오래 걸리기 때문에 커밋 코드에 SimHash 알고리즘을 적용하여 코드의 정보를 잃는 것을 최소한으로 차원을 축소하고 일정 이상의 해밍 거리 (Hamming distance) 보다 멀면 너무 다른 문자열이라 판단하여 비교를 건너뛰는 방식을 사용하였다 [3].

문자열 유사도 비교 알고리즘으로는 자카드 거리와 레벤슈타인 거리를 사용하였다. 자카드 거리는 두 문자열을 집합으로 보고 비율적으로 얼마나 포함하는지를 측정하는 알고리즘이다. 레벤슈타인 거리는 두 문자열의 수정되는 횟수로 거리를 측정하는 알고리즘이기 때문에 순서를 고려할 수 있다. 따라서 이 두 알고리즘의 조합을 사용하였다.

```
private XMLInputSource getInputSource() throws IOException {
  try {
    return new XMLInputSource(aePath);
  } catch (IOException e) {
    return new XMLInputSource(getClass().getResource(aePath));
  }
}
```

그림 2.1 문자열 비교 Lucene-solr 예시 (제거된 라인)

```
private XMLInputSource getInputSource() throws IOException {
  try {
    return new XMLInputSource(aePath);
  } catch (Exception e) {
    return new XMLInputSource(getClass().getResource(aePath));
  }
}
```

그림 2.2 문자열 비교 Lucene-solr 예시 (추가된 라인

```
        LOG.warn("Request remote address is NULL");
        hostname = "???";
    }
} catch (Exception ex) {
    LOG.warn("Request remote address could not be resolved,
```

그림 2.3 문자열 비교 Oozie 예시 (제거된 라인)

```
        LOG.warn("Request remote address is NULL");
        hostname = "???";
    }
} catch (UnknownHostException ex) {
    LOG.warn("Request remote address could not be resolved,
```

그림 2.4 문자열 비교 Oozie 예시 (추가된 라인)

3.2.2. 결과

그림 2 는 코드 커밋을 문자열로서 비교했을 때 상관계수가 0.86 이 나온 예시이다. 결과와 같이 수정의 정도가 매우 짧은 인스턴스들만이 높은 상관계수를 보였다. 더 긴 예시들도 있었지만 상관계수가 낮기 때문에 실용성이 떨어지는 것을 볼 수 있었다.

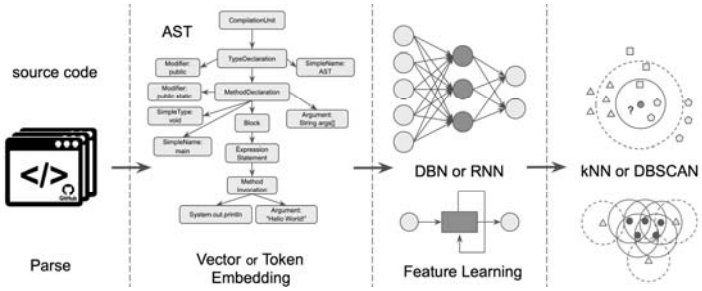


그림 3 딥러닝을 이용한 방법의 시스템 개요도

3.3. 딥러닝을 활용한 유사 커밋 검색

3.1 과 3.2 의 방법은 AST 노드 출현 횟수나 문자열의 유사도와 같이 사람이 직접 특징들을 찾아야 하는 방법이지만 의미적인 특징들 없이 문자적 특징들만 갖고 있다는 단점이 있다. 따라서 의미적인 특징을 컴퓨터가 직접 뽑을 수 있도록 딥러닝 방법을 적용하여 커밋들을 분류하는 방법을 제안한다.

그림 3 은 딥러닝을 활용한 유사 커밋 검색 시스템에 대한 개요도를 보여준다. 의미적 특징을 뽑기 위해 커밋 코드를 고차원 벡터 공간에 임베딩하고 DBN(Deep Belief Network)이나 RNN 같은 딥러닝 알고리즘을 적용하여 각 인스턴스의 특징들을 뽑아 낸다. 이러한 방법으로 코드 뭉치의 의미적 특징을 뽑아내는 연구로 [4, 5]와 같은 연구들이 있었지만 이 연구들은 버그 예측 분야에서 활용하였기 때문에 최종적으로 이진 분류가 (버그/클린) 목적이다. 딥러닝 모델을 사용한 버그 예측 분류기들이 기존 파일 단위 버그 예측 연구와 JIT 버그 예측 연구에서 성능을 크게 향상시켰다. 따라서 딥러닝을 활용하여 의미적 특징들을 뽑아 내는 것이 기존의 사람들이 뽑은 특징보다 더 효과적일 수 있다. 이런 연구들처럼 딥러닝으로 특징들을 뽑아내고 kNN(k-Nearest Neighbor)이나 DBSCAN 과 같은 알고리즘을 사용하여 이진 분류가 아닌 특징적으로 거리가 가까운 유사 커밋을 찾아 내는 것이 목표이다.

[참고 문헌]

[1] Kim, Sunghun, et al. "Automatic identification of bug-introducing changes." 21st IEEE/ACM International Conference on Automated Software Engineering (ASE'06). IEEE, 2006.

[2] Falleri, Jean-Rémy, et al. "Fine-grained and accurate source code differencing." Proceedings of the 29th ACM/IEEE international conference on Automated software engineering. ACM, 2014.

[3] Uddin, Md Sharif, et al. "On the effectiveness of simhash for detecting near-miss clones in large scale software systems." 2011 18th Working Conference on Reverse Engineering. IEEE, 2011.

[3] Wang, Song, Taiyue Liu, and Lin Tan. "Automatically learning semantic features for defect prediction." 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). IEEE, 2016.

[4] Yang, Xinli, et al. "Deep learning for just-in-time defect prediction." 2015 IEEE International Conference on Software Quality, Reliability and Security. IEEE, 2015.

블록체인 기반 실시간 수하물 트레킹 시스템

이열국 O, 이동현, 박수용
서강대학교 컴퓨터 공학과
hotgodj@hanmail.net, ldh1428a@sogang.ac.kr, sypark@sogang.ac.kr

RealTime Blockchain-Based Baggage Tracking System

Yeolkook Lee O, DongHyun Lee, Sooyong Park
Department of Computer Science and Engineering, Sogang University

요 약

현재 몇몇 항공사에서 수하물 추적 서비스를 실시하고 있는데 RFID기술을 기반으로 운용하고 있기 때문에 전파의 범위가 한정적이고 고비용이라는 문제점이 있다. 이런 문제점들을 해결하기 위해서 블록체인 기술을 수하물 트레킹 서비스에 적용하여 수하물 분실율을 낮추고 효율적인 수하물 관리를 할 수 있다. 블록체인 기술은 중요한 데이터의 무결성과 투명성을 보장하고 분산 데이터 저장 환경에 관리 대상 데이터를 저장함으로써 누구도 임의로 데이터를 수정하거나 삭제 할 수 없는 분산 컴퓨팅 기술 기반의 원장 관리 기술이다. 블록체인은 데이터의 투명성과 무결성을 보장하기 때문에 신뢰성이 있어야하는 서비스 분야에 적용할 수 있다. 본 논문에서는 기존의 RFID 방식이 아닌 블록체인 기반의 트레킹 시스템을 소개한다.

1. 서 론

비행기를 타고 목적지 공항에 내렸는데 수하물 카운터에 자신의 짐이 보이지 않는 경우 가 종종 있다. 항공사에서 분실된 짐을 찾아 뒤늦게 호텔로 보내주는 경우도 있지만, 수하물이 엉뚱한 공항으로 날아가 여행 내내 불편을 겪거나 영영 짐을 찾지 못하는 경우도 있다. 영국 일간 더 타임스는 국제항공통신협회(SITA)가 발간한 연례 보고서를 인용, 작년(2018 년) 전 세계 각지의 공항에서 분실된 여행가방이 약 2 천 480 만개에 이른다고 보도했다.[1] 항공업계 관계자에 따르면 대부분의 수하물 분실 및 지연 사고는 항공사가 환승 등의 과정에서 승객 수하물을 잘못 취급하거나 적재 실패, 지연 등으로 발생 한다. 이에 따른 항공사의 피해 승객에 대한 보상금도 약 24 억달러(약 2 조 8 천억원)에 달한다. 분실 보상액은 크게 두가지의 국제 기준을 따른다. 바르샤바 협약을 적용하면 1kg 에 20 달러(약 2 만 4 천원), 몬트리올 협약을 적용하면 최대 1131SDR 을 보상 받을 수 있다. (SDR 은 국제통화기금이 정한 특별 인출권이고, 1131SDR 은 약 180 만원이다.) 이러한 보상 대책 때문에 수하물 분실 피해를 입은 승객은 자신의 짐과 귀중품의 가치에 대한 온전한 보상을 받기 어렵고 최대 금액을 보상받으려면 수하물 내에 있는 내용물을 증명해야 하는 어려움이 있다.

승객들이 수하물 분실에 대비할 수 있는 방법은 현실적으로 많지 않다. 수하물 누락이 자주 발생하는 환승 공항에서 항공사 데스크로 찾아가 수하물이 항공기에 제대로 적재 되었는지 확인하는 방법이 있지만 환승 항공기로 이동하기도 버거울 만큼 환승 시간이 짧은 경우가 대부분이고, 실시간으로 수하물을 확인 할 수 있는 첨단 시스템이 마련되지 않은 항공사들도 많기 때문이다. 이러한 항공 수하물이 분실되는 문제를 예방하기 위해서 블록체인 기술을 도입하는 연구가 이루어지고 있다.

2008 년 Satoshi Nakamoto 의 “Bitcoin: A peer to-peer electronic cash system” 논문[2]이 나오면서 블록체인에 대한 관심과 활용분야에 대한 연구는 점점 커져 갔다. 블록체인 기술은 기존의 데이터를 중앙에서 관리(Centralized)하는 방식이 아닌 블록체인 네트워크에 참여하는 모든 사용자가 모든 거래 내역 등의 데이터를 분산, 저장한다. 블록체인의 ‘블록’에는 개인의 거래(P2P)의 데이터가 기록되며 기록된 데이터는 무결성(integrity)과 투명성(transparency)을 지니고 있기 때문에 데이터의 위,변조가 어렵다. 그림 1 은 기존의 블록체인이 어떻게 구성되었는지 보여주는 구성도이다.

* 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음" (IITP-2019-2017-0-01628*)

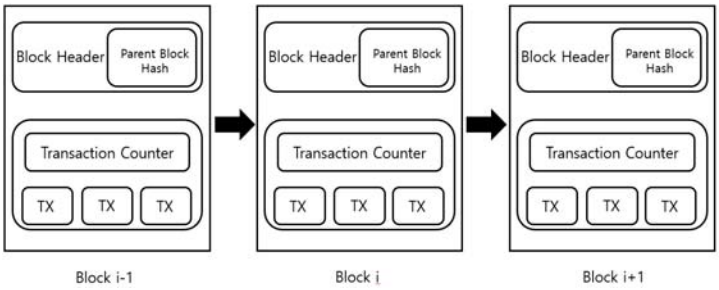


그림 1. 기존 블록체인 구성도

본 논문에서는 항공 수하물 분실을 줄이기 위해서 블록체인의 특징을 적용한 항공 수하물 실시간 트레킹 시스템을 소개한다.

2. 관련 연구

Reshma Kamath의 Food Traceability 연구[3]에서는 식품의 잘못된 관리와 오염으로 인해 매년 식중독으로 42만 명 이상이 사망하는 등의 인명피해를 해결하기 위해서 식품 공급망에 블록체인을 적용했다. 세계적인 식품 오염 문제에 대응하여 월마트는 Hyperledger Fabric을 기반으로 한 IBM의 블록체인 솔루션을 사용하여 중국의 돼지고기와 아메리카대륙의 망고를 성공적으로 추적한 결과 원산지 추적시간을 7일에서 2.2초로 단축하고 식품 공급망의 투명성을 높였다.

월마트 유통센터 및 점포 추적 블록체인을 통해 조달담당자는 식품에 대한 모든 정보(유통기한, 창고온도, 배치 번호, 토양 품질 및 비료, 배송 세부사항 등)를 원격으로 추적할 수 있다. 이 전체적인 추적 솔루션은 제품 회수 비용을 절감하고, 공정 비효율성을 줄이며, 전 세계 식품 생태계 전체에 블록체인 솔루션을 배치하여 안전성을 높일 수 있는 방법을 제안한다.

또한 최근에 몇몇 항공사에서 RFID(Radio-Frequency Identification)을 기반으로 항공 수하물 추적시스템을 도입했다. 수하물에 RFID칩을 내장하여 수하물 위치를 추적하는데 상용화된 RFID 기술들은 보안이 취약하고 전파의 적용 범위가 한정적이며 가격이 비싸다는 문제점이 있다.[4] 본 논문에서는 RFID의 단점을 극복하기 위한 블록체인 기반 정보 저장 아키텍처를 제안한다.

3. 블록체인 기반 수하물 정보 저장 아키텍처

수하물 트레킹 시스템에 블록체인을 도입하면 기존의 시스템보다 효율적인 시스템을 구축할 수 있다. 승객 개개인의 수하물에 대한 데이터를 블록체인에 저장 한 후 바코드를 발행한다. 발행된 바코드는 수하물에 부착되어 사용되고 승객은 항공사로부터 자신의 수하물을 확인할 수 있는 월렛을 부여 받는다. 월렛의 경우 두가지 종류가 있는데 온라인 상에서 모든 Transaction의 내역들을 분산화(Decentralized)방식으로 블록체인에 보관하는 형태의 핫월렛과 오프라인에서 작동하는 콜드월렛으로 나뉜다. 핫월렛은 온라인에 항상 연결되어 있어 편리하지만 해킹에 취약하고 콜드월렛은 개인키를 오프라인에서 USB 등으로 보관하기 때문에 해킹이 어렵지만 분실 시 복구하기 어렵다는 특징이 있다.

항공사 측은 공항에서 부치는 수하물에 고유번호를 부여하고 수하물을 관리 한다. 고유번호 또한 블록체인에 기록된다. 그림2는 Transaction에 어떤 데이터가 저장되는지 보여준다.

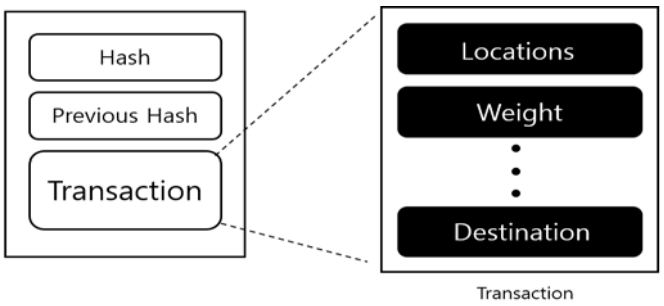


그림 2. Transaction에 저장된 수하물 데이터

승객들의 경우 Transaction 내에 자신의 수하물 데이터를 작성한 후 개인키를 이용한 서명을 통해 암호화 한다. 암호화 된 Transaction은 항공사 측에 전송되어 공개키를 통해 검증된다. 승객은 자신이 부여 받은 고유번호와 바코드를 토대로 수하물을 월렛을 통해 실시간으로 모니터링 할 수 있다. 트레킹 시스템이 블록체인을 기반으로 관리되기 때문에 RFID방식보다 저렴하게 운용할 수가 있고 수하물 분실, 파손 시 그에 따른 기록이 블록체인에 남기 때문에 분실율을 줄일 수 있다. 만약 수하물이 분실되었을 경우 분실된 공항을 블록체인을 통해 추적하여 수하물을 찾을 수 있고 분실로 인한 피해를 입었을 경우, 승객의 월렛을 통해 보상할 수 있는 방법도 제안 할 수 있다.

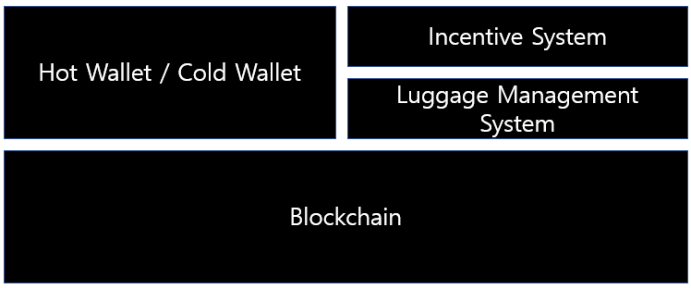


그림 3. 수하물 트레킹 블록체인 구성도

4. 결론 및 향후 연구

본 논문에서는 블록체인 기반 실시간 트레킹 시스템을 통해 공항의 수하물 분실율을 줄이고 승객들이 자신의 수하물을 실시간으로 모니터링 할 수 있는 방법을 논의하였다. 본 연구를 토대로 블록체인을 기반한

수하물 분실 및 파손에 대한 보상을 항공사 측에서 월렛을 통해 어떠한 프로세스로 승객들에게 보상해 줄 것인지에 대한 방법을 구체화하는 것이 앞으로의 향후 연구이다.

참 고 문 헌

[1]http://www.ihalla.com/read.php3?aid=1556636400628658073, 한라일보
[2] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." (2008).
[3] Reshma Kamath. “Food Traceability on Blockchain: Walmart’s Pork and Mango Pilots with IBM(2018)
[4] Understanding RFID Technology(2005)

이슈 보고서와 사용자 매뉴얼 간의 추적성 수립을 위한 머신러닝 기법들의 비교

조희태[1], *박도제[2], 이선아[3]
경상대학교 정보과학과[1], 항공우주 및 소프트웨어 공학과[2,3]
cht3205@gnu.ac.kr[1], parkdo6195@naver.com[2], saleese@gmail.com[3]

Comparison of Machine Learning Techniques for Traceability Establishment between Issue Reports and User Manuals

Hee-tae Cho[1], *Do-je Park[2], Seon-ah Lee[3]
Department of Informatics[1], Aerospace and Software Engineering[2,3]
Gyeongsang National University

요 약

요 약 : 오픈 소스 소프트웨어로 개발하는 경우 이슈 관리 시스템을 일반적으로 활용하며, 이 경우 사용자가 이슈를 보고한 이슈 보고서가 생성된다. 이슈 보고서는 개발자가 소프트웨어를 개선하는 데 많은 도움이 되는 정보를 가지고 있다. 하지만 이슈 보고서는 정해진 형식이 없어 품질이 상이하며, 중복된 내용도 많다. 특히 유명 프로젝트의 경우 하루에도 수십 개의 이슈 보고서가 작성되는데, 개발자들이 직접 읽고 문제를 식별하고 코드를 수정하기 위해서는 큰 노력과 시간이 든다. 본 연구에서는 개발자들이 이슈 보고서를 읽고 주요한 작업을 식별하는 노력을 줄이기 위해 사용자 설명서에 따라 이슈보고서를 분류하는 연구 중 이러한 분류 작업을 수행할 수 있는 머신 러닝 기법들의 성능 비교 실험을 진행한다.

키워드 : 머신 러닝, 추적성, 이슈 보고서, 소프트웨어 산출물

1. 서 론

오늘날의 오픈 소스 소프트웨어 개발은 Github와 같은 오픈 소스 플랫폼을 사용하며, 이러한 플랫폼을 사용할 경우 이슈 관리 시스템을 함께 사용한다. 사용자는 해당 소프트웨어를 사용하면서 발생한 문제점이나 불편한 점 등을 이슈 보고서로 작성한다. 개발자는 이슈 보고서를 통해 현재 소프트웨어의 문제점 등을 식별하고 해결해 소프트웨어를 개선한다. 하지만 유명한 프로젝트의 경우 한 달에 수천 개에 달하는 이슈 보고서가 작성된다. 예를 들어 Github에 있는 VScode 프로젝트 같은 경우, 2019년 11월 한 달 동안 2,000개가 넘는 이슈 보고서가 작성되었다. 또한 이슈 보고서는 권고하는 형식은 있지만 정해진 형식이 없어 품질이 상이하며, 동일한 내용이 작성된 경우도 많다. 즉, 개발자들은 중복된 내용의 이슈 보고서를 포함해 다양한 방법으로 작성된 이슈 보고서를 하루에도 수십 개씩 읽어가며 문제를 식별해야 하며, 코드를 수정해야 한다. 이러한 작업은 개발자들에게 많은 시간과 노력을 사용하게 만든다.

이러한 문제를 해결하기 위해 다양한 연구가 진행되었다. 예를 들면, 이슈 보고서에서 기능적 요구사항을 찾아내거나 [1], 작성된 내용의 심각성을 파악하기 위해 내용의 감정 분석을 진행하는 연구[2]가 있었다. 그리고 문서를 요약해 양적 부담을 줄이는 연구[3, 4]가 있었다. 하지만 이러한 연구에는 여전히 개발자가 설계 문서나 코드의 위치를 직접 찾아야 하는 문제점이 남아있다. 이 작업 역시 개발자를 지치게 한다. 이 작업을 도우려고 산출물 사이의 추적성을 수립하는 연구 및 추적성 수립에 도움이 되도록 커밋 메시지를 추천하는 연구[5]가 진행되었다. 하지만 이슈와 코드의 변경 커밋의 추적성을 수립할 뿐, 축적해서 쌓이는 이슈에 대한 개괄적인 이해를 돕지는 못한다.

본 논문에서는 이슈 보고서와 사용자 매뉴얼 사이의 추적성을 자동으로 수립하는 것을 목적으로 사용자 매뉴얼의 내용에 따라 이슈를 자동으로 분류하는 연구를 진행한다. , 이러한 자동 분류에 필요한 자연어의 벡터화 기법과 기계학습 기법의 비교를 통해 적절한 기법을 찾는 연구를 진행한다. 목표는 총 2가지 연구 질문에 답하는 것이다.

RQ1: 자연어 벡터화 기법에 따른 분류의 성능은 어떠한가?

RQ2: 기계학습 기법에 따른 분류의 성능은 어떠한가?

두 연구 질문에 답하기 위해 본 논문에서는 다음을 진행하였다. 먼저 Github에서 3개의 프로젝트를 선정하고, 각 프로젝트의 이슈 보고서와 사용자 매뉴얼을 수집하였다. 그런 다음 사용자 매뉴얼과 이슈 보고서에 자연어 벡터화 기법을 적용했다. 마지막으로 기계학습 모델에 사용자 매뉴얼을 학습시키고, 이슈 보고서를 입력해 관련 있는 사용자 매뉴얼로 분류하였다. 본 논문에서는 자연어 벡터화 기법으로 Bag-of-Words(BoW)와 임베딩 레이어를 사용하며, 기게 학습 모델로는 Naive bayes 모델과 Logistic Regression 모델을 사용하였다. 제안 방법의 평가는 하나의 이슈 보고서를 위한 커밋에 사용자 매뉴얼이 함께 변경된 것을 정답 셋으로 사용했다.

실험 결과, RQ1인 자연어 벡터화 방법에 따른 분류의 성능은 거의 모든 결과에서 BoW보다 임베딩 레이어를

사용한 자연어 벡터화 기법이 더 우수했다. RQ2인 학습 모델에 따른 분류의 성능은 VScode 프로젝트는 자연어 벡터화 기법에 상관없이 Logistic Regression 모델이 더 우수 했으며, Atom 프로젝트는 BoW를 사용했을 때는 Logistic Regression 모델이, 임베딩 레이어를 사용했을 때는 Naïve bayes 모델이 우수한 성능을 보였다. 마지막으로 komodo 프로젝트는 Naive bayes 모델과 BoW를 사용했을 때 가장 우수한 성능을 보였다. 우리는 이러한 비교 실험의 결과를 기반으로 효과적인 이슈 분류 및 검토 시스템을 개발해 나가고자 한다.

본 연구의 구성은 다음과 같다. 2절은 관련 연구를 소개한다. 3절은 자연어 벡터화 기법에 대해서 설명한다. 4절은 기계 학습 모델에 대해서 설명한다. 5절은 실험 계획에 대해서 설명하며, 6절은 실험 결과에 대해 보고한다. 7절과 8절은 논문의 결론, 한계점, 향후 연구에 대해 논의한다.

2. 관련 연구

본 연구는 OSS 프로젝트의 이슈 보고서를 분류하는 연구에 속한다. 이와 유사한 연구로 Thorsten Merton과 Merten과 동료들은 이슈 관리 시스템(issue tracking system)에 작성되는 보고서가 소프트웨어 진화에 유용하지만 현실적으로 작성된 보고서에서 유용한 정보만을 찾아내기 어렵다는 문제점을 제기하며, 다양한 기계학습 알고리즘과 자연어 처리 기법을 사용해 많은 보고서 중 “requests”, “clarifications”, “solution proposals”에 관련된 보고서를 분류하는 연구를 진행하였다[1]. Geunseok Yang과 동료들은 소프트웨어에서 유지보수의 중요성과 보고서에 작성된 버그 중 먼저 해결해야 하는 버그를 찾는 것의 유용함을 제시하며, 보고서에 작성된 텍스트에서 감정적인 표현을 분석해 버그의 심각도의 레벨을 분류하는 연구를 진행하였다[2].

본 연구는 사용자 매뉴얼에 나온 제품의 피처 설명에 따라 분류하는 연구로 기존보다 상세한 수준에서 분류를 진행한다. 본 연구는 추적성 수립 연구에 해당할 수도 있다. 과거 소프트웨어 산출물건의 추적성에 관련된 연구들이 있었다. Michael Rath와 그 동료들은 기존의 Information Retrieval 기반의 추적성 생성 방법은 하나의 오타라도 있으면 불완전해진다는 문제를 제기하며, 6개의 OSS 프로젝트를 대상으로 이슈와 커밋을 분석하여 시간적 관련성을 사용해 둘 사이의 누락된 추적성을 생성하는 기법을 제시했다[3]. 다른 연구로 Chris Mills와 동료들은 산출물건의 1:1 연결(e.g., UC:CC, CC:TC, etc.)을 생성하는 TRAIL이라는 분류 기법을 제시했다[4]. 또한 Beatriz A. Sanchez는 실제 기업에서는 동일한 개발 주기에서도 다양한 도구를 사용해 다양한 형식의 산출물이 생성되어 관리가 어려움을 문제로 제시하며, 각 개발 주기에서 생성되는 다양한 형식의 산출물을 관리할 수 있도록 하는 방법을 제시하였다[5]. 본 연구는 사용자 매뉴얼과 이슈 보고서 간의 추적성 수립을 목표로 함으로써 상기 연구들과의 차이점이 있다.

3. 자연어 벡터화 기법

3.1 Bag-of-Words

Bag of Words (BoW)는 자연어를 벡터화하는 가장 간단한 기법 중 하나이다. BoW는 하나의 글에 등장하는 단어의 집합과 그 단어의 등장 횟수를 기반으로 생성한다. 예를 들어 하나의 문장을 대상으로 BoW를 생성할 때, 먼저 해당 문장에 등장하는 모든 단어에서 중복된 단어를 제거한 단어 집합을 만든다. 다음으로 집합의 크기와 동일한 크기의 배열을 만들고 0으로 초기화시킨다. 마지막으로 문장을 단어 단위로 자르고, 단어별로 단어 집합에서 해당 단어의 위치 i를 찾아 0으로 초기화한 배열의 i번째 값에 1을 더해주면 해당 문장의 BoW가 완성된다. 그 결과 각 단어의 등장 횟수를 가진 배열을 획득한다.

다수의 문장을 대상으로 할 때는, 모든 문장에 존재하는 단어를 대상으로 중복된 단어를 제거한 단어 집합을 만들고 문장마다 나머지 과정을 진행한다. 예를 들어 그림 1과 같이 문장 S1과 S2가 있을 때, 문장별 BoW는 다음과 같은 순서로 생성된다. 먼저 문장 S1 “traceability is important”와 문장 S2 “to recover traceability is harder then to make traceability”의 두 문장에서 단어의 중복을 제거한 단어 집합 (traceability, is, important, to, recover, harder, then, make)을 생성한다. 다음으로 생성된 단어 집합의 크기가 8이므로, 크기가 8이고 0으로 초기화된 배열 BoW_S1과 BoW_S2를 생성한다. 마지막으로 문장 S1, S2를 단어 단위로 잘라, 단어 집합에서 각 단어의 위치 i를 찾고, 각각 BoW_S1과 BoW_S2의 i번째 값에 1을 더해준다. 그 결과 BoW_S1은 [1, 1, 1, 0, 0, 0, 0, 0]의 값을 가지는 배열이 되며, BoW_S2는 [2, 1, 0, 2, 1, 1, 1, 1]의 값을 가지는 배열이 된다.

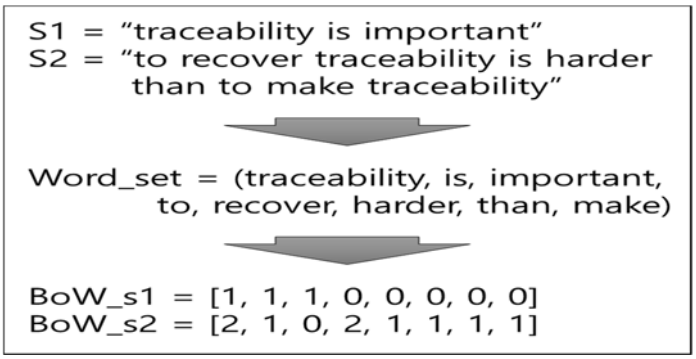


그림 1 Bag-of-Words 생성 예시

3.2 Word2Vector

Word2Vector는 하나의 단어와 주변 단어 사이의 관계를 파악해 단어의 의미를 대표할 수 있도록 하는 벡터를 생성하는 방법이다. Word2Vector 모델을 학습시키는 대표적인 방법으로는 continuous bag-of-words (CBOW)와 Skip-gram이 있다. 하지만 이 방법으로 모델을 만들기

위해서는 정갈하게 정리된 대량의 데이터가 필요하다. 이 문제는 Google에서 제공하는 Word2Vec 모델이나 GloVe 모델을 사용하면 해결할 수 있지만, 해당 모델들은 사전에 훈련된 단어들만 포함하고 있기 때문에 이슈 보고서나 사용자 매뉴얼에 등장하는 특수한 단어나 코드를 포함하고 있지 않다. 따라서 본 논문에서는 분석하는 사용자 매뉴얼과 이슈 리포트에서 등장하는 단어들의 벡터를 만들기 위해 pytorch 모듈에서 제공하는 임베딩 레이어를 사용한다.

4. 기계 학습 모델

4.1. Native Bayes

Native Bayes 모델은 다양한 분야에서 분류나 예측 연구에서 널리 사용되며 성능 또한 좋게 나온다. 이 모델은 입력된 데이터를 Bayes' Theorem에 기반하여 학습한다[6]. Bayes' Theorem이란 조건부 확률을 계산하기 위한 Bayes의 확률론 중의 하나이다. 예를 들어 사건 A가 일어날 확률 $P(A)$와 사건 B가 일어날 확률 $P(B)$ 사건 A가 일어나고 B가 일어날 확률 $P(A|B)$와 사건B가 일어나고 A가 일어날 확률이 $P(B|A)$ 라고 할 때에 $P(B|A)$를 구할 수 있다면 다음 식 $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$을 이용하여 사건 A가 일어나고 B가 일어날 확률을 구할 수 있다. 이것이 Bayes' Theorem이다. 본 연구에서는 python의 scikit-learn[1] 모듈의 Multinomial NB를 를 기본 설정으로 사용한다.

4.2. Logistic Regression

Logistic Regression 모델은 다양한 분야에서 분류나 예측 연구에서 폭넓게 사용된다. 이 모델은 입력된 데이터를 logistic function을 기반으로 학습하게 된다[7]. 로지스틱 회귀 모델이란 직선으로 분류할 수 없는 데이터를 분류하는 모델로, 시그모이드 함수를 사용하여 최적의 곡선을 찾아내는 것이다. 주어진 데이터에서 문제를 풀기 위한 가설 식은 $H(X) = \frac{1}{1+e^{-W^T X}}$ 과같이 시그모이드 함수로 표현되며, 가설 식에서 나오는 e는 자연 상숫값이고 W는 임의의 가중치이고 T는 W의 전치행렬을 의미하며, 마지막으로 X는 입력값이다. 모델의 학습은 실제 값과 예측값 사이의 오차를 최소화하는 방법으로 진행된다. 두 값 사이의 오차를 최소화하기 위해 사용하는 식은 $cost(W) = -\frac{1}{m}\sum ylog\big(H(x)\big) + (1-y)(\log(1-H(x))$ 로 표현되고 손실 함수라 부른다. 손실 함수에서 나타나는 W는 임의의 가중치이며, m은 전체 데이터의 개수이고, y는 입력값에 대한 실제 정답 값이다. 이미 정해진 값인 X와 y는 변경할 수 없다. 즉, 손실 함수를 최소화하기 위해서 적절한 가중치 값을 찾아야 한다. 적절한 가중치 값을 찾기 위해서는 경사 하강법과 W:=W- α ∂/∂W cost(W)로 표현되는 가중치 갱신을 위한 식을 사용한다. 가중치 갱신을 위한 식에서 W:=W일 때 가장 최적화된 가중치 값이 된다. 즉, cost(W)를 미분한 기울기 ∂/∂W의 값이 0이 되면 된다. 경사 하강법을 통해 지속해서 W를 갱신하는 작업을 진행한다. 본 연구에서는 python의 sci kit-learn 모듈의 Logistic Regression을 사용하며, 설정으로는 multi class='multinomial' solver='newton-cg'를 제외한 나머지를 기본설정으로 사용한다

[1] https://scikit-learn.org/stable/index.html

5. 실험 계획

본 5절에서는 실험의 평가를 위한 연구 질문과 실험 대상, 실험 절차와 평가 기준을 설명하며 실험 결과를 기술한다.

5.1 RQ1. 자연어 벡터화 방법에 따른 분류의 성능은 어떠한가?

컴퓨터가 자연어를 이해하기 위해서는 자연어를 컴퓨터가 이해할 수 있는 형식으로 바꾸어야 한다. 그중 가장 대표적인 방법이 자연어를 벡터로 변환하는 것이다. 하지만 자연어를 벡터로 변환하는 방법은 다양하며 자연어 벡터화 방법이 분류의 성능에 영향을 미칠 것이다.

5.2 RQ2. 학습 모델에 따른 분류의 성능은 어떠한가?

기존의 연구 틀에서도 본 연구에 사용되는 모델들을 이미 많이 사용하나 통일한 도메인에서 비슷한 연구를 진행하더라도 좋은 결과를 보인 모델들은 상이하다. 따라서 학습 모델에 따라 분류의 성능에 차이가 있을 것이다.

5.3 실험 대상

실험 대상의 선정은 오픈 소스 프로젝트를 대상으로, 프로젝트의 이슈 보고서에 작성된 내용이 해당 프로젝트의 사용자 매뉴얼에 작성된 기능으로 구분될 수 있을 것 같은 도메인을 선정하였다. 이에 따라 선정된 프로젝트는 총 3개의 통합개발환경 프로젝트 Atom, Komodo, Vscode이다. 실험에는 선정된 프로젝트의 이슈 보고서와 사용자 메뉴얼를 사용하였다.

5.4 실험

실험은 수집한 사용자 매뉴얼에 전처리를 진행하고 벡터화하여 각각의 모델에 학습시킨다. 그런 다음 이슈 보고서에도 통일한 전처리를 진행하여 학습된 모델에 입력하여 출력되는 결과를 실제값과 비교한다. 실제값은 이슈 보고서를 직접 읽고 가장 관련성이 크다고 생각되는 사용자 매뉴얼로 직접 연결하는 작업을 진행하여 만들었다.

5.5 평가 기준

평가의 기준은 추천/분류 도메인에서 가장 많이 사용되는 정밀도(Precision)와 재현율(Recall)을 사용하여 평가를 진행한다. 정밀도와 재현율을 평가지표로 보고 F1-score을 성능의 지표로 본다. 본 연구에서의 정밀도, 재현율과

F1-score는 아래의 식을 이용하여 계산한다. 각 학습 데이터와 테스트 데이터의 크기는 표1을 참조한다. 각 학습 데이터에 대한 단어의 수와 임베딩 사이즈는 표2를 참조한다.

$$Precision = \frac{\sum_{i=0}^{N} Precision_i}{N} \quad Precision_i = \frac{A_i}{A_i \cup C_i}$$

정밀도 관련 식 1 정밀도 관련 식 2

$$Recall = \frac{\sum_{i=0}^{N} Recall_i}{N} \quad Recall_i = \frac{A_i}{A_i \cup B_i}$$

재현율 관련 식 3 재현율 관련 식 4

$$\mathbf{F1-score} = 2 * \frac{Precision * Recall}{Precision + recall}$$

F1-score 관련 식

표1 각 학습 데이터 및 테스트 데이터

	Train (이슈 리포트)	Test (유저 매뉴얼)
Atom	56	246
Komodo	58	244
Vscode	107	238

표2 각 단어의 수와 임베딩 사이즈

	#of word	Embedding_size
Atom	3068	300
Komodo	8842	300
Vscode	12439	300

6. 실험 결과

6.1 RQ1. 자연어 벡터화 방법에 따른 분류의 성능은 어떠한가?

결과는 그림2, 그림3을 참조한다. Atom과 Komodo 프로젝트는 Logistic Regression 모델 모두에서 BoW를 이용한 벡터화 보다, Pytorch 모듈의 임베딩 이어를 통해 벡터화한 결과가 더 좋았다. 하지만 Logistic Regression 모델을 사용했을 때 BoW를 이용한 벡터화 방법이 더 우수한 성능을 보였다.

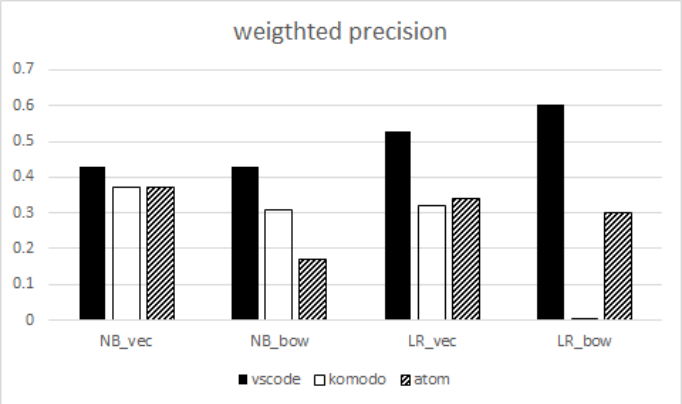


그림 2 벡터화 방법 및 학습 모델에 따른 각 프로젝트 별 정밀도

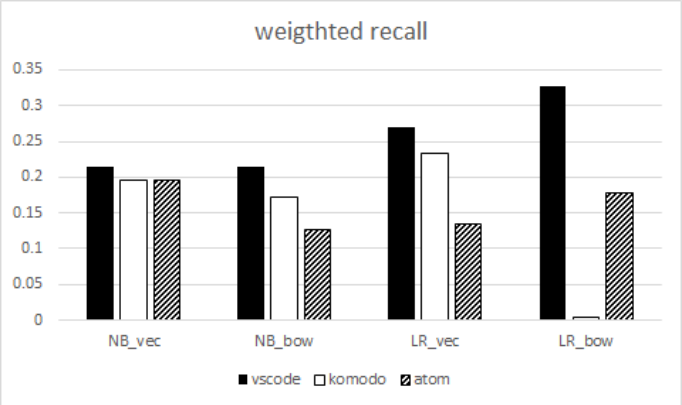


그림 3 벡터화 방법 및 학습 모델에 따른 각 프로젝트 별 재현율

6.2 RQ2. 학습 모델에 따른 분류의 성능은 어떠한가?

데이터가 동일해도 학습 모델에 따라 분류의 성능이 달랐으며, 이에 근거한 자료는 표 3 그림 2와 그림 3을 참조한다. Atom과 Komodo 프로젝트는 Native Bates 모델과 임베딩 레이어를 통한 벡터화를 사용했을 때 가장 높은 성능을 보였다. 하지만 Vscode 프로젝트는 정반대의 결과를 보인다. 즉, Logistic Regression 모델과 BoW를 통한 벡터화를 사용했을 때 가장 높은 성능을 보였다.

7. 한계점 및 향후 연구 과제

본 연구의 한계점은 먼저 실험의 도메인이 통합개발환경 하나인 점과 그 중에서도 3개의 프로젝트만 선정하여 다양성이 부족하다. 다음으로 실험에 사용되는 데이터의 신뢰도를 정확한 수치로 표현할 수 없다는 점이다. 세 번째로 연구 질문을 해결하기 위한 비교 대상들의 수가 적다는 점이다. 마지막으로 최종 결과의 성능이 그렇게 높지 않다는 점이 있다.

표3 각 프로젝트 별 성능

	Precision	Recall	F1-score
Atom	0.37	0.20	0.23
Komodo	0.37	0.18	0.23
Vscode	0.68	0.31	0.27

7.1 개선방안

실험 대상을 다양한 도메인에서 선정하여 축적하는 형태로 프로젝트 선정과 관련된 한계점은 제거할 수 있다. 다음으로 성능 부분은 더 세부적인 데이터 전처리와 다양한 모델을 사용하여 개선의 여지가 있다.

7.2 향후 연구 과제

추적성 자동 수립의 성능 개선이 일차적인 과제이며, 더 나아가 새로운 요구사항 추가에 따른 추적성 자동 갱신까지 연구 할 수 있다.

8. 결론

본 논문의 목적은 이슈 보고서와 사용자 매뉴얼 사이의 추적성을 자동으로 생성하기 위한 적절한 자연어 벡터화 기법과 기계학습 기법을 비교해 찾는 것이었다. 그 결과 정확도 측면에서 Word2vec 을 활용한 Logistic Regression 기 세 개의 프로젝트에 대해 비교적 합리적인 정확도와 재현율을 보임을 확인하였다. 하지만 실험에 사용한 프로젝트가 3 개, 자연어 벡터화 기법이 2 개, 기계학습 기법이 2 개로 전체적으로 다양성이 부족한 점과 성능이 저조한 한계점이 존재한다. 이러한 한계점들을 극복할 방법들을 모색해 추가적인 연구를 진행한다면 더 적절한 기법을 발견할 것이며, 우수한 성능을 낼 수 있을 것으로 기대한다.

참고 문헌

[1] Merten, T., Falis, M., Huner, P., Quirchmayr, T., Busner, S., andech, B. “Software feature request detection in issue tracking systems.”In 2016 IEEE 24th International Requirements Engineering Conference (RE) (pp. 166-175). IEEE. 2016.

[2] Yang, G., Baek, S., Lee, J. W., and Lee, B. “Analyzing emotion words to predict severity of software bugs: A case study of open source projects.” In Proceedings of the Symposium on Applied Computing (SAC) (pp. 1280-1287). ACM. 2017.

[3] Rath, M., Rendall, J., Guo, J. L., Cleland-Huang, J., and Mader, P. “Traceability in the wild: automatically augmenting incomplete trace links.” In 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE) (pp. 834-845). IEEE. 2018.

[4] Mills, C., Escobar-Avila, J., and Haiduc, S. “Automatic Traceability Maintenance via Machine Learning Classification.” In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 369-380). IEEE. 2018.

[5] Sanchez, B. A. “Context-aware traceability across heterogeneous modelling environments.” In Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings (pp. 174-179). ACM. 2018.

[6] Patil, T. R., and Sherekar, S. S. “Performance analysis of Naive Bayes and J48 classification algorithm for data classification.” International journal of computer science and applications, 6(2), 256-261. 2013.

[7] King, G., and Zeng, L. “Logistic regression in rare events data.” Political analysis, 9(2), 137-163. 2001.

E-Learning을 위한 Open CV 기반 집중도 측정 시스템 설계

임대근 ○, 조재춘

상명대학교 스마트정보통신공학과

eorms7427@naver.com, Jae@smu.ac.kr

Design of Concentration Measurement System Design based on Open CV for E-Learning

Dae-Geun Yim○, Jaechoon Jo

Department of Information and Communication Engineering Sangmyung University

요 약

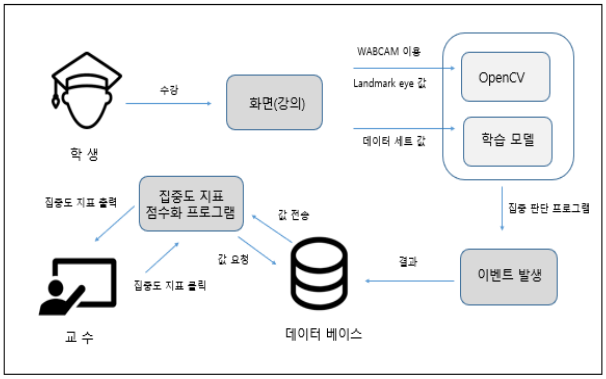
빠르게 발전하고 있는 정보화 시대에 맞춰 교육환경에서도 많은 발전과 영향이 있다. 이에 대표적인 발전으로 온라인으로 수업을 받을 수 있는 이러닝(E-Learning)이 있다. 그러나 이러닝은 오프라인의 수업에서의 장점인 직접적인 교류와 집중도, 성취도를 기대하기 어렵다. 본 논문은 이러닝을 사용하는 사용자의 집중도를 Open CV 기반으로 사용자 눈 깜빡임에 연관시켜 집중도 지표를 알 수 있는 시스템을 구축하였다. 이러닝의 강의를 듣는 학생의 눈의 깜빡임을 검출하고 학생의 집중도 여부를 판단하여 교수가 이러닝에서 학생의 집중도를 확인한다. 위의 시스템을 이용해 오프라인 수업의 장점인 높은 수업 참여도와 집중도를 온라인 수업인 이러닝에서도 기대할 수 있다.

1. 서 론

대학에서는 많은 학생들이 이러닝을 통해 학교를 가지 않고도 수업을 진행하고 정부에서는 앞으로 중,고등학교에서도 이러닝 수업을 확대하겠다는 방침을 내놓았다[1]. 하지만 면대면 공간에서 소통하는 형식이 아닌 학습방법을 학생들의 수업집중도와 참여도를 부정적으로 판단하고 있다. 오프라인 수업에서의 장점인 직접적인 교류와 집중도, 성취도를 바라기 어렵다. 따라서, 본 논문에서 제시하는 시스템은 OpenCV를 이용하여 이러닝을 이용하는 학생들의 화면 응시를 분석하여 수업 중인 화면에서의 체크 포인트 생성 횟수, 포인트 클릭 횟수를 측정하고 이를 계산하여 집중도와 연관시켜 온라인의 수업에서의 집중도와 참여도를 오프라인 수업 정도의 집중도와 참여도를 목표할 수 있는 시스템을 제안한다.

2. E-Learning에서의 집중도 지표 측정 시스템

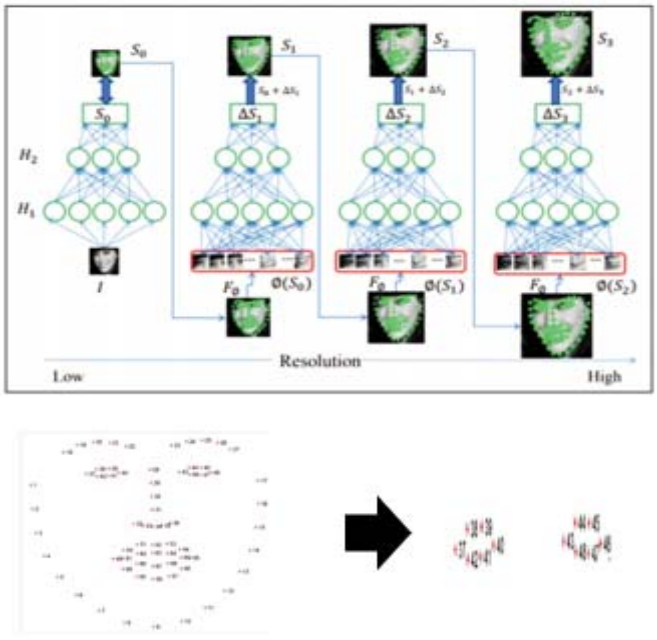
본 시스템은 교수, 강의, 학생이 이러닝에서 하는 온라인 수업의 단점인 학생의 낮은 성취도와 집중도 향상을 목표로 한다.



<그림 1> 시스템 전체 진행도

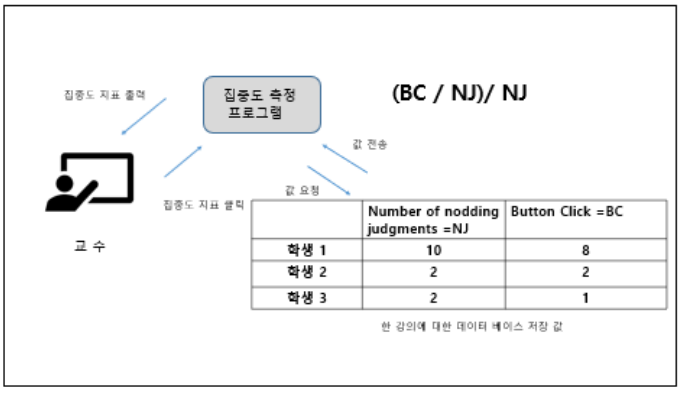
본 시스템은 평소와 같이 강의를 시청하고 해당 모습을 영상으로 취득하여 실시간 영상처리를 하게 된다. 영상을 OpenCV 를 이용하여 얼굴 랜드마크 검출을 위한 Cascade 기반의 랜드마크 검출 방식이다[2]. 본 논문에

서는 랜드마크의 눈을 찾게 되고 눈을 뜨고 있는지 감고 있는지를 실시간으로 확인하고 지표로 표현한다. 또한 딥러닝(Deep Learning)을 이용하여 정수리를 추출(Detection)을 하게 되고 정수리가 추출되면 집중하고 있는지를 판단하고 지표로 표현한다. 본 지표를 이용하여 집중하고 있는지를 판단하고, 해당 지표는 강의하는 교수가 볼 수 있는 시스템이다.



<그림 2> Cascade 기반 랜드마크 검출 및 눈 검출

그림 2 는 집중 판단 프로그램에서의 눈 검출의 기반이 되는 알고리즘이다. landmark 알고리즘에서 눈에 해당되는 37 번부터 48 번까지를 추출하게 된다. 눈을 뜨고 있을 경우 추출이 되고 안 뜨고 있거나 다른 곳을 보고 있다면 추출에 실패하게 된다. 추출 30 초 이상 실패시 집중을 못하고 있다고 판단하게 된다. 집중 못하고 있다고 판단되면 모니터에 버튼 이벤트를 무작위로 표시된다. 버튼 생성 횟수, 버튼 클릭 횟수, 버튼 클릭 비율이 데이터 베이스에 저장된다. 이 후, 교수가 집중도 지표를 클릭하게 되면 데이터베이스에 있는 값을 불러와 집중도 지표 점수화 프로그램에 의해 점수화 되어 교수가 보기 쉽게 표시된다.



<그림 3> 집중도 측정 프로그램

그림 3 은 교수의 필요로 자신의 강의에 대한 집중도 점수를 확인할 수 있다. BC 는 버튼 클릭 횟수를 나타내고 N J 는 졸음 판단 횟수를 나타낸다. 집중도 측정 프로그램은 밑에 예시와 같다.

<예시 1> 한 강의에 대한 데이터 베이스

학생 1 (졸음 판단 횟수 = 10, 버튼 클릭 횟수 = 08)
학생 2 (졸음 판단 횟수 = 02, 버튼 클릭 횟수 = 02)
학생 3 (졸음 판단 횟수 = 02, 버튼 클릭 횟수 = 01)

<예시 2> 점수 지표화 프로그램

((버튼 클릭 횟수/졸은 판단 횟수)/졸음 판단 횟수))

학생 1 은 (8/10)/10 = 0.08 점
학생 2 은 (2/2)/2 = 0.5 점
학생 3 은 (1/2)/2 = 0.25 점

예시 1 은 데이터베이스에 저장된 값을 가지고 온 것이다. 예시 2 은 예시 1 으로 집중도 지표를 점수화한 것이다. 예시 2 처럼 버튼 클릭 비율이 좋거나 수가 많다고 지표가 높은 것이 아니라 우선적으로 졸음 판단 횟수가 적은 것이 중요하다. 값이 적을수록 집중도 지표가 낮다고 판단한다. 교수는 자신의 강의에 대한 학생들의 집중도 지표 점수를 확인할 수 있다.

3. 결론

기존의 이러닝의 단점인 낮은 인식의 목표를 심어주고 출석이 학생들의 참여율, 성취도와 비례하지 않다는 문제점을 해결하고자 본 논문에서 제시한 시스템은 눈의 움직임을 기준으로 수업 중 집중도를 점수화하고 교수가 직접 확인할 수 있도록 제안한 시스템이다. 이러닝의 단점을 극복하고 이러닝에서의 성취도를 오프라인 수업 정도로 맞출 수 있다고 기대된다. 하지만 단순 눈 움직임 조금 더 집중도를 확실하게 확인할 수 있는 알고리즘을 추가하여 개선할 여지가 있다. 점수 지표화 프로그램에서의 연산을 과학적으로 입증할 필요도 있다.

감사의 글

본 연구는 문화체육관광부 및 한국콘텐츠진흥원의 2017 년도 문화기술 연구개발 지원사업으로 수행되었음[R2017030045]. 상명대학교의 재원으로 지원을 받아 산업연계교육활성화선도대학(PRIME) 후속사업의 일환으로 수행된 연구임.

4. 참고문헌

[1] 이러닝(전자학습)산업 발전 및 이러닝 활용 촉진에 관한 법률, 법률 제14998호

[2] Y. Sun, X. Wang, X. Tang, “Deep Convolutional Network Cascade for Facial Point Detection”, CVPR 2013)

[3] 공도현, 곽근창, “눈 개폐의 빈도수를 통한 운전자의 졸음판단 분석”, 한국지능시스템학회 논문지 제26권 제6호)

[4] 이소영, 김형준, “이러닝 환경에서 몰입에 영향을

미치는 요인 연구", 한국콘텐츠학회논문지 제19권 제10호)
[5]김대옥, 홍종광, 변혜란, "비제약적 환경에서 얼굴 주요위치 특징 서술자 기반의 얼굴인식", 정보과학회논문지 제41권 제9호)

온라인 교육을 위한 Word2vec 기반의 핵심 문장 추출 시스템 설계

유준영◦, 유재준, 조재춘
상명대학교 스마트정보통신공학과
jyyu7777@gmail.com◦, eileenyoo1@naver.com, jae@smu.ac.kr

Design of key sentences extraction system based on Word2vec

Junyoung Yoo◦, Jaejun Yoo, Jaechoon Jo
Department of Smart Information and Telecommunication Enginerring,
Sangmyung University

요 약

온라인 강의 학습효과를 높이기 위한 많은 연구들이 진행되고 있다. 본 논문은 학습자의 강의집중보다 학습 성과 향상을 중점에 두었다. 온라인 강의를 수강하는 학습자들의 핵심내용 파악 및 성취도 향상을 위한 문장 추출 시스템을 설계했다. 본 시스템은 TF-IDF와 Word2vec방식을 이용하여 강의마다 TF-IDF값을 통한 키워드를 추출하고 Word2vec의 Skip-gram방식을 이용하여 입력된 키워드를 통해 문장을 검색한다. 검색하여 나온 문장의 보충 설명이 필요한 경우 코사인 유사도를 통해 앞뒤 문장과의 핵심내용을 출력한다. 시스템을 통해 추출된 내용들은 이러닝 수강자들이 참고하여 강좌의 목표와 내용을 이해하는데 더 효과적으로 도울 수 있을 것으로 기대된다.

1. 서 론

이러닝은 평생교육의 시대에서 급속도로 발전했다. 특히 바쁜 일상에서 지식 습득의 효용성을 높여주었고 여러가지 제약 상황을 해결해준다[1]. 이러닝의 학습 성과는 학업성취도, 학습만족도 등 다양한 지표로 측정되고 있지만 실질적으로 학습자들이 강의를 듣고 성취를 이뤘는지 여부는 판단하기 힘들다. 이러닝과 관련하여 학습자의 집중을 요구하는 방식도 있지만, 본 논문에선 학습자들의 성과를 높이기 위한 방법으로 TF-IDF와 Word2vec 알고리즘을 이용한 시스템을 제안한다.

TF-IDF는 정보검색과 텍스트 마이닝 분야에서 활용되고 있으며 여러 문서에서 단어가 특정 문서 내에서 얼마나 중요한지 나타내는 통계기반 기술이다[2]. 이러닝 강의 키워드를 추출하기위해 TF_IDF를 이용하여 문서에 있는 자막들을 추출하고 각 단어들의 TF-IDF값을 계산한다.

주요 핵심 문장 추출을 위한 방안으로 TF-IDF와 Word2vec방식을 결합한 모델을 제안한다. 키워드를 통해 핵심 문장 추출을 위한 두가지 방식으로 CBOW(Continuous Bag of Words)와 Skip-Gram이 있다. CBOW는 주변단어를 통해 중간단어를 예측하고 Skip-Gram은 중간단어를 이용해 주변단어를 예측하는 방식이다. 본 논문에서는 Skip-Gram 방식을 이용하여 키워드를 입력하고 핵심 문장을 추출하는 모델을 개발했다. TF-IDF를 통해 키워드 추출 후 Skip-gram을 통해 핵심 문장을 추출한다. 추출된 문장의 결과가 부연설명이 필요한 짧은 결과가 나올 경우 앞뒤 문장에 포함된 단어의 코사인 유사도를 측정하여 부족할 수 있는 내용을 보충한다.

시스템을 통해 추출된 문장들은 학습자의 이러닝 수강과정에서 이해하기 힘든 부분에 있어서 참고 내용으로 활용할 수 있도록 학습자에게 제안한다.

2. 본 론

2.1 키워드 처리

키워드 처리를 하기 위해 통계적 기법으로 사용되는 대표적인 알고리즘으로 Term Frequency-Inverse Document Frequency(TF-IDF)가 있다[3]. TF-IDF는 단어의 빈도와 역 문서 빈도를 사용하여 Document-Term Matrix(DTM) 내의 각 단어들마다 중요한 정도를 가중치로 주는 방법이다.

TF는 특정한 단어가 문서 내에 얼마나 자주 등장하는지 나타내는 값으로 값이 높을수록 문서내의 언급횟수가 높다. 반면 문서군 내에서 자주 사용되는 경우, 흔한 단어임을 의미한다. 이것을 DF라고 하며 이 값을 역수를 취해서 TF값과 곱하면 불용어와 같이 흔하게 쓰는 단어들은 다른 문서군 내에서도 많이 활용되어 키워드에서 제외시킬 가능성이 높다. TF는 하나의 강좌에서 언급된 횟수를 의미하고 IDF값은 같은 학습 목표나 동일한 분야인 강의 영상 내에서만 계산한다. TF값과 IDF값을 통한 가중치 w값을 구하는 계산식은 수식 1과 같다.

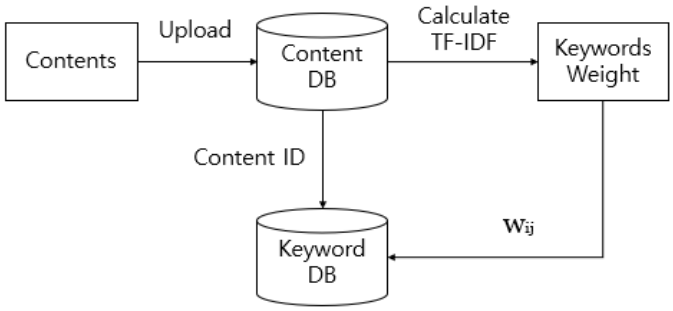
$$tf_{ij} = \max \frac{f_{ij}}{max\{f_{1j}, f_{2j}, \cdots f_{ij}\}}$$

$$idf_i = \log \frac{N}{df_i}$$

$$w_{ij} = tf_{ij} * idf_i$$

$$i = \text{단어}, j = \text{강좌번호}, N = \text{강좌의 수} \quad (1)$$

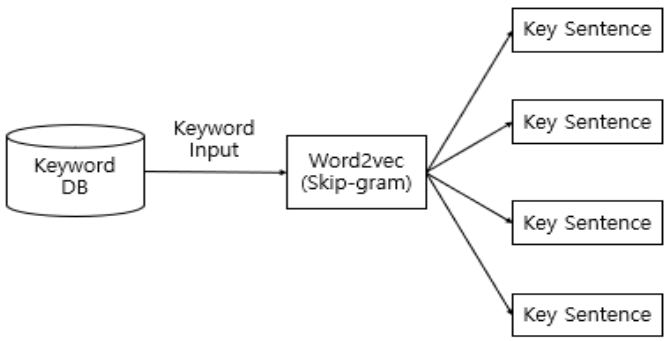
이러닝 강좌가 Content DB에 업로드 되면 시스템은 자막을 추출하여 TF-IDF 알고리즘을 수행한다. 계산된 각 단어들은 TF-IDF값을 부여 받고 강좌 키워드로 활용한다. 산출된 w(Captions Weight)값을 가진 단어들은 Word2vec 방식을 이용한 핵심 문장 추출을 위해 Keyword DB에 강좌의 고유 번호(Content ID)와 함께 저장된다. [그림1]은 키워드 처리하는 모델이다.



[그림1] 강좌의 키워드 처리를 위한 모델 구조

2.2 핵심 문장 추출

핵심 내용을 판단하기위해 Word2vec 방식을 이용한다. 앞서 TF-IDF를 수행하고 키워드로 저장된 단어들은 관련된 문장을 검색하기 위해 Skip-gram 방식을 이용한다. 키워드를 입력 값으로 하여 주변 단어들을 검색 후 문장을 찾아내는 방식이다. 전체적인 문장을 검색하는 과정은 [그림2]와 같다.



[그림2] 키워드를 통한 문장검색 과정

Keyword DB에서 키워드를 불러오면 Word2vec으로 임베딩된 단어들 중 키워드 주변의 단어가 포함된 문장들을 추출한다. 강의 별 키워드의 수는 값에 따라 차이가 있을 수 있다. 키워드가 입력되면 주변 단어를 찾고 문장을 검색하여 핵심 문장들을 추출한다.

TF-IDF를 통한 자막들의 벡터화를 통해 자막 문서 안의 문장 벡터 값을 구한다. 추출된 문장들은 앞, 뒤 문장과의 코사인 유사도를 계산하여 핵심 문장의 부연설명을 해줄 문장인지 고려한 후 문장을 보충하게 된다. 키워드를 통한 문장이 추출되면 해당 문장의 앞, 뒤 문장의 유사도를 고려하여 문장을 추가한다. 키워드를 통해 추출되지 않아도 문맥상 유사한 문장이면 추출된 문장에서 내용을 추가한다.

3. 결 론

본 논문에서 키워드 추출을 TF_IDF 값을 통해 언급횟수에 따라 키워드로 판단하여 수행한다. 키워드를 통한 문장검색은 Skip-gram방식을 통해 키워드를 중심으로 주변 문장을 검색하는 것이다. 단순 검색만으로 부실 할 수 있는 내용을 추가하기 위해 코사인 유사도를 이용하여 부연 설명을 해줄 앞, 뒤 문장과의 관계를 측정하고 내용을 추가할 수 있게 한다.

시스템을 통해 추출된 문장들은 학습자가 온라인 교육 수강 중 이해하기 어려운 내용에 대해 참고할 만한 내용을 제안하기위해 시스템을 설계하였다.

이러닝의 목적은 원하는 정보를 쉽게 찾을 수 있고 효율적으로 시간을 투자하여 학습할 수 있는 강점을 갖고 있지만 정상적인 수강이 이루어지지 않거나 내용의 이해를 하지 못한다면 본래의 목적을 달성할 수 없다. 본 논문에서는 한계를 극복할 수 있는 해결방안으로 이 모델을 제안한다. 학습자가 이러닝 강좌를 수강하면서 시스템에서 제안하는 문장들을 활용하면 학습효과를 올리는데 기여할 것으로 보인다.

감사의 글

본 연구는 문화체육관광부 및 한국콘텐츠진흥원의 2017년도 문화기술 연구개발 지원사업으로 수행되었음[R2017030045]. 상명대학교의 재원으로 지원을 받아 산업연계교육활성화선도대학(PRIME) 후속사업의 일환으로 수행된 연구임.

4. 참고문헌

[1] 이러닝학회, “설립 취지”, URL:http://www.selearning.org/index.php?mid=page_SeYf02(2019. 12. 19)

[2] 허강호, 양진우, 김동현, 복경수, 유재수(2019). 단어 빈도와 유사도 분석 기반의 회의록 요약 시스템 설계 및 구현. 한국콘텐츠학회논문지, 19(10), 620-629, 2019

[3] 이말례, 배환국(2002). TFIDF 를 이용한 키워드 추출 시스템 설계. 인지과학, 13(1), 1-11.

[4] 박대서, 김화종(2018). TF_IDF 기반 키워드 추출에서의 의미적 요소 반영을 위한 결합벡터 제안. 한국정보기술학회논문지, 16(2), 1-16
[5] 이정훈, 정윤경(2019). Word2vec 임베딩을 이용한 책 추천 시스템. 한국정보과학회 학술발표논문집, 922-924
[6] 강형석, 양장훈(2019). Word2vec 모델로 학습된 단어 벡터의 의미 관계 분석. 정보과학회논문지, 46(10), 1088-1093, 2019

Python 코드를 위한 정적 분석기 개발

홍제성[1], 박보경[1*], 장우성[1**], 이원영[1***], 정세준[1+], 김영철[2]
홍익대학교 소프트웨어공학연구실
{hong[1], park[1*], jang[1**], leewy[1***], jeong[1+] , bob[2]}@selab.hongik.ac.kr

Development of Static Analysis Tool for Python Source Code

Je Seong Hong[1], Bokyung Park[1*], Woo sung Jang[1**], Won Young Lee[1***], SeJun Jung[1+],
R. Young Chul Kim[2]
SE Lab, Dept. of Software & Communications Engineering, Hongik University[1, 2]

요 약

앞으로의 AI 및 빅데이터 관련 소프트웨어의 오류, 오작동으로 인해 큰 인명 피해 등의 손실 발생이 가능하다. 특히 이들의 소프트웨어 코드의 크기와 내부 복잡도가 품질에 비중이 커질 것이다. 그러나 기존 절차식 언어 초점인 McCabe의 복잡도로는 새로운 Python 코드 복잡도 측정은 매우 부족한 실정입니다. 특히 거대한 소프트웨어의 비가시성은 내부 복잡성을 알기 어렵다. 또한 기존의 Python 코드 정적 분석 도구는 가시화 기능이 부족하다. 이를 위해, 소프트웨어의 내부를 가시화하고, 소프트웨어의 품질을 관리하고, 유지보수에 용이하기 위한 정석 분석기를 제안하다. 이를 기반으로 소프트웨어의 내부 구조 관계와 클래스, 모듈 내부의 복잡도 메트릭스 그리고 클래스 내부의 응집도를 측정하여 가시화를 함으로써 소프트웨어의 품질 향상에 기여한다.

1. 서 론

최근 4차 산업혁명과 더불어, 소프트웨어는 다양한 곳에 적용되며 그 비중 또한 매우 커지고 있다. 커진 소프트웨어의 비중만큼 소프트웨어의 오작동, 오류는 막대한 손실을 일으킨다. 국내 소프트웨어 산업은 열악한 개발환경, 인력, 비용 등의 문제로 제대로 된 소프트웨어 유지보수, 품질관리가 어렵다. 또한, 소프트웨어의 비가시성의 특징으로 복잡하고 크기가 큰 소스코드 내부를 쉽게 파악하기 어렵다. 이러한 문제를 해결하기 위해서, 소스코드의 내부를 가시화한다. 가시화한 결과를 토대로 소스코드의 수정, 관리에 도움을 줄 것으로 기대한다.

본 논문은 AI, Big Data에 주로 쓰이며, 계속해서 사용자가 많아지고 있는 Python 프로그래밍 언어를 대상으로 정적 분석 및 가시화를 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 소프트웨어 정적 분석 메트릭스, 소프트웨어 응집도, Python AST를 설명한다. 3장에서는 기존의 Python 정적 분석 도구와 본 논문에서의 분석 도구를 비교 및 설명을 한다. 4장에서는 도구 적용 사례를 설명한다. 마지막으로 5장에서는 결론 및 향후 연구를 설명한다.

2. 관련 연구

2.1 소프트웨어 정적 분석 메트릭스

소프트웨어의 코드 메트릭스는 소프트웨어 프로그램 코드의 복잡성, 유지보수성, 비용 등을 정량적으로 파악한다. 이를 통해, 잠재적 위험을 식별하고, 품질을 측정할 수 있다. 본 논문에서 활용한 정적 메트릭스는 다음과 같다.

2.1.1 McCabe's Cyclomatic Complexity

McCabe's Cyclomatic Complexity[1]는 소프트웨어 프로그램 코드의 복잡도를 측정하기 위해 프로그램의 제어 흐름을 기반으로 한다. 이 제어 흐름은 그래프(Graph: G), 선(Edge: e), 마디(Node: n)로 표현할 수 있다. 아래 그림[1]의 공식은 Cyclomatic Complexity의 공식이다.

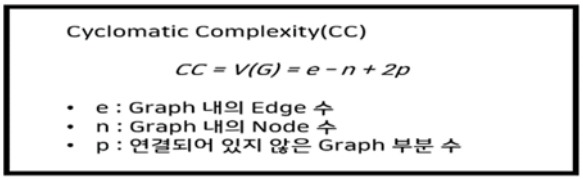


그림 1 Cyclomatic Complexity 공식

복잡도를 구하는 공식을 통해 계산된 복잡도 수치를 아래의 표[1]과 같이 A부터 F 등급으로 구분한다.

표 1 Cyclomatic Complexity's Rank

Cyclomatic Complexity (CC)	*Rank*
1 – 5	A
6 – 10	B
11 – 20	C
21 – 30	D
31 – 40	E
CC > 40	F

복잡도가 낮을수록 프로그램의 구조가 안정적이며, 높은 테스트 가능성과 낮은 위험도와 비용이 든다. 이와 반대로 복잡도가 높을수록 비 구조적임을 알 수 있다. 이는 낮은 테스트 가능성, 높은 위험도 및 비용이 든다.

2.1.2 Raw Metrics

Raw Metrics[2]는 원천 코드의 라인을 분석하는 것으로 아래 표[2]의 항목을 측정한다.

표 2 Raw Metrics 측정 항목 표

측정 항목	*내용*
LOC (Lines of Code)	코드 라인 수의 총합
LLOC (Logical Lines of Code)	코드의 논리적 라인 수
SLOC (Source Lines of Code)	코드의 소스 라인 수
Comments	주석의 라인 수
Multi	여러 줄로 나타낸 문자열 수
Blank	공백 라인 수
Single Comments	주석만 있는 라인의 수

2.1.3 Halstead Metrics

Halstead Metrics[3]는 프로그램의 코드가 만들어져 완성된 후 측정하는 방법이다. 코드 내부의 연산자와 피연산자의 수를 기반하여 프로그램의 논리의 규모를 산정하는 메트릭스이다. 아래의 표[3]는 Halstead Metrics의 연산자 추출 항목을 나타낸 것이다.

표 3 Halstead Metrics 연산자 추출 항목 표

추출 항목	*내용*
n_1	코드에서의 연산자 개수
n_2	코드에서의 피연산자 개수
N_1	연산자의 총 발생 빈도수
N_2	피연산자의 총 발생 빈도수

위 표[3]에 나타난 항목들의 관계를 통해 아래의 표[4]와 같이 프로그램의 복잡도를 측정한다.

표 4 Halstead Metrics 측정 항목 표

측정 항목	*내용*
Program vocabulary	$n = n_1 + n_2$
Program Length	$N = N_1 + N_2$
Calculated program length	$\hat{N} = n_1 log n_1 + n_2 log n_2$
Volume	$\mathrm{V} = N log_2 n$
Difficulty	$\mathrm{D} = \frac{n_1}{2} \times \frac{N_2}{n_2}$
Effort	$\mathrm{E} = \mathrm{D} \times \mathrm{V}$
Time required to program	$\mathrm{T} = \frac{E}{18}$
Number of delivered bugs	$\mathrm{B} = \frac{V}{3000}$

2.2 소프트웨어 응집도

소프트웨어 응집도(cohesion)는 모듈안의 요소들이 서로 연관되어 있는 정도로 모듈이 독립적인 기능으로 되어 있는 정도를 나타낸 것이다. 모듈의 응집도가 높을수록 독립적인 모듈이라 할 수 있다[4]. 여러 기능을 수행하는 경우 기능이 제대로 분산되지 않아 재사용하기 어렵다[5]. 따라서 소프트웨어는 모듈의 응집도가 높을수록 좋다.

본 논문에서 측정하는 응집도는 아래 그림[2]과 같이 Python 코드의 객체 설계 관점에서의 응집도를 측정한다.

```
class MyClass(): //클래스 MyClass

    class_var = 0 //클래스 멤버 변수 class_var

    def func1(self): //메소드 func1
        self.instance_var = 0 //인스턴스 변수 instance_var
        return class_var+self.instance_var
        //멤버 변수, 인스턴스 변수 사용

    def func2(self): //메소드 func2
        return self.instance_var+1
        //인스턴스 변수 사용
```

그림 2 클래스와 내부 변수와의 응집도

클래스의 메소드들이 클래스의 멤버 변수, 인스턴스 변수를 하나 이상 사용하는 것에 대한 응집도를 측정한다. 이는 클래스에 속한 메소드와 변수가 서로 묶여 의존하는 구조이기 때문이다.

아래의 표[5]는 이외의 응집도의 종류와 정의를 나타낸 것이다.

표 5 응집도의 종류와 정의

응집도	*정의*
기능적 응집도	모듈 내부의 모든 기능 요소들이 단일 문제와 연관되어 수행될 경우의 응집도
순차적 응집도	모듈 내의 하나의 활동으로부터 나온 출력 데이터를 그 다음 활동의 입력 데이터로 사용할 경우의 응집도
교환(통신)적 응집도	동일한 입력과 출력을 사용하여 서로 다른 기능을 수행하는 요소들이 모인 경우의 응집도
절차적 응집도	다수의 관련 기능을 순차적으로 수행하는 경우의 응집도
시간적 응집도	특정한 시간대에 수행되는 몇 개의 기능들이 모여 모듈화 된 경우의 응집도
논리적 응집도	모듈의 요소가 논리적으로 관계가 있는 경우의 응집도
우연적 응집도	서로 관계가 없는 요소가 모여 모듈을 구성하는 경우의 응집도

2.3 Python 추상 구문 트리(Abstract Syntax Tree)

추상 구문 트리(Abstract Syntax Tree, AST)는 프로그램의 소스코드의 구조를 트리형태의 자료구조로 나타낸 것이다. 이 추상 구문 트리로 심볼 테이블을 구성하여 바이트코드를 생성할 수 있다[6]. Python 언어에는 이 추상화 구문 트리를 생성하는 ast 모듈이 내재되어 있다. ast 모듈의 사용 예시로, 아래 그림[3]의 Python 소스 코드를 ast 모듈로 파싱을 하면 다음과 같이 구분되어 파싱된다.

```
def my_func(a, b):
    """
    Function example
    """
    return a + b
```

그림 3 Python 예시 소스 코드

```
Module(body=[FunctionDef(name='my_func',
args=arguments(args=[arg(arg='a',
annotation=None), arg(arg='b', annotation=None)],
vararg=None, kwonlyargs=[], kw_defaults=[],
kwarg=None, defaults=[]),
body=[Return(value=BinOp(left=Name(id='a',
ctx=Load()), op=Add(), right=Name(id='b',
ctx=Load()))))], decorator_list=[], returns=None)])
```

아래 그림[4]는 Python 소스코드를 ast 모듈을 사용해 파싱한 추상 구문 트리를 그림으로 표현한 것이다.

ast 모듈을 사용하여 코드의 정적 분석에 필요한 정보를 분석하여 원하는 정보를 추출을 통해 분석기를 구현할 수 있다.

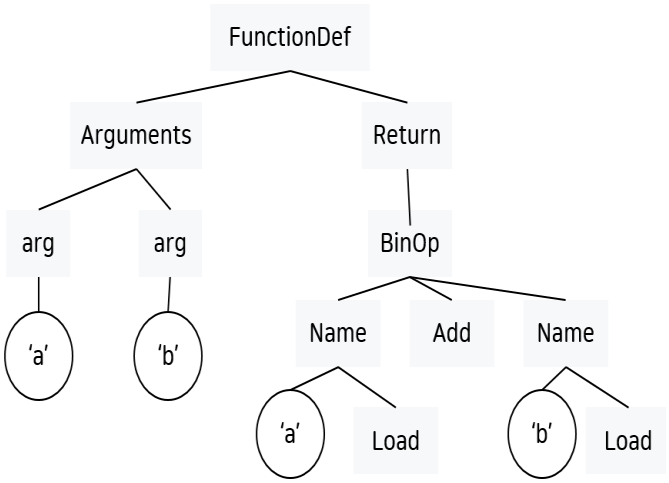


그림 4 추상 구문 트리의 결과 그림

3. 본론

3.1 기존의 Python 코드 정적 분석기와의 비교

기존에 있던 Python 코드 정적 분석기는 Source Meter와 Sonarqube 가 있다. 이 분석기들은 C/C++, Java, C#, Python 등의 프로그래밍 언어의 정적 소스코드 분석을 위한 도구이다. 정적 분석에 사용되는 도구인 Cppcheck, PMD, FindBugs, FxCop, Pylint 등의 도구들을 포함한 기능을 수행한다. 이와 함께, Python 코드의 각 모듈 사이의 호출관계, 호출 횟수, 소요시간, 메모리 사용량을 그래프화 하여 그리는 패키지인 python call graph가 있다.

Source meter와 Sonarqube는 Python 코딩 룰 체크, 코드 복잡도의 정도를 수치화 하여 보여주는 것에 그친다. python call graph는 모듈 간의 호출 관계, 횟수, 소요 시간, 메모리 사용량 정보를 그래프화 하여 보여주는 데에 그친다. 그러나, 본 논문의 python 코드 정적 분석기는 기존의 분석 도구들과는 다르게 모듈 호출 관계, 횟수, 호출에 필요한 파라미터 개수와 각각의 복잡도 메트릭스(McCabe's complexity, Raw metrics, Halstead metrics) 계산 정보, 응집도 정보를 그래프에 나타낸다. 그려낸 그래프에서 모듈 간의 호출 횟수가 설정한 기준보다 많이 호출되는 호출 관계에는 빨간색 굵은 실선으로 표현하여 가시화한다. 그리고, 2장에서 언급한 복잡도의 기준을 알파벳으로 등급을 나누었듯, 코드의 복잡도가 C등급 아래인 구성요소들을 빨간색으로 표현하여 복잡도가 높다는 것을 가시화한다.

3.2 Python 코드 정적 분석기

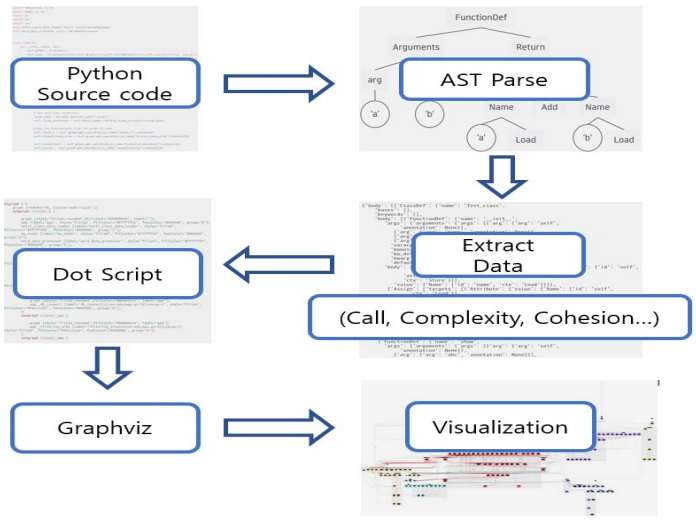


그림 5 Python 코드 정적 분석기 가시화 과정

그림[5]는 Python 코드를 정적 분석하여 가시화하는 과정을 나타낸 것이다. 1)먼저 분석하고자 하는 Python 코드 파일들을 입력한다. 2)입력된 코드들을 AST 구조로 구성하여 파싱을 한다. 파싱된 정보들은 2.3절에 기술된 것과 같이 ast 모듈을 사용하여 값을 반환받는다. 모듈, 클래스, 함수 내부의 구조를 추출할 수 있다. 3)ast 구조로 파싱된 결과를 통해 호출 정보,

복잡도 정보, 응집도 정보 등 필요한 정보를 추출한다. Cyclomatic 복잡도의 정보는 파싱한 ast 구조에서 분기점이 되는 항목을 찾는다. 기본적인 복잡도 수치는 1이다. 복잡도를 만드는 추출 항목은 다음과 같다. if, elif, for, while, except, with, assert, Comprehension, Boolean Operator. 이러한 구조가 발견되면 Cyclomatic 복잡도를 1씩 증가하는 계산을 한다. Raw, Halstead 복잡도 추출도 역시 ast 파싱 결과를 통해 2장에 언급된 측정 항목 표의 수식으로 계산하여 프로그램의 라인 수, 크기 등의 항목을 추출한다.

응집도 항목은 클래스와 그 내부의 변수 사이의 응집도를 계산한다. 클래스에 생성된 멤버 변수, 인스턴스 변수의 개수를 계산하며, 각 함수에서 몇 개의 클래스에서 생성된 변수를 사용하는지를 계산한다. 4)추출된 정보를 사용해 그래프를 그리기 위한 Dot Script를 생성한다. 5)생성된 Dot Script를 Graphviz[7]에 입력하여 그래프를 생성한다. 6)코드 정적 분석을 위한 가시화 결과를 나타낸다.

4. Python 코드 정적 분석기 활용 사례

분석으로 입력된 프로그램은 Text-CNN을 활용한 교육 프로그램 추천 프로그램[8]의 한 부분이다. 각 클래스, 함수들은 타원형으로 표현된다. 각각의 그룹화를 위해 클래스와 그 내부에 생성된 메소드는 같은 계열의 색으로 표현되며, 채도가 더 진하게 표현된다. 클래스, 함수의 이름과 호출에 필요한 파라미터 개수, 각 모듈, 클래스, 클래스 내부에서의 호출 관계를 실선으로 표현한다. 호출 횟수는 실선위에 숫자로 표기가 된다.

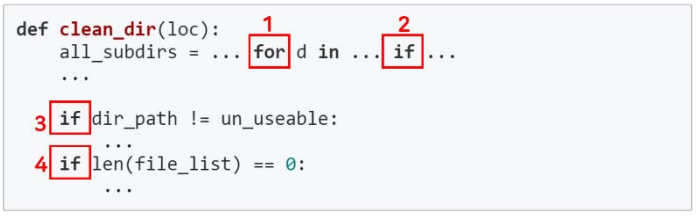


그림 7 clean_dir 함수의 구조

그림[7]은 clean_dir 함수의 구조이다. 이 함수의 Cyclomatic 복잡도는 for문 1개 if문 3개의 구조로 이루어져 5점의 값인 A등급이 표시된다. 6)그러나 전달받은 값을 특정 조건에 맞게 필터링하는 filtering_cnn_rate 함수는 Cyclomatic 복잡도가 높게 측정된다.

그림[8]은 filtering_cnn_rate 함수의 구조를 나타낸 것이다. 이 함수의 제어 흐름 구조는 총 10개가 추출되어 11점의 C등급으로 빨간 칸으로 표시된다. 이러한 가시화 결과를 통해 프로그램의 어떤 함수에서 복잡도를 높이고 있는지 알 수 있다.

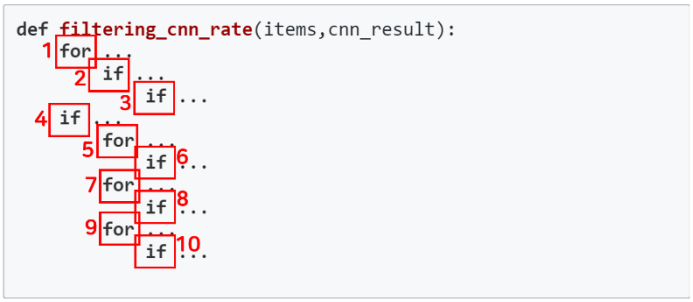


그림 8 filtering_cnn_rate 함수의 구조

5. 결론 및 향후 연구

본 논문은 Python 언어로 된 프로그램의 정적 분석을 통한 가시화를 수행한다. 소프트웨어의 품질을 높이기 위해 모듈 간, 모듈 내부에서의 호출, 복잡도, 응집도를 가시화한다. 분석한 가시화 결과를 사용자에게 제공함으로써 프로그램의 구조를 알 수 있으며, 소스 코드의 어느 부분을 수정하여 복잡도를 낮추고 품질을 더 높일 수 있을지의 척도가 될 수 있다. 향후에는 더 많은 응집도와 결합도를 추가적으로 확장하여 연구할 예정이다.

Acknowledge

본 논문은 2019년도 산업통상자원부의 '창의산업융합 특성화 인재양성사업'(과제번호 N0000717)과 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2017R1D1A3B03035421)

6. 참고 문헌

[1] T.J.McCabe, "A Complexity Measure", IEEE Trans. SE, Vol.2, No.6, pp.308-320, Dec. 1976.
[2] Radon, https://radon.readthedocs.io
[3] Halstead, Maurice H. (Maurice Howard) Elements of software science. Elsevier, New York, 1977.
[4] Chidamber Shyam and kemerer Chris, "A metrics suite for object oriented design", IEEE Transactions on Software Engineering, Vol. 20, No. 6, pp476-493, 1994.
[5] E.-Y. Byun, H.-S. Son, S.-Y. Moon, W.-S. Jang, B.-K. Park, and R. Y. Kim, "정적/동적 분석 기반의 재사용 메트릭과 가시화 구축", 한국정보처리학회, Vol. 24, No. 1, pp.621-624, 2017.
[6] Python ast, https://docs.python.org/3/library/ast.html
[7] Graphviz, https://www.graphviz.org/
[8] 홍제성, 박보경, 곽제일, 손현승, 김영철, "Text-CNN 알고리즘 적용한 교육장터 플랫폼 기반 맞춤형 교육 컨텐츠 추천 메커니즘 개발", 한국정보처리학회, Vol. 26, No. 2, pp. 965-967, 2019.

블록체인 코드의 복잡도 측정을 위한 코드 가시화

안현식[1], 박지훈[2], 김기두[2*], 박보경[1], 조재형[1], Md Ibrahim Khalil[1], 김동호[3], 김영철[1*]
홍익대학교 소프트웨어공학연구실[1], 한국정보통신기술협회(TTA)[2],
데이터메트릭스(Datametrex)[3]
{ahn[1], park[1], cho[1], Ibrahim[1], bob[1*]}@selab.hongik.ac.kr

Code Visualization for Measuring Complexity of Blockchain Code

Hyun-sik Ahn[1], Jihoon Park[2], Kidu Kim[2*], Bokyung Park[1], Jae Hyeoung Cho[1], Md Ibrahim Khalil[1], Stephen D. Kim[3], R. YoungChul Kim[1*]
SE Lab[1], Hongik University
TTA[2], Datametrex[3]

요 약

현재 4차 산업 혁명 시대에 다양한 영역의 많은 소프트웨어에대해 고품질 고려가 미비하다. 특히 블록체인 관련 소프트웨어 품질이 중요한 이슈가 될 것이다. 기존의 소프트웨어 개발보다 더 많은 품질 요소(병렬, 동시화, 성능, 전력)가 필요하다. 또한 새롭게 등장한 Go 기반 블록체인 코드에 대한 분석기의 부재와 모듈 내 두 리턴 값으로 내부 코드의 복잡도가 매우 높다. 이를 해결하기 위해 Go기반 코드 정적분석기를 제안한다. 이를 기반으로 코드 내부 복잡도 추출 및 가시화하여 복잡도 개선 방법을 제안한다. 이런 코드 가시화 방법은 보다 쉽게 코드를 유지보수 할 수 있으며 이는 블록체인 시스템의 소프트웨어 고품질화가 가능하게 된다.

1. 서 론

최근 4차 산업혁명이 가장 큰 이슈로 떠오르면서 AI, 자율주행, 블록체인 등 새롭고 다양한 시스템들이 등장하고 있다. 그 중 수많은 블록체인 기반 플랫폼들이 개발되고 있지만, 블록체인 소프트웨어의 신뢰성, 가용성, 데이터 무결성 확보 등에 대한 검증은 누구도 하고 있지 않다. 이처럼 고품질 소프트웨어와 거리가 먼 구현 위주의 개발을 하게 되면 시스템 전체의 복잡도가 높아지고 유지보수가 어렵게 된다. 복잡도가 지나치게 높다면 전체적인 소프트웨어의 아키텍처를 파악해 복잡도를 줄이고 모듈간의 결합도를 줄여 모듈간의 독립성을 만들어야 한다. 이렇게 소프트웨어 공학적 접근을 통한 아키텍처가 가시화됨으로써 전체 시스템의 복잡도를 파악할 수 있고 모듈간의 결합도를 판단할 수 있다.

본 연구에서는 블록체인 코드의 정적 분석을 통해 복잡도, 결합도를 가시화하는 방법을 제안한다. 2장에서는 관련연구로 아키텍처 가시화와 결합도, Go AST Viewer에 대해 설명한다. 3장에서는 Go language로 구현된 소프트웨어 가시화 과정에 대해 설명하고 4장에서는 결합도 측정 방법에 대해서 설명한다. 5장에서는 실제 프로그램에 대한 적용사례를 보여준다. 6장에서는 결론 및 향후 연구에 대해서 언급한다.

2. 관련연구

2.1 아키텍처 가시화

소프트웨어 아키텍처 가시화는 소프트웨어를 개발하는데 가장 어려운 소프트웨어의 비가시성을 극복할 수 있다. 이를 통해 소프트웨어의 구조를 파악함으로써 비전문가도 고품질을 위한 소프트웨어 개발의 품질 관리가 가능하다[1,2]. 코드를 정적 분석하여 클래스, 변수, 함수, 등으로 구분 지어 데이터를 추출한다. 이 데이터들을 데이터베이스(SQLite)에 저장하고, 필요한 정보를 쿼리문으로 SELECT하고 이 데이터들로 Dot script를 작성한다[3,4]. 작성된 Dot script는 Graphviz에서 그래프로 그려진다.

2.2 Go AST Viewer

Go AST Viewer는 Go 코드를 Abstract Syntax Tree 구조로 가시화하여 보여주는 웹 기반의 오픈소스이다. 소스코드를 파싱하기 필요한 해당 문구의 AST Node 정보를 얻는데 사용된다. 해당 Node의 정보로 Go Parser 패키지 내의 inspect 함수의 모든 노드 깊이 우선탐색 과정을 통해 필요한 Node를 조건문으로 추출한다[5].

2.3 결합도

결합도는 모듈과 모듈 사이 연관관계 또는 상호의존관계를 의미한다. 재사용성 관점에서 바라봤을 때 결합도가 강할 경우, 모듈간의 의존성이 높아 변경 및 유지보수가 힘들어진다. 결합도가 약할 경우, 모듈간의 상호 의존성이 줄어들어 모듈간의 독립성이 높아지고, 독립성이 높으면 모듈간의 영향이 적어 유지보수하기 좋은 소프트웨어이다. 결합도는 Data, Stamp, Control, External, Common, Content 총 6개의 종류가 있다. Data 결합도일수록 모듈간 독립성이 높고, Content 결합도일수록 모듈간의 상호의존성이 높다.

3. Go language 가시화 과정

다음 그림 1은 Go Language 소프트웨어 가시화의 전 과정을 보여준다.

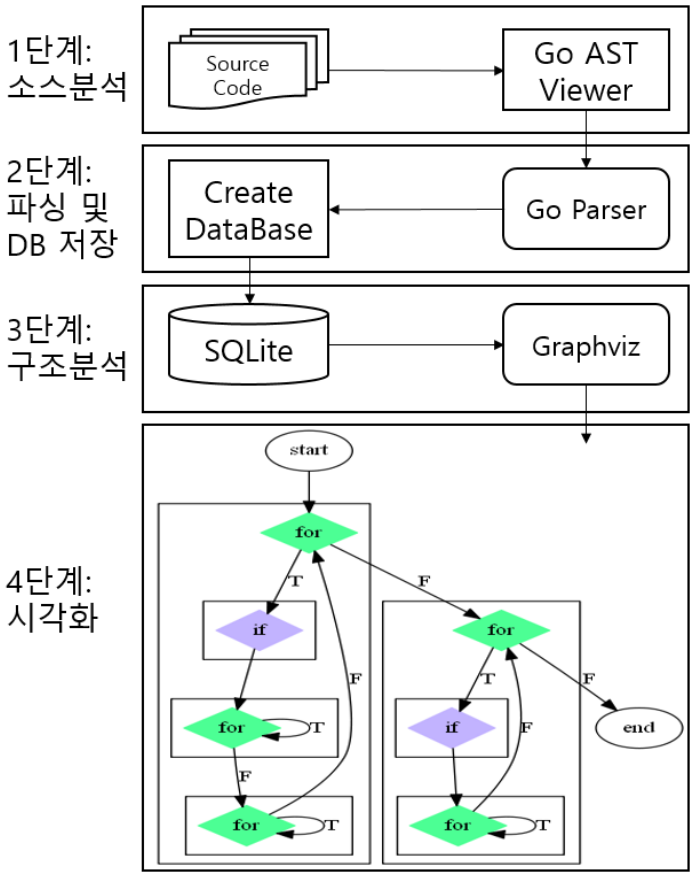


그림 1 Go Language 가시화 과정

소프트웨어 가시화를 위해서는 소스분석, 파싱 및 데이터베이스에 저장, 구조분석, 시각화 4가지 단계로 구성된다. 다음 그림 2는 Go AST Viewer를 사용한 파싱 과정이다. 그림 2의 ①처럼 Go 언어로 이루어진 코드를 Go AST Viewer에 입력하고 Parse 버튼을 누르면 각 문구의 AST Node 구조가 나온다. 그림2의 ②에서 확인 할 수 있듯이 ast.Ident 노드는 해당 코드의 packge name의 정보를 갖고 있는 노드인걸 알 수 있다. 이렇게 찾은 정보를 파싱 및 데이터베이스 저장 단계에서 그림 2의 ③처럼 Go Parser 내부의 inspect함수의 모든 노드 깊이 우선 탐색과정으로 조건문을 통해 추출한다. 이런 방식으로 결합도를 도출하기 위해 필요한 데이터를 저장할 수 있도록 정제과정을 거친다. 구조 분석 단계에서는 저장된 데이터들 중 가시화 과정에서 필요한 정보들을 쿼리문으로 SELECT하여 Dot Script를 작성한다. 시각화 단계에서는 작성된 스크립트를 통화여 이미지화 한다[6].

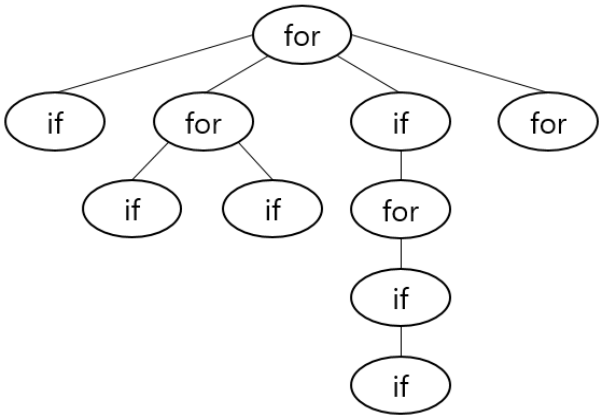


그림 3 Flow Chart를 그리기 위한 함수 내 구문 트리 구조

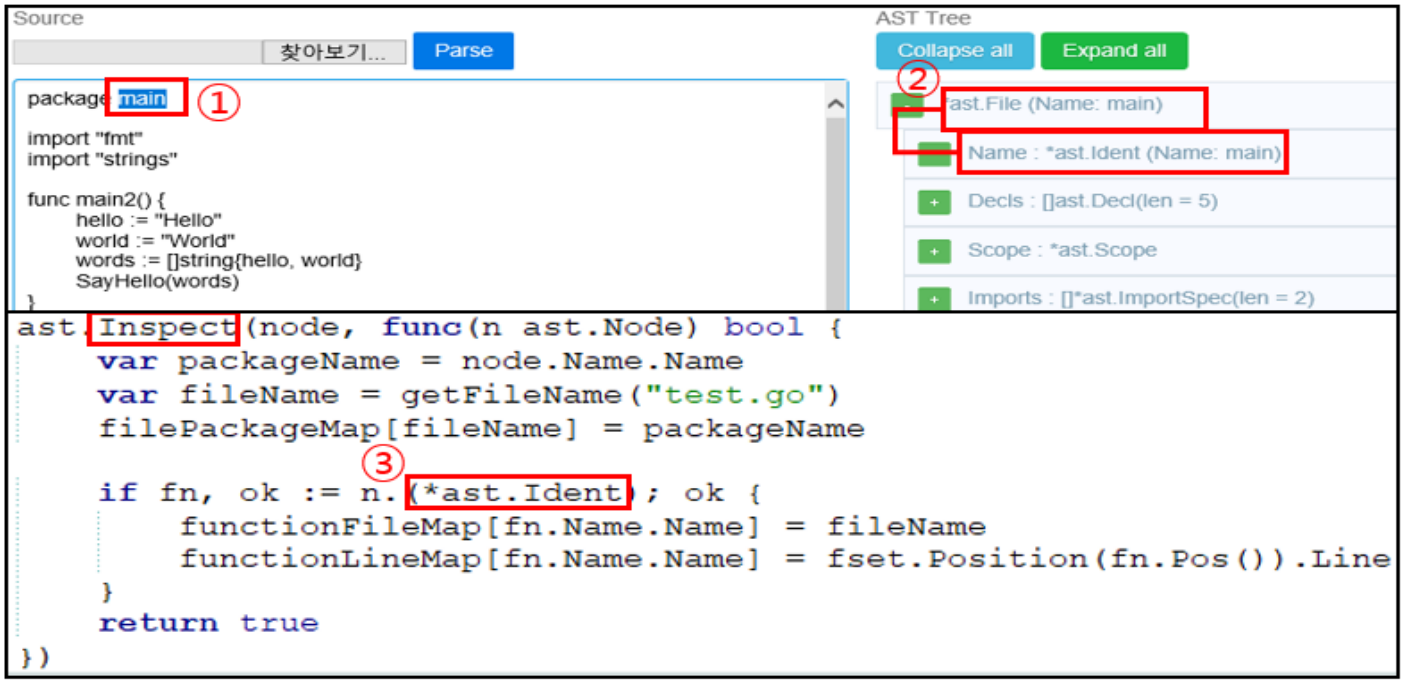


그림 2 Go AST Viewer를 사용한 파싱 과정

4. 결합도 측정 방법

Go 코드 내에서 함수가 호출할 때, 호출될 때 서로 주고받는 매개변수의 종류에 따라서 결합도를 측정할 수 있다. 아래의 그림 4는 Data Coupling에 대한 예제 코드와 Data Coupling 일 때 그려지는 그래프이다.

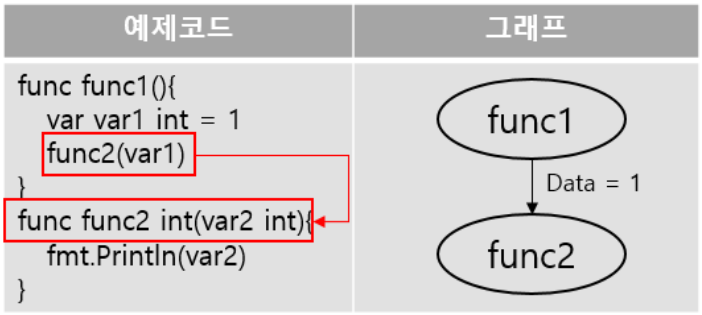


그림 4 Data Coupling 예제코드 및 그래프

아래의 그림 5는 Stamp Coupling에 대한 예제 코드와 그래프이다.

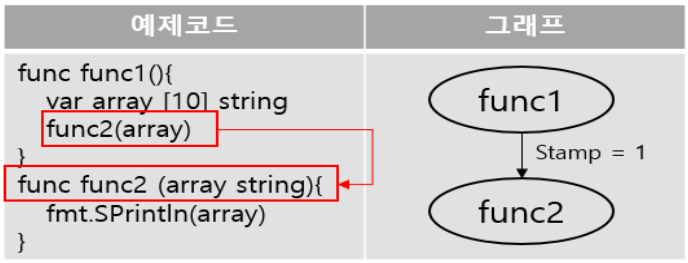


그림 5 Stamp Coupling 예제코드 및 그래프

아래의 그림 6은 Control Coupling에 대한 예제 코드와 그래프이다.

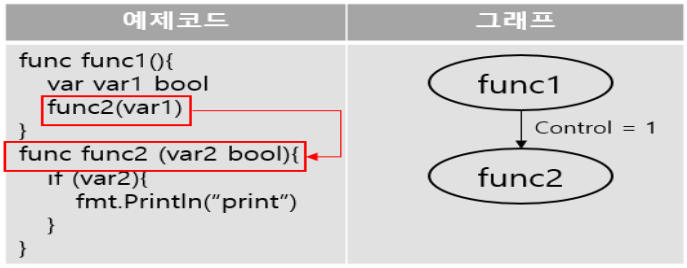


그림 6 Control Coupling 예제코드 및 그래프

아래의 그림 7은 External Coupling에 대한 예제 코드와 그래프이다.

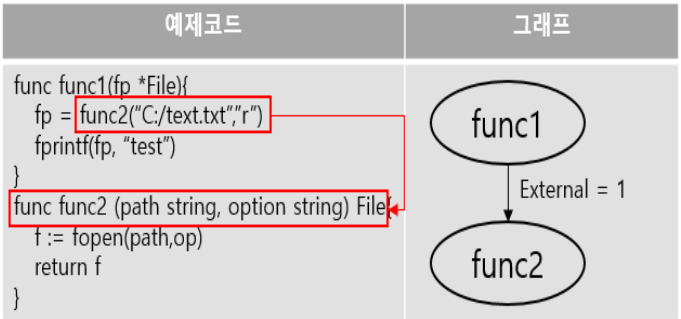


그림 7 External Coupling 예제코드 및 그래프

다음의 그림 8은 Common Coupling에 대한 예제코드와 그래프이다.

그림 8 Common Coupling 예제코드 및 그래프

다음의 그림 9는 Content Coupling에 대한 예제 코드와 그래프이다.

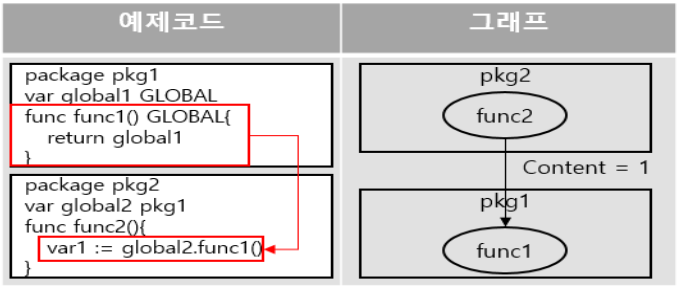


그림 9 Content Coupling 예제코드 및 그래프

예제코드에 pkg1, pkg2라는 패키지 2개가 있다. pkg1에는 GLOBAL 타입의 전역변수 global1과 func1 함수가 있고 func1은 global1을 반환한다. pkg2의 func2 함수는 var1이라는 GLOBAL 타입을 선언할 때 pkg1 패키지의 전역변수 global1을 직접 불러오지 않고 func1의 반환 값을 통해 가져온다. 이와 같이 직접 선언하지 않은 변수를 다른 객체, 함수의 반환 값 등을 통해 호출하는 경우 Content Coupling으로 인식한다. 그래프는 Common Coupling과 마찬가지로 서로 다른 패키지를 구분하기 위해 패키지 그림이 추가된다.

5. 적용사례

다음 그림 10은 코드 가시화의 적용사례로 실제 Go language를 가시화 해주는 코드 중 데이터베이스를 연결해주고 데이터를 저장할 테이블을 create해주는 Database.go와 이를 실행 시키고 결합도를 측정해주는 main.go 두 파일을 사용했다.

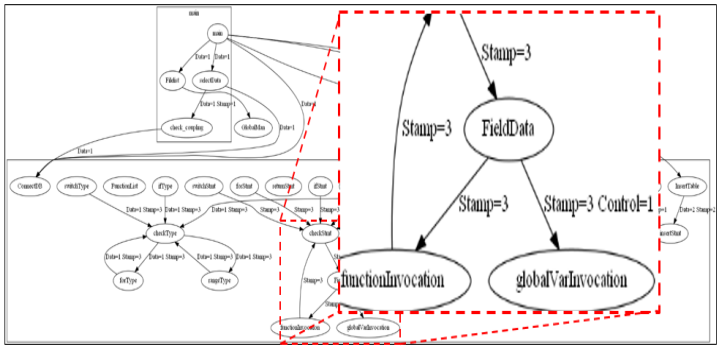


그림 10 리펙토링 전 결합도 가시화 그래프

가시화 그래프에서 확인할 수 있듯이 FieldData함수가 functionInvocation, globalVarInvocation함수를 호출할 때 각각 Stamp 결합도가 3번씩 발생한다. Stamp 결합도는 상대적으로 Data 결합도보다 높은 결합도이다. Stamp 결합도를 상대적으로 낮은 Data 결합도로 리펙토링하면 소프트웨어의 품질을 높일 수 있다. 하지만 롱 파라미터 리스트라는 나쁜 냄새를 유발할 수 있기 때문에 변경에 유의해야한다.

6. 결론 및 향후 연구

본 연구는 고품질 소프트웨어를 위해 Go 코드의 아키텍처 가시화를 통하여 모듈간의 결합도를 시각화하여 조사 분석한다. 이로 인해 어떤 모듈에서 높은 결합도가 존재하는지 알 수 있으며 이를 낮은 결합도로 리펙토링하여 결합도 수치를 감소시킨다. 이를 통해 평가 기준에 따라 해당 함수가 안정된 코드인지, 복잡한 코드인지를 확인한다. 현재 소프트웨어 고품질을 위한 지표로써 결합도 경우의 수를 가시화하여 전부는 아니어도 찾을 수 있었다. 향후 연구로 모든 가능한 경우의 수를 찾고, 응집도, Bad Smell등 다양한 지표들을 추가하여 연구하고, 다양한 사례에 적용하여 연구를 진행하고자 한다.

ACKNOWLEDGE

본 논문은 2019년도 산업통상자원부의 '창의산업융합 특성화 인재양성사업'(과제번호 N0000717)과 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2017R1D1A3B03035421)

참고 문헌

[1] NIPA SW Engineering Center "SW Development Quality Management Manual(SW Visualization)" 2013. 12.
[2] 박지훈, "정적 분석 기반 블록체인 코드 취약성의 자동 가시화 연구", 2018
[3] SQLite, https://www.sqlite.org/
[4] Graphviz, http://www.graphviz.org/
[5] Go AST Viewer, http://goast.yuroyoro.net
[6] 안현식, 박지훈, 박보경, 김영철, "Go language로 구성된 블록체인 코드를 측정하기 위한 효과적인 코드 정적분석기" 2019. 12.

‘아동 및 어린이’ 키워드를 이용한 뉴스 기사 분석

최은지*, 박지연**, 김용환*, 노기섭**
*청주대학교 문헌정보학과, **청주대학교 소프트웨어융합학부
(cake5105, withtaylors, kimyounghwan, kafa46)@cju.ac.kr

Analysis on News Articles using Keyword ‘Infant and Children’

Eunji Choi*, Jiyeon Park**, Yonghwan Kim*, Giseop Noh**
*Dept. of Library & Information Science, Cheongju University
**Dept. of Software Convergence, Cheongju University

요 약

본 연구에서는 다양한 언론사로부터 수집한 최근 1년간 뉴스에서의 ‘아동, 어린이’에 대한 연관어와 키워드 트렌드를 알아보고 그에 따른 사람들의 관심도를 파악한다. 해당 키워드 수집 및 분석을 위하여 공공데이터 플랫폼을 이용하여 데이터를 수집하고 텍스트 데이터의 트렌드를 분석한다. 분석결과를 직관적 이해를 돕기 위해 추가적인 시각화를 수행하고, 최종적으로 뉴스기사 키워드 분석을 통한 트렌드 분석 결과를 제시한다.

1. 서 론

최근 아동과 관련한 여러 사건이 이슈화되고 있다. ‘한음이, 해인이, 하준이, 민식이, 태호 유찬이’ 이 아이들은 특별히 위험한 곳이 아니라, 통학버스, 주차장, 스쿨존 같은 우리 주변의 일상적인 곳에서 날벼락 같은 죽음을 맞이했다. 2015년 ‘세림이법’이 시행됐지만, 어린이 통학 안전사고와 교통사고가 끊이지 않고 있다. 그럴 때마다 피해 부모님들은 ‘다시는 이런 일이 일어나서는 안 된다’며 법률 개정을 위해 거리에서 전단을 뿌리고 때로는 청와대 청원에 나섰다. 하지만 <어린이 생명안전법안>은 여전히 각 정당의 주요 관심사에 밀려 논의조차 되고 있지 못한 상황이었다. 하지만 끊임없는 청원과 부모들의 노력으로 2019년 9월 충남 아산의 한 어린이보호구역(스쿨존)에서 교통사고로 사망한 김민식 군(당시 9세) 사고 이후 발의된 법안이 12월 10일 국회 본회의를 통과했다. 문 대통령은 “국민은 재난에서 안전할 권리, 위험에서 보호받을 권리가 있고, 국민의 생명과 안전에 대한 국가의 책임은 무한하다”고 강조했다. 이러한 일들로 인해 아동에 대한 법안들이 주목을 받고 있다.

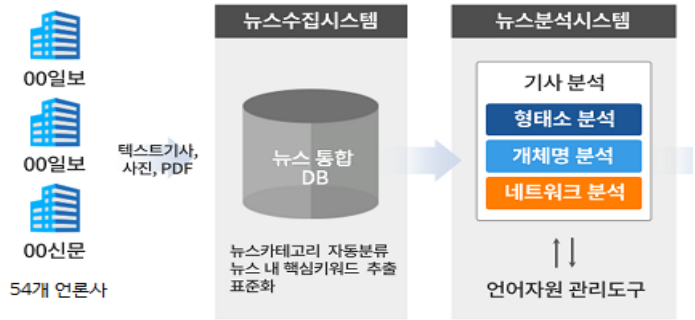
인터넷 데이터의 수집에 있어서 수집 키워드의 선정은 매우 중요하다[1]. 키워드는 연구의 목적과 취지에 맞게 선정되어야 하며 타당성을 갖추어야 한다. 따라서 본 논문에서는 ‘아동, 어린이’에 대한 연관어와 키워드 트렌드를 알아보고 그에 따른 사람들의 관심도를 파악할 것이다. 이와 같은 목표를 달성하기 위해 본 연구는 인터넷에서 언급된 콘텐츠를 수집하고자 한다. 일반인에게 아직 알려지지 않은 새로운 소식이나 일반적으로 시사성(時事性)이 있다고 판단되는 보도내용을 뉴스라고 한다. 뉴스 기사를 분석함으로써 키워드가 왜 이슈가 되었

는가에 대한 결과가 나타날 수 있다[2]. 본 연구는 '아동, 어린이'를 키워드의 주제로 한 뉴스 기사들을 수집하였다. 초기에는 '어린이'라는 키워드만 사용하였으나, 이후 그와 동의어인 '아동' 이라는 키워드를 함께 사용하여 더욱 구체적인 자료들을 분석하였다.

2. 연구 방법

본 연구에서는 다양한 언론사로부터 수집한 뉴스로 구성된 통합 데이터베이스에 빅데이터 분석 기술을 접목하여 만든 새로운 뉴스 분석 서비스인 빅카인즈[1] 서비스와 텍스트마이닝을 바탕으로 연구를 하였다[3].

본 논문에서 활용한 빅카인즈 시스템은 그림 1과 같이 뉴스 수집시스템, 분석시스템, 저장시스템 등으로 구성되어 있으며, 저장된 뉴스 분석 정보는 국민, 언론사, 학계, 스타트업 등이 활용할 수 있는 빅데이터들을 분석할 수 있다[4]. 가장 큰 특징으로는 비정형 텍스트를 분석이 가능한 정형화된 데이터로 바꾸어, 사회현상을 분석할 수 있는 기초 자료를 제공받을 수 있다는 것이다. 본 연구에서 수집하는 뉴스는 인터넷 뉴스 기사로 대부분 비정형 문서로 이루어져 있다. 본 논문에서는 비정형 문서들에서 본 연구에서 유용한 정보들을 추출한다. 특히, '아동, 어린이'라는 키워드에 대한 키워드 트렌드(검색한 키워드가 포함된 뉴스 건수를 일간/주간/월간/연간 그래프로 제공하는 서비스)를 활용하였다.



[1] 빅카인즈: https://www.bigkinds.or.kr

그림 1. 빅카인즈 서비스 개념도

Fig. 1 Conceptual Diagram of the Big Kinds Service

3. 분석 결과

본 논문에서는 일간지 11곳(경향신문, 국민일보, 내일신문, 동아일보, 문화일보, 서울신문, 세계일보, 조선일보, 중앙일보, 한겨레, 한국일보)에서 '아동, 어린이'를 검색하였을 때 나타나는 2018년 11월 1일부터 2019년 11월 30일 까지의 1년치 콘텐츠 내용들을 수집하였다.

3.1 키워드 트렌드

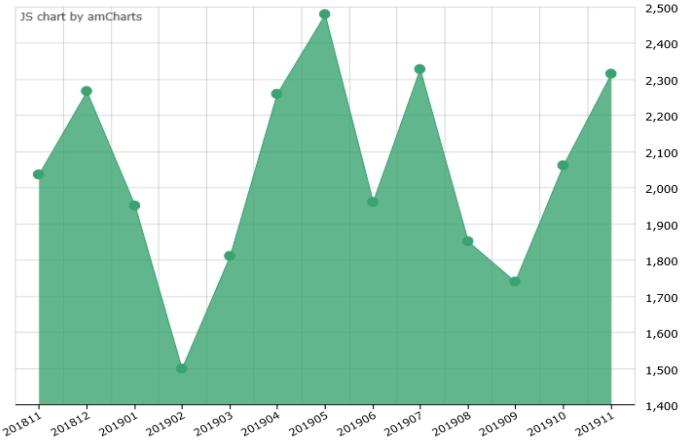


그림 2. 키워드 트렌드 월간 그래프 '아동, 어린이' 키워드를 이용한 뉴스 기사 건수

Fig 2. Monthly graph of the number of news articles using keywords 'infant, infant' from Keyword trending

그림 2에서 2018년 12월까지는 단어의 출현빈도가 잠시 증가하는 것으로 보인다. 그러나 빈도수가 급격히 감소하여 2월달에 최저치를 기록하였다. 이후 다시 빈도수가 증가하여 5월달에 최고치를 기록하였고 그 뒤로는 사건사고에 따라 빈도수의 증감이 계속되었다. 9월에 들어서는 다시금 출현 빈도가 증가하고 있으며 최근에 일어난 어린이집 성폭행 사건과 '해인이법' 등을 비롯한 사건들에 대한 사람들의 관심이 모아지면서 12월달에도 계속해서 증가할 것으로 예상된다.

3.2 빈도 기반 분석 워드 클라우드

그림 3. 2018년 11월 워드클라우드

Fig 3. WordCloud in November 2018.

그림 4. 2019년 11월 워드클라우드

Fig 15. WordCloud in November 2019.

본 논문에서는 분석 결과를 시각화하기 위하여 워드클라우드를 사용하였다. 워드 클라우드는 단어 빈도에 따라 글자 크기와 위치를 배치해 주는 자동화 도구이다.

순위	1위	2위	3위
'18년 11월	유치원	보고서	학부모
'18년 12월	산타	도서관	성범죄
'19년 1월	중국	미세먼지	김씨
'19년 2월	스쿨미투	성폭력	유치원
'19년 3월	미세먼지	장애인	유튜브
'19년 4월	아이돌보미	김씨	피해자
'19년 5월	방정환	청와대	보건복지부
'19년 6월	유튜브	장애인	성폭행
'19년 7월	피해자	배스킨라빈스	성관계
'19년 8월	리얼돌	성범죄	인천
'19년 9월	유튜브	도서관	장애인
'19년 10월	음란물	손씨	운영자
'19년 11월	음란물	민식이법	스쿨존

표 1. 월별 키워드 상위 3위('아이들' 키워드 제외)

워드클라우드 시각화 결과를 그림 3과 4에 제시하였다. 그림 3은 2018년 11월의 뉴스 중심을 표현한 것으로 '아이들','유치원','보고서' 등이 자리 잡고 있다. 그림 4의 경우 '아이들', '민식이법', '음란물' 등에 대한 뉴스가 중심을 이루고 있음을 보여주고 있다. 2018년 12월부터 2019년 10월까지의 워드클라우드는 지면 관계상 생략한다. 시간의 흐름에 따른 관심도 분석을 위해 월별 빈도기반 분석을 위해 월별로 '아동, 어린이' 키워드에 대한 최상위 키워드를 표 1에 정리하였다. 키워드 '아동'은 뉴스 데이터 추출 시 검색 키워드로 사용하였으므로 모든 기간에서 최고 빈도수를 차지하였다. 검색 키워드에 대한 자명한 빈도수 순위이므로 표 1의 현황에서는 제외하였다. 시각화를 통한 직관적 분석을 위해 표 1을 빈도 그래프로 그리면 그림 5와 같다. 그림 5의 세로축 값의 의미는 3: 해당 기간의 최다 출현, 2는 상위 2번째 빈도, 1은 상위 3번째 출현 빈도를 의미한다.

그림 5. 월별 키워드 순위 ('19. 3월 - '19. 11월)

월별 최상위 출현 단어를 살펴보면 '아동, 어린이' 관련 키워드의 연관 분석 결과 성관련 키워드('음란물, 리얼돌, 성폭행, 성관계)가 비교적 자주 그리고 장기적으로('19년 6월, 7월, 8월, 10월, 11월) 등장한 것으로 보아 아동과 관련된 성폭력/성희롱에 대한 관심이 높은 것을 확인할 수 있었다. 또한 추가적으로 '유튜브'에 대한 연관 키워드가 지속적으로 등장하고 있는 것으로 보아 아동 문제와 온라인 정보제공 매체와 '아동, 어린이'와 일정부분 관계성이 존재함을 확인하였다.

4. 결 론

본 논문에서는 최근 1년간 뉴스에서의 '아동, 어린이'에 대한 연관어와 키워드 트렌드를 알아보고 그에 따른 사람들의 관심도를 파악하였다. 이와 관련하여 빅카인즈의 키워드트렌드와 워드 클라우드의 분석을 적용하였으며, 최근 1년(2018.11.1~2019.11.30) 동안 뉴스에서 '어린이, 아동'과 연관어로 가장 많이 언급된 키워드는 '음란물, 성폭력, 성 보호, 성범죄, 피해자' 등으로 부정적 단어들과 관련이 있었으며 이를 통해 지난 1년간 아동 성관련 문제점이 사회적으로 지속되고 있음을 확인하였다.

본 논문에서는 키워드 트렌드에 대한 정량화 분석은 수행되지 못하였다. 향후 추가 연구를 통하여 키워드를 범주화 하고 특정 범주에서 지속적으로 등장하는 다빈도 키워드에 대한 분석을 수행할 예정이다.

Acknowledgement

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2018R1C1B5083091).

참고 문헌

[1] 송은지, "빅 데이터를 이용한 고객평판 사례분석에 관한 연구," *한국정보통신학회논문지,* vol. 17, no. 10, pp. 2439-2446, 2013.

[2] 유홍연, 이승우, and 고영중, "실시간 뉴스 기반의 이슈 분석을 위한 점증적 군집화 및 다중 문서 요약," *정보과학회논문지,* vol. 46, no. 4, pp. 355-362, 2019.
[3] M. Song, *Text Mining*. South Korea: Chungram, 2017.
[4] 이은별, 전진오, and 백지선, "서울의 다문화 공간 연구: 뉴스 빅데이터 분석 시스템 '빅카인즈'를 이용한 국내 언론의 외국인 마을 보도 (1990～ 2016) 분석," *미디어 경제와 문화,* vol. 15, no. 2, pp. 7-43, 2017.

IoT 운영체제 모델 기반 파밴드 어플리케이션 안전성 검증

양송 ○, 신영술*, 김동우, 김동영, 최윤자

경북대학교 컴퓨터학부, 경북대학교 자율군집 소프트웨어 연구센터*

yangsong666@knu.ac.kr, youngsulshin@gmail.com,
kdw9242@gmail.com, withdongyeong@gmail.com, yuchoi76@knu.ac.kr

Verification of PARR Band application on IoT OS model

Song Yang ○, Youngsul Shin*, Dongwoo Kim, Dongyeong Kim, Yunja Choi

School of Computer Science and Engineering, Kyungpook National University,
Center of Self-Organizing Software, Kyungpook National University*

요 약

심박 측정과 같은 의료 기능을 포함한 소프트웨어에 대해서는 극히 낮은 확률로 발생하는 미세한 오류라도 환자의 생명안전을 위협할 수가 있기 때문에 철저한 검증을 요구한다. 따라서, 본 연구에서는 의료 기능들을 포함한 파밴드(PARR) 어플리케이션의 안전성을 확보하기 위하여 파밴드 어플리케이션에서 준수되어야하는 속성 5개를 식별하고 이들의 준수여부를 검증하고자 한다. 파밴드 어플리케이션을 검증하기 위해 운영체제 C언어 모델링 기법을 확장하여 파밴드의 운영체제를 모델링하였다. 또한, 복잡한 요소로 구성된 어플리케이션을 검증하는 경우 상태 공간 폭발로 인해 검증 결과를 얻을 수 없는 문제를 해결하기 위해서 운영체제 모델을 어플리케이션과 결합하여 속성 기반 코드 자르기 기법을 적용하고 잘라진 통합 모델에 대해 주요 속성들이 만족함을 성공적으로 검증하였다.

1. 서 론

심박 측정과 같은 의료 기능을 포함한 소프트웨어에 대해서는 극히 낮은 확률로 발생하는 미세한 오류로 인해 전체 시스템이 응답하지 않거나 Deadlock 과 같은 문제들이 발생할 수 있으며 이 경우 심장 질병 환자의 생명 안전을 위협할 수가 있으므로 철저한 검증이 필요하다. 내장형 소프트웨어는 운영체제와 어플리케이션이 함께 컴파일되어 하나의 소프트웨어를 이루므로 둘의 행위를 모두 고려해야만 정확한 검증이 가능하다. 하지만, 현존하는 검증 기법들은 운영체제의 행위를 추상화하여 검증함으로써 실제로 발생하지 않는 행위를 보고(오탐)하거나 내장형 운영체제를 모델링 언어를 이용하여 정의함으로써 C언어로 정의되는 어플리케이션과 언어적 불일치성이 유발된다는 문제점이 있다.

이 문제들을 해결하기 위해 이전 연구[1]에서는 차량 전장용 내장형 운영체제 표준 OSEK/VDX[2] 에 따라 C 언어로 운영체제를 모델링하고 내장형 어플리케이션을 검증하는 방법을 제안하였다. 하지만, 내장형 운영체제는 OSEK/VDX 외에도 다수 존재하며, 다른 운영체제에서 작동하는 어플리케이션을 검증하기 위해서는 새 운영체제 모델이 필요하다는 문제가 있다. 또한, 이전 연구에서 소개한 어플리케이션 검증 기법은 규모가 작은 어플리케이션에만 적용 가능하며 상용 어플리케이션에 적용 시 상태 폭발로 인해 정확한 결과를 얻을 수 없다는 문제가 있다.

그런데 내장형 운영체제 간에는 어플리케이션 행위를 구현하는 태스크, 임계 구역 명시를 위한 리소스, 태스크간 실행 순서를 명시하는 이벤트 등 공통 구성요소가 많아 운영체제마다 각각 모델링하는 대신 기존 모델을 확장하여 다른 내장형 운영체제를 정의할 수 있으며, 복잡한 소프트웨어 검증 시 발생하는 상태 폭발 문제를 피하기 위해 파밴드 어플리케이션과 운영체제 모델이 결합된 코드에 속성 기반 코드 자르기를 적용함을 통해 검증 속성과 관련된 모델 요소 및 행위만을 유지함으로써 검증 복잡도를 낮출 수 있다.

본 연구에서는 이전 연구에서 C언어로 정의한 OSEK/VDX 운영체제 모델을 확장하여 IoT 기기에 사용되는 Ubinos[3] 운영체제를 정의한 방법을 소개하며 Ubinos 운영체제 위에서 작동하는 파밴드(PARR Band) 어플리케이션과 Ubinos 운영체제 모델을 결합하여 코드 자르기 기법을 적용하고 기능적 요구사항을 검증한 방법을 소개한다.

그림 1의 왼쪽은 Ubinos 운영체제 모델 생성과 검증 과정을 소개하고 오른쪽은 파밴드 어플리케이션의 검증 과정을 소개한다. Ubinos 운영체제는 OSEK/VDX 운영체제에 포함되지 않은 라운드 로빈 스케줄링, 뮤텍스, 세마포어와 메시지 큐를 포함한다. 본 연구에서는 각 구성요소의 행위를 Ubinos의 사양에 따라 상태 기계로 정의하고 C언어로 나타내었다.

그리고 Ubinos에서 도출한 속성들을 이용하여 Ubinos 운영체제 모델의 타당성을 검증하였다. 다음으로, 운영체제 모델과 함께 동작하기 위한 문맥 전환점을 어플리케이션에 삽입[1]하고 구성된 운영체제 모델과 통합하여 속성 기반 코드 자르기 기법을 적용한다. 잘라진 통합 모델은 CBMC 검증 도구[4]를 통해 검증되었다.

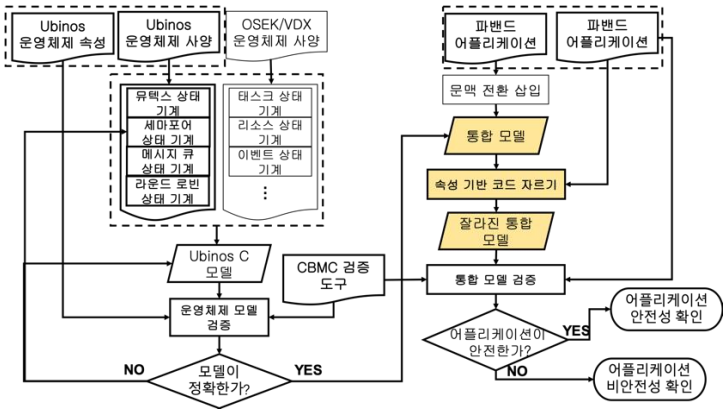


그림 1. 파밴드 어플리케이션 검증 기법

2. 연구 배경

2.1. OSEK/VDX 및 Ubinos 운영체제 비교

OSEK/VDX[2]는 자동차 산업의 컨소시엄에서 발표된 표준이다. OSEK/VDX는 모든 타입의 제어장치에서 사용할 수 있도록 표준화된 인터페이스, 범위성, 오류검사 및 응용 프로그램 소프트웨어의 이식성을 중심으로 설계되었다. OSEK/VDX 운영체제의 주요 특징은 (1) 단일 프로세서에서 실행되는 것으로 가정하고 (2) 필요한 모든 시스템 변수를 정적으로 할당해야 한다는 점이다.

표 1에서는 OSEK/VDX와 Ubinos 운영체제의 차이점을 보여주고 있다. Ubinos 운영체제는 OSEK/VDX 운영체제에서 지원하지 않는 라운드로빈 스케줄링, 세마포어, 뮤텍스 및 메시지 큐 기능들을 가지고 있다.

2.2. 파밴드 어플리케이션

파밴드 어플리케이션은 손목에 착용하는 디바이스에서 작동하는 Ubinos 운영체제[3]에서 작동하며, 심박, 혈당 및 걸음걸이와 같은 의료 기능들이 포함되어 있다. 또한, 자동차와 연결해서 차문과 트렁크 문을 여는 기능을 수행할 수 있다.

그림 2는 파밴드 어플리케이션의 태스크 구조를 보여주고 있다. 파밴드 어플리케이션은 총 10480 라인의 코드가 있으며 5개의 태스크, 3개의 뮤텍스, 3개의 세마포어 및 5개의 메시지 큐를 통해 자원 접근 및 태스크 간의 통신이 구현된다. 또한, 5개의 태스크는 각각 데이터 수집, 수집된 데이터 전달, 태스크 간의 이벤트 관리, 심박 및 혈당 측정 및 심박과 같은 데이터를 디스플레이로 출력하는 기능을 담당하고 있다.

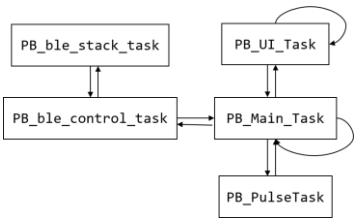


그림 2. 파밴드 어플리케이션 태스크 구조

2.3. 속성 기반 코드 자르기 기법

속성 기반 코드 자르기 기법[5]은 검증 속성 기반으로 제어의존성 및 자료의존성을 분석해서 속성과 의존성이 있는 구문들을 추출하는 기법이다. 속성 기반 코드 자르기 기법을 사용하여 관심 대상의 행위와 코드만을 남기고 불필요한 코드를 제거함으로써 대상의 본연의 행위를 유지하면서 대상의 복잡도 및 크기를 줄일 수 있다.

```
1 void test(){
2     int i,sum,product,n;
3     read(n);
4     i = 1;
5     sum = 0;
6     product = 1;
7     while( i <= n){
8         sum = sum + i;
9         product = product * i;
10        i = i + 1;
11    }
12    print(sum);
13    assert(product == some_value);
```

그림 3. 속성 기반 코드 자르기 예제

예를 들어, 그림 3의 예제 프로그램에서 product가 프로그램 종료시점에서 특정 값을 가짐을 검증하려고 할 때, 5, 8 및 12번 줄은 변수 product의 값에 어떤 영향을 주지 않으므로 제거하여도 검증 속성 product == some_value를 검증하는데 영향을 주지 않는다.

3. 검증 기법

이 장에서는 운영체제를 모델링하는 과정과 검증 속성을 소개한다. 또한, 파밴드 어플리케이션의 검증 속성

표 1. OSEK/VDX와 Ubinos 운영체제의 특징 비교

OS 종류	스케줄링	우선순위 순서	메시지 전송 방식	ISR	Deadlock 발생 여부	뮤텍스	세마포어	메시지 큐
OSEK/VDX	Priority	오름차순	비동기화	●	X	X	X	X
Ubinos	Priority, RR	오름차순	비동기화	●	X	●	●	●

표 2. 세마포어 관련 검증 속성 리스트

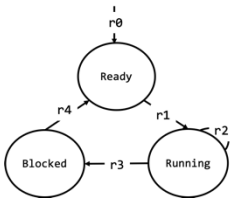
No.	검증 속성
M1	세마포어를 주는 API 가 호출될 때 대기열에서 세마포어 획득 API 때문에 수면 중인 태스크가 있는 경우 해당 태스크가 Ready 상태로 변경되어야 한다.
M2	세마포어와 관련된 대기 열에 수면 중인 태스크가 두 개 이상인 경우 대기 열에 있는 태스크들을 우선순위에 기반해 정렬해야 한다.
M3	세마포어 획득 API 가 호출됐을 때, 세마포어의 변수가 0 과 똑같거나 작은 경우 API 를 호출한 태스크가 대기열로 들어가고 수면상태가 된다.

을 소개한다. 마지막으로 검증 비용을 최대한 낮추기 위한 코드 자르기 기법 적용 방법을 소개한다.

3.1. 운영체제 모델

Ubinos 운영체제를 모델링하기 위해 라운드 로빈 스케줄링, 세마포어, 뮤텍스 및 메시지 큐를 모델링하였으며 이장에서는 세마포어 모델링에 관해 소개한다.

세마포어는 멀티태스킹 환경에서 공유자원에 접근하는 방법이며 Ubinos는 자원 할당 및 해제를 위한 sem_give API 함수와 sem_take API 함수를 제공한다.



상태 번호	상태 전이 조건
r0	API = task_create(tid)
r1	API = ubik_comp_start() | scheduler.time = 1ms
r2	API = (sem_give(sem_pt) & Sem_WQ = null) | (sem_take(sem_pt) & sem_pt.counter > 0)
r3	API = sem_take(sem_pt) & sem_pt.counter = 0
r4	API = sem_give(sem_pt) & sem_pt.counter >0

그림 4. 세마포어 상태 기계

그림 4는 세마포어 상태 기계 및 대응하는 상태 전이 조건을 보여주고 있다. 예를 들면 r3 상태전이는 어떤 태스크가 sem_take API 함수를 호출할 때 해당 세마포어의 카운터 값이 0인 경우 API를 호출한 태스크를 수면상태(Blocked)로 만들어 공유 자원이 생기기를 기다리도록 한다.

같은 방식으로 뮤텍스, 메시지 큐, 라운드로빈 스케줄링을 모델링할 수 있으며 추가된 모델들과 OSEK/VDX 모델을 합쳐 Ubinos 운영체제 모델을 구성한다. C언어로 구현된 Ubinos 모델은 2873 라인의 코드로 구성된다.

3.2. 검증 속성 정의

Ubinos C언어 운영체제 모델의 검증 속성은 표 2에서 보여준다. 이 속성들은 Ubinos 운영체제의 사양에서 도출하였다. 검증 대상은 메시지 큐, 라운드 로빈 스케줄링, 세마포어 및 뮤텍스 4가지 사항으로 구분되며 표 2는 세마포어의 검증 속성을 예로 보여준다.

표 3은 파밴드 어플리케이션의 행위에 따라 도출한 일부 속성이다. 이 속성들은 C언어의 assert 문으로 변환되어 코드에 삽입된 후, 모델 검증기 CBMC를 이용하여 검증된다. 예를 들면 A1 속성에 대응하는 C언어 표현식은 “assert(!(msgq_list[BLE_Msgq].R > 0));”로 표현되며 BLE_Msgq의 Rear 값이 증가되면 해당 메시지가 메시지 큐에 저장된다는 것을 의미한다.

표 3. 파밴드 어플리케이션 속성

No.	검증 속성
A1	첫 번째 메시지가 send 함수를 통해 메시지 큐로 보내지면 메시지 큐의 Rear 값이 0 보다 커야 한다.
A2	send 함수를 통해 메시지 큐로 보낸 메시지는 메시지 큐에 저장되어 있지 않으면 안된다.
A3	receive 함수가 전혀 호출되지 않는 경우 메시지 큐가 overflow 되는 경우가 없다.
A4	PB_ble_stack 태스크는 메시지 큐를 통해 이벤드 PB_BLE_EVT 를 전달받을 수 있다.
A5	PB_ble_stack 태스크는 메시지 큐를 통해 3 가지상태 PB_BLE_CONNECTION_EVT_ST, PB_BLE_PACKET_REV_EVT_ST 및 PB_BLE_DISCONNECTION_EVT_ST 이외에 있는 메시지를 받을 수 있다.

3.3. 파밴드 검증 효율을 향상 시키기 위한 속성 기반 코드 자르기

2873 라인을 가지는 운영체제 모델과 10480 라인을 가지는 파밴드 프로그램을 결합해서 검증하는 경우, 높은 복잡도로 인해 검증 결과를 얻을 수 없다.

이 문제를 해결하기 위해서 코드 자르기 도구 Frama-C[6]를 사용하였으며 통합 모델에서 검증 속성과 의존성이 있는 코드만 남기고 불필요한 코드를 삭제하여 검증 비용을 낮추었다.

4. 실험

모든 실험은 Linux version 4.4.0-154-generic, x86_64 구조, 20 Intel(R) Xeon(R) W-2155 CPU@ 3.30GHz CPU, 32GB 및 CBMC 5.11 환경에서 수행되었다. 또한, 모든 실험은 검증 과정에서 unwind 값 1부터 적용하였으며 여러 번의 검증을 통해 사양에서 추출한 속성의 만족 여부가 확인될 때 까지 unwind 값을 1씩 증가시켜가며 재검증하였다.

표 4는 표 2에서 보여준 세마포어 속성들을 검증 도구 CBMC로 검증한 결과이다. 모든 속성에 대해서, unwind 값이 7인 경우에 기대한 결과가 나온다. 모든 검증 결과는 1분 이내에 보고되었으며 메모리는 최대 277MB 사용하였다.

표 4. 세마포어 C언어 모델 검증 결과

No.	Unwind	시간 (sec)	메모리 (MB)	기대결과	결과
M1	7	11.4	87	TRUE	TRUE
M2	7	37.2	277	TRUE	TRUE
M3	7	16.2	147	TRUE	TRUE

속성 기반 코드 자르기 기법을 적용한 소스코드에서 도출한 어플리케이션 속성을 CBMC 검증 도구를 통해 검증하였다.

그림 5는 검증 속성 A1에 대하여 unwind 값을 8까지 적용하였을 때 메모리가 부족하여 정확한 검증 결과를 얻을 수 없었음을 보인다. 이에 반해, 통합 모델을 속성 기반 코드 자르기 후에 기존 약 만 삼천 라인의 코드를 오천 라인으로 줄였기 때문에 unwind 값이 10일 때도 정확한 검증 결과를 얻을 수 있었다.

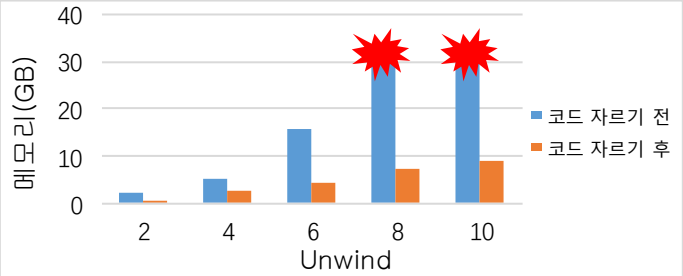


그림 5. 코드 자르기 전후 검증 메모리 비교

표 5에서는 통합 모델에 코드 자르기를 적용한 후의 검증 결과를 보여 준다. 모든 검증 행위는 수작업으로 확인한 결과와 동일함을 확인하였다.

표 5. 파밴드 어플리케이션 검증 결과

No.	Unwind	시간(min)	메모리(GB)	기대결과	결과
A1	6	3.46	4.45	TRUE	TRUE
A2	8	2.22	4.12	FALSE	FALSE
A3	8	11.22	11.03	FALSE	FALSE
A4	8	10.55	9.72	TRUE	TRUE
A5	10	10.14	10 .14	FALSE	FALSE

5. 결 론

본 연구에서는 의료 기능을 포함한 파밴드 어플리케이션을 검증하기 위해서 이전 연구에서 제안된 운영체제를 C언어로 모델링하는 기법을 확장하여 Ubinos 운영체제를 모델링하였다. 다음으로, 높은 검증 비용으로 검증 결과를 얻을 수 없는 문제점을 피하기 위해서 통합 모델 코드에 속성 기반 코드 자르기 기법을 적용하여 성공적으로 검증하였다.

현 검증 기법의 한계점은 파밴드 어플리케이션에서 사용하는 시간 관련 API 들을 상대시간으로 모델링하였다는 점이며 이로 인해 실제로 발생가능하지 않은 행위가 검증 결과로 보고될 수 있다. 또 하나의 문제점은 CBMC는 unwind 값에 따라 기하급수적으로 높은 검증 비용을 요구한다는 점이며 검증 속성에 따라서 높은 unwind 값이 꼭 필요한 경우가 있다. 향후 연구에서는 적은 unwind 값을 이용해서도 검증할 수 있도록 운영체제 모델의 행위나 환경을 추상화하여 검증할 예정이다.

감사의 글

이 논문은 2018, 2019년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2018R1A6A1A03025109, NRF-2016R1D1A3B01011685)

참고 문헌

[1] Y. Chung, D. Kim, and Y. Choi, “Modeling OSEK/VDX OS requirements in C,” in 2017 24th *Asia-Pacific Software Engineering Conference*, pp. 398-407, 2017.

[2] OSEK Group. "OSEK/VDX Operating System Specification." site web: http://www. osek-vdx. org (2005).

[3] "Introduction", *ubinos*, 2020. [Online]. Available: http://ubinos.org/.

[4] D. Kroening, and M. Tautsching, "CBMC–C bounded model checker." *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 389-391, 2014.

[5] M. Park, D. Kim, and Y. Choi, “Codeant: Code slicing tool for effective software verification,” *KIPS transactions on software and data engineering*, vol. 4, pp. 1-8, 2015.

[6] C. Pascal, K. Florent, K. Nikolai, P. Virgile, S. Julien, and Y. Boris, "Frama-c." *International Conference on Software Engineering and Formal Methods*, pp. 233-247, 2012.

PPMI와 NPMI를 이용한 자동화 된 감성 사전 구축 방법

류기곤 ○
고려대학교 정보창의교육연구소
gon0121@korea.ac.kr

A Method for Constructing Sentiment Lexicon Automatically using PPMI and NPMI

Kigon Lyu○
Creative Informatics and Computing Institute, Korea University

요 약

감성 분석(sentiment analysis)은 특정 대상에 대한 사람들의 태도, 느낌 등 주관적인 의견을 분석하는 방법이다. 감성 어휘로 구성 된 감성 사전은 감성을 모델링 하기 위한 중요한 자질이 되고 극성 판단의 기준이 되기 때문에 매우 중요하다. 말뭉치 기반의 감성 사전 구축 방법은 감성 분류가 이미 잘 알려진 소수의 감성 어휘를 선정하여 말뭉치 내 어휘들과 극성 관계를 분석하여 구축한다. 본 논문에서는 말뭉치 기반의 감성 사전 구축 방법을 제안한다. PPMI와 NPMI를 이용하여 각 어휘의 감성 극성을 확률로 표현하고, PMI를 이용하여 감성 어휘를 자동으로 선정하여 감성 사전 자동으로 구축한다. 실험 결과 말뭉치를 이용하여 사람의 개입없이 자동으로 감성 사전을 구축하여 영화평에 대한 감성 분석을 수행하였다. F1-Score가 0.73 이상을 보이며 충분히 데이터 기반의 분석 방법도 가능함을 알 수 있었다.

1. 서 론

감성 분석(sentiment analysis)은 특정 대상에 대한 사람들의 태도, 느낌 등 주관적인 의견을 분석하는 방법이다[1]. 일반적으로 감성을 긍정/부정 양 극단으로 정의하고 기계 학습을 이용하여 주관적인 의견들을 하나의 감성으로 분류하거나 감성 어휘를 이용하여 극성(polarity)에 대한 감성 강도(strength)를 정량화하여 분석한다[2].

감성 어휘로 구성 된 감성 사전은 감성을 모델링 하기 위한 중요한 자질(feature)이 되고 극성 판단의 기준이 되기 때문에 매우 중요하다. 감성 사전을 구축하기 위한 방법은 크게 세 가지로 구분되며 전문가 기반, 사전 기반 및 말뭉치 기반이다. 전문가와 사전 기반은 많은 시간과 노력이 필요하고 사전에 등재된 단어로만 구성되기 때문에 축약어, 신조어, 관용어 등 문화적, 시대적 특징을 반영하기 어려운 한계가 있다. 따라서 소셜 미디어와 같은 빅데이터를 활용하여 말뭉치 기반으로 감성 사전을 구축하기 위한 연구가 활발히 진행되고 있다.[3-4]

말뭉치 기반의 감성 사전 구축 방법은 감성 분류가 이미 잘 알려진 소수의 감성 어휘를 선정하여 말뭉치 내 어휘들과 극성 관계를 분석하여 구축한다. 극성 판단은 PMI(pointwise mutual information)를 이용하는 것이 대표적이며 SO(semantic orientation)를 통해 감성을 판단할 수 있다[5-6]. 말뭉치 기반은 빈도를 통해 확률을 추정하기 때문에 말뭉치에 출현하지 않은 단어들에 대해서는 PMI가 음의 무한대가 되는 문제가 있다. PPMI(positive PMI)는 0 또는 양의 상관관계만을 판단하기 때문에 빈도로 인해 상관관계가 음수가 나오는 PMI의 문제를 해결할 수 있다. NPMI(normalized PMI)는 PMI를 -1부터 1사이의 값으로 정규화 하여 모든 어휘의 상호 정보량 PMI를 동일한 범위로 해석할 수 있도록 하며, PPMI를 정규화 하면 0부터 1사이의 값으로 정규화 되기 때문에 확률로 해석할 수 있다[7].

본 논문에서는 말뭉치 기반의 감성 사전 구축 방법을 제안한다. NPMI를 이용하여 말뭉치로부터 감성 어휘를 자동으로 선정하고, PPMI와 NPMI를 이용하여 선정 된 감성 어휘와 다른 어휘 들과의 감성 극성을 확률로 표현한다. 특히 PPMI와 NPMI를 결합하여 양의 상관관계만을 판단하는 PNPMI(positive NPMI)를 제안하여 말뭉치 기반의 감성 사전을 자동으로 구축한다.

2. 본 론

2.1 말뭉치 수집

특정 대상에 대한 사람들의 다양한 주관적인 의견을 수집하기 위해 영화평을 수집하였다. 2019년 한 해 동안 개봉한 885개의 영화로부터 795,613개의 사용자 평을 수집하였고, 영화평은 1부터 10까지 10단계로 점수가 부여되어 있다. 점수가 낮을수록 영화에 대한 평가가 낮기 때문에 부정으로 판단하였고, 높을수록 긍정으로 판단하였다.

2.2 극성 분석

두 확률변수 x, y의 상호정보량 PMI는 다음처럼 결합

확률과 개별 확률을 이용하여 계산할 수 있다.

$$PMI(x,y) = \log\left[\frac{p(x,y)}{p(x)p(y)}\right]$$

여기서는 빈도를 통해 확률을 추정해야 하기 때문에 다음처럼 WebPMI를 이용하여 계산한다.

$$WebPMI(x,y) = \log\left[\frac{\frac{n(x\wedge y)}{N}}{\frac{n(x)}{N}\frac{n(y)}{N}}\right] = \log\left[\frac{n(x\wedge y)}{n(x)n(y)}\right]$$

$n(x)$는 어휘 x가 출현한 문서의 수이고, N은 수집 된 전체 문서의 수이다. 빈도가 0이 되는 문제를 해결하기 위해 라플라스 스무딩(laplace smoothing)을 적용한다.

$$-\infty \le PMI(x,y) \le \min[-\log p(x), -\log p(y)]$$

PMI는 위와 같은 값의 범위를 갖기 때문에 빈도에 따라 값의 편차가 매우 크다. 따라서 빈도에 둔감 하게 일정한 범위로 정규화 할 필요가 있고 여기서는 NPMI를 이용하여 -1부터 1사이로 정규화 한다.

$$NPMI(x,y) = \frac{PMI(x,y)}{-\log p(x,y)}$$

$$-1 \le NPMI(x,y) \le 1$$

PMI의 목적은 말뭉치 내 출현한 모든 어휘에 대해 어떤 감성 어휘와 상관성이 높은지를 판단하기 위한 것이므로 PPMI를 이용하여 양의 상관관계만을 고려한다. PPMI는 0 또는 양의 상관관계를 갖는 PMI 중 큰 값을 취하기 때문에 음의 상관관계는 0이 된다. PMI에서 0은 두 확률변수가 서로 독립이기 때문에 마찬가지로 감성 어휘와 말뭉치 내 어휘의 관계를 독립이라고 판단할 수 있다.

$$PPMI(x,y) = \max(0, PMI(x,y))$$

단, 여기서 PMI 대신 NPMI를 이용하여 0부터 1사이의 값을 갖도록 식을 변형한다.

$$PNPMI(x,y) = \max(0, NPMI(x,y))$$

2.3 감성 판단

양 극성으로 분류되는 감성 어휘와 다른 어휘 들간의 상호정보량 PMI 합 차이를 이용하여 감성 극성을 판단하는 의미지향성 SO는 다음처럼 계산할 수 있다.

$$SO = \sum_{y\in Positive} PMI(x,y) - \sum_{y\in Negative} PMI(x,y)$$

긍정에 속하는 감성 어휘와의 PMI 합과 부정에 속하는 감성 어휘와의 PMI 합 간의 차이를 이용하여 양수이면 긍정, 음수이면 부정으로 판단한다. 여기서는 PMI 뿐 아니라 NPMI, PPMI, PNPMI를 통해 감성 어휘와의 상관관계를 계산하였기 때문에 SO를 다양하게 변형하여 각각 사용하였다.

3. 실험 및 평가

3.1 영화평 수집 결과

2019년에 한 해 동안 국내에서 개봉한 885개의 영화로부터 795,613개의 영화평을 수집하였다. 이 중 양 극단에 있는 1점과 10점에는 총 534,740개가 있었고 대부분 영화에 대한 논란으로 인해 평점이 극과 극인 경우였다. 따라서 양 극단에 가까우면서 데이터의 수가 균형 있게 분포되어 잇는 2점과 9점인 평점들을 분석 대상으로 선정하였다. 2점은 37,840개이고, 9점은 42,594개이다.

3.2 감성 어휘 선정

자동화 된 감성 사전 구축을 위해 사람의 개입이나 사전 지식 없이 데이터로부터 감성 어휘를 추출하였고, 2점을 부정 집합, 9점을 긍정 집합으로 정의하였다. 구어체 중심의 말뭉치를 분석하기 위해 어휘 기반 감성 분석에서 널리 사용되는 형용사, 부사, 동사 뿐 아니라 형식형태소를 제외한 명사, 형용사, 동사 및 어근을 분석하였다.

NPMI를 이용하여 빈도에 의한 극성 차이가 특정 범위 내에서 표현되도록 긍정-부정 집합과의 극성을 분석하였고, 두 극성 간의 차이를 통해 의미지향성을 계산하여 감성을 판단하였다. 각 감성 분류에서 상위 5개의 어휘를 선정하여 총 10개의 감성 어휘를 기준으로 결정하였다.

<표 1> 감성 어휘

긍정 어휘	SO	부정 어휘	SO
기대되	0.4437	최악	-0.6103
재밌	0.4220	재미없	-0.5066
재미있	0.4075	아깝	-0.5029
유쾌	0.3946	알바	-0.4786
깔끔	0.3903	낭비	-0.4435

부호는 긍정과 부정 감성을 나타내고 감성 강도는 0부터 1사이의 값으로 정규화 됐기 때문에 확률로

해석할 수 있고, 다른 감성 어휘와의 상대적인 차이를 비교할 수 있다. 따라서 같은 감성 어휘일지라도 어떤 어휘가 더 많은 긍정 또는 부정 감성을 나타내는지 판단할 수 있다.

3.3 감성 사전 구축

앞에서 결정 된 감성 어휘를 이용하여 모든 어휘에 대해 극성 분석을 다시 수행하였다. 단 10개의 감성 어휘와의 극성을 각각 계산하였고, 평가를 위해 PMI, NPMI, PPMI를 구분하였다. 각 감성 어휘와의 극성을 긍정과 부정 두 감성 분류로 구분하여 합하였고 긍정-부정 차이를 통해 의미지향성을 계산하였다.

의미지향성에 의해 판단 된 말뭉치 출현 어휘들의 감성 분석 결과 상위 5개는 다음과 같다. 단 기준으로 선정 된 감성 어휘는 목록에서 제외하였다.

<표 2> PMI 감성 어휘

긍정 어휘	SO	부정 어휘	SO
신선	2.0938	쓰레기	-1.9506
가볍	1.9982	평점	-1.5987
매력적	1.9584	낚이	-1.1339
즐겁	1.9438	나가	-1.1269
강추	1.8578	마라	-1.1236

<표 3> NPMI 감성 어휘

긍정 어휘	SO	부정 어휘	SO
가볍	0.2275	쓰레기	-0.1842
신선	0.2272	평점	-0.1614
즐겁	0.2047	시간	-0.1403
다음	0.2040	주고	-0.1370
매력적	0.1968	낚이	-0.1339

<표 4> PPMI 감성 어휘

긍정 어휘	SO	부정 어휘	SO
연휴	1.5207	낚이	-1.2044
군더더기	1.3868	시간	-1.1630
가벼운	1.3620	평점	-1.0134
기대치	1.3046	역대	-0.9795
쏠쏠	1.2884	주고	-0.9389

결과에서 볼 수 있듯이 PMI, NPMI, PPMI에 따라 단어의 순서와 감성 정도는 조금씩 차이가 나지만, 전반적으로 긍정 어휘와 부정 어휘 간의 극성 차이가 있음을 확인할 수 있고 신조어, 축약어와 같은 구어체적 특징도 분석 가능함을 알 수 있다.

또한 NPMI를 제외한 다른 방법들에서는 어휘들의 상대적인 극성 차이를 비교하기 어렵지만, NPMI에서는 정규화 되었기 때문에 어휘들 간의 상대적인 감성 강도의 차이를 비교하는 것이 가능함을 알 수 있다.

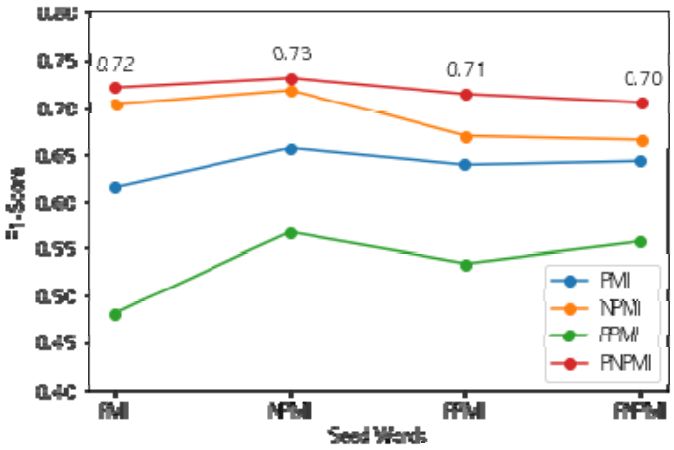
3.4 평가

말뭉치로부터 자동으로 선정 된 감성 어휘를 이용하여 감성 사전을 구축하였을 때 감성 분석 결과의 성능 차이를 분석하기 위해 PMI, NPMI, PPMI, PNPMI를 구분하여 10회 교차 검증(10-folds cross validation) 하였다. 교차 검증에 사용 된 데이터는 감성 사전 구축과 마찬가지로 평점이 2점과 9점인 영화평들만 대상으로 하였고 무작위로 정렬하였다.

본 논문에서 제안하는 자동 사전 구축 방법의 성능을 비교 평가하기 위해 PMI, NPMI, PPMI 및 PNPMI를 이용하여 감성 어휘와 감성 강도를 각각 선정하여 비교하였다. 실험 결과 NPMI를 이용하여 감성 어휘를 선정하는 방식이 가장 좋은 성능을 보였고, 소수의 감성 어휘를 이용한 사전 기반 감성 분석에서는 PNPMI가 가장 좋은 성능을 보였다.

다음 그림에서 X축은 감성 어휘 선정 방식이고, Y축은 F1-Socre를 나타낸다. F1-Score는 순서대로 0.72, 0.73, 0.71, 0.70 이었고, 감성은 주관적이기 때문에 70% 이상이면 비교적 정확하다고 판단할 수 있다[8].

<그림 1> 성능평가 결과



어휘 기반 감성 분석에서 가장 중요한 기준 어휘를 선정하는 방식에서는 NPMI가 다른 방법들보다 성능이 가장 좋았다. 이는 PMI에 비해 감성 강도가 빈도에 영향을 덜 받기 때문에 상대적인 감성 강도의 편차가 줄어들었기 때문이다. 또한 기준이 되는 감성 어휘 기반 감성 분석에서는 PNPMI가 다른 방법들보다 전반적인 성능이 가장 좋았다. NPMI를 통해 감성 강도의 상대적인 편차가 줄어들었고 감성 어휘와의 양의 상관관계만을 고려하기 때문에 음의 상관관계와 같은 노이즈를 줄이는 효과를 가져왔다.

실험을 통해 본 논문에서 제안하는 것처럼 기존 어휘 기반 감성분석에서의 어려움인 감성 사전 구축 방법에서 사람의 개입이나 사전 지식 없이 수집 된 말뭉치만으로도 감성 사전을 구축하여 감성 분석이 가능함을

알 수 있었다.

4. 결 론

말뭉치 기반의 감성 사전 구축 방법은 감성 분류가 이미 잘 알려진 소수의 감성 어휘를 선정하여 말뭉치 내 어휘들과 극성 관계를 분석하여 구축하는 것이 일반적이다. 감성 어휘를 어떻게 선정하는지에 따라 전체 감성 분석에 영향을 주기 때문에 사람의 지식이 반드시 필요하였다.

본 연구에서는 말뭉치를 이용하여 사람의 개입없이 자동으로 감성 사전을 구축하여 영화평에 대한 감성 분석을 수행하였다. F1-Score가 0.73을 보이며 충분히 데이터 기반의 분석 방법도 가능함을 알 수 있었다. PMI는 빈도에 영향을 크게 받기 때문에 NPMI를 통해 정규화 하여 감성 사전을 구축하였고, 구축 된 사전을 통해 PNPMI를 이용한 어휘 기반 감성 분석에서의 성능을 향상시킬 수 있었다.

언어적인 지식을 통해 품사, 구 또는 다른 문법적 패턴에 대한 고려 없이 임의로 특정 품사들을 추출하였지만 말뭉치 기반의 자동화 된 감성 사전 구축 가능성을 확인하였다. 향후 자연어처리를 이용한 다양한 전처리 기법과 문법적 규칙을 적용하면 양질의 감성 사전을 구축할 수 있을 것으로 기대할 수 있다.

5. 참고문헌

[1] Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., Kappas, A., Sentiment strength detection in short informal text. Journal of the American Society for Information Science and Technology, 61(12), 2544-2558, 2010.

[2] Thelwall, M., Buckley, K., Paltoglou, G., Sentiment strength detection for the social web. Journal of the American Society for Information Science and Technology, 63(1), 163-173, 2012.

[3] Liu, B., Zhang, L., A survey of opinion mining and sentiment analysis. In Mining text data, Springer US, 415-463, 2012.

[4] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K. J., Introduction to WordNet: An on-line lexical database. International journal of lexicography, 3(4), 235-244, 1990.

[5] Turney, P. D. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics. 417-424, 2002.

[6] Turney, P., Littman, M. L., Unsupervised learning of semantic orientation from a hundred-billion-word corpus, 2002.

[7] Bouma, G. Normalized (pointwise) mutual information in collocation extraction. Proceedings of GSCL, 31-40, 2009.

[8] Kennedy, Helen. Perspectives on sentiment analysis. Journal of Broadcasting & Electronic Media 56(4), 435-450, 2012.

사사

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2017R1D1A1B03033137).

국내외 인공지능 지식 구조 및 변동 분석: 한국, 미국, 중국의 논문 계량분석을 중심으로

박종현, 김문구
한국전자통신연구원, 경제사회연구실

Comparison of Knowledge Structure Analysis of Artificial Intelligence: Focusing on Korea, USA and China

Jong-Hyun Park, Moon-Koo Kim
Electronics and Telecommunications Research Institute, Economics & Social Research Section

요 약

4차 산업혁명의 핵심 기술로 인공지능이 부각되고 있다. 그러나, 인공지능 분야에서 우리나라의 역량은 세계 10위권 밖으로 글로벌 경쟁력이 낮은 것이 현실이다. 이에 인공지능 주요 강국인 미국과 중국 대비 우리나라의 인공지능 지식 구조 및 변동에 대한 비교 연구가 필요하다. 본 연구에서는 2010년 이후 한국, 미국, 중국을 대상으로 인공지능, 딥러닝의 핵심기술, 인공지능을 활용하는 자율주행차, 인공지능 헬스 분야에 대한 지식 구조 변동 비교 연구를 통해 우리나라의 글로벌 인공지능 경쟁력 강화를 위한 시사점을 파악하고자 한다.

1. 서 론

4차 산업혁명의 핵심 기술로 인공지능이 부각되고 있다. 인공지능은 다양한 산업분야에서 활용이 점차 확대되고 있으며 미국을 비롯한 주요국에서 인공지능을 차세대 핵심 성장 및 유망 기술로 선정하여 R&D 투자 확대 및 적극적인 정책적 지원을 집중하고 있다[1-5]. 이에 우리나라도 인공지능 역량 강화를 위한 다양한 정책 및 R&D 전략을 추진하고 있다[1-2]. 그러나, 인공지능 분야에서 우리나라의 역량은 세계 10위권 밖으로 글로벌 경쟁력이 낮은 것이 현실이다[1, 6]. 이에 인공지능 주요 강국인 미국과 중국 대비 우리나라의 인공지능 지식 구조 및 변동에 대한 비교 연구가 필요하다.

본 연구에서는 2010년 이후 한국, 미국, 중국을 대상으로 인공지능, 딥러닝의 핵심기술, 인공지능을 활용하는 자율주행차, 인공지능 헬스 분야에 대한 지식 구조 변동 비교 연구를 통해 우리나라의 글로벌 인공지능 경쟁력 강화를 위한 시사점을 파악하고자 한다.

2. 연구 방법

본 연구는 논문 계량분석을 통해 한국, 미국, 중국간 인공지능 지식구조 및 변동을 파악하기 위하여 다음과 같은 절차에 의거하여 연구를 수행하였다.

(그림 1) 분석 프로세스

1단계	Scopus 데이터베이스로부터 키워드 검색 실행
2단계	연도별, 국가별, 분야별 지식구조 파악
3단계	한국과 미국, 중국간 지식구조 변동 비교 분석
4단계	시사점 파악

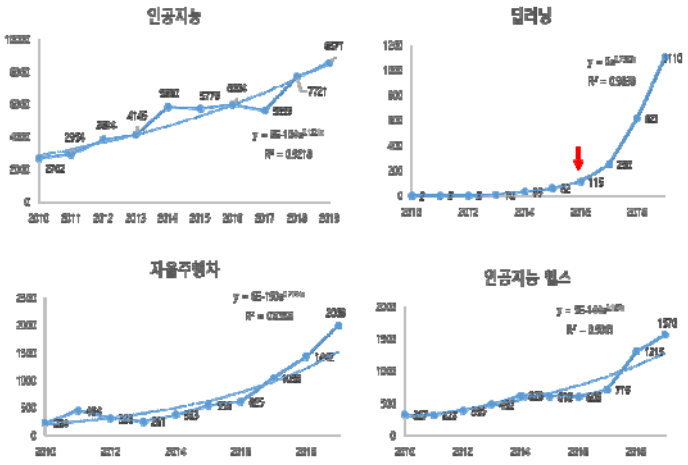
연구 분석을 위한 검색방법은 다음과 같다. 우선, 데이터를 확보하기 위해 학술 전문 데이터베이스인 'Scopus'에서 저널에 실린 논문과 리뷰를 대상으로 하였다. 다음으로 분석 기간은 2019년 12월 10일 검색 기준일을 기점으로 2010년부터 2019년까지 총 10년이다. 검색 분야는 인공지능, 딥러닝, 자율주행차, 인공지능 헬스 등 4가지를 대상으로 하였다. 검색결과 분야별 논문과 리뷰 편수는 인공지능 53,239개, 딥러닝 2,2096개, 자율주행차 7,354개, 인공지능 헬스 6,993개로로 나타났다.

3. 인공지능 논문 계량분석

3.1 인공지능 분야의 지식 생산

전반적으로 인공지능 분야에 대한 연구는 빠른 성장추세를 보이는 가운데 특히, 2016년 이후 딥러닝 분야의 급성장이 두드러진 것으로 나타났다.

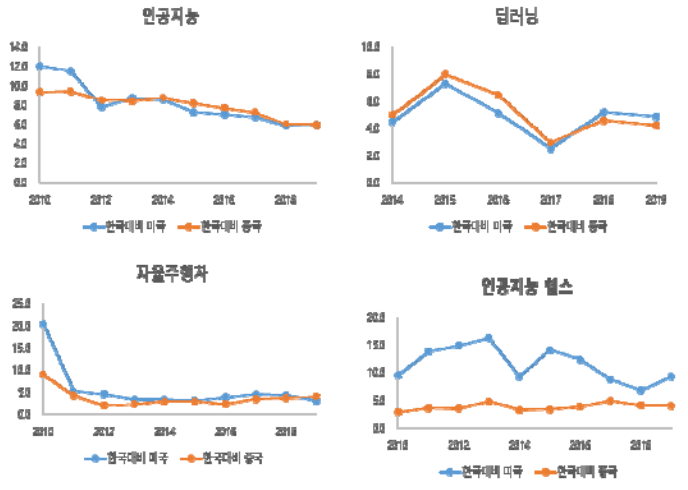
(그림 2) 인공지능 분야별 지식 생산 추이



3.2 한국 대비 미국과 중국의 지식 생산 비교

인공지능 분야의 글로벌 강대국인 미국과 중국 대비 한국의 지식 생산 격차가 2010년 이후 축소되는 경향을 보이고 있으나 최근에는 그 격차 해소가 무뎌지는 것으로 나타나고 있다. 인공지능, 딥러닝, 자율주행차 분야에서 미국과 중국의 양적인 격차가 거의 나타나지 않으나, 인공지능 헬스는 미국이 상대적 우위를 보이고 있다.

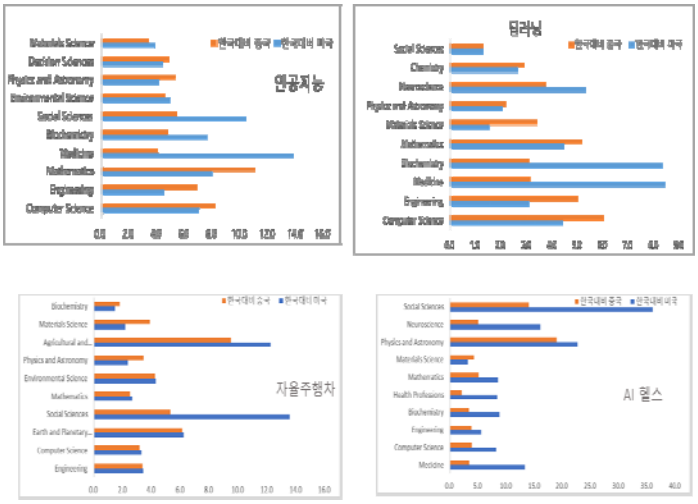
(그림 3) 한국 대비 미국과 중국의 지식 생산 추세 비교



3.3 한국 대비 미국과 중국의 강점 연구 분야

미국은 의학(인공지능), 생화학(딥러닝), 사회과학(자율주행차), 사회과학(인공지능 헬스) 등에서 강점을 보인 반면 중국은 수학(인공지능), 컴퓨터과학(딥러닝), 농업(자율주행차), 물리학(인공지능 헬스) 등의 분야에서 강점을 보유하고 있는 것으로 나타났다.

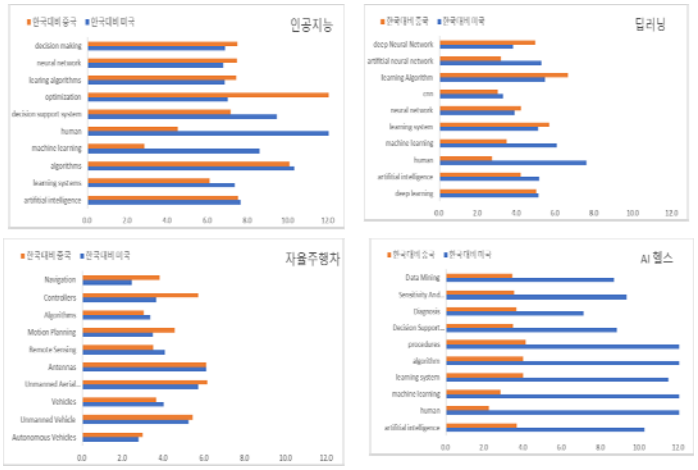
(그림 4) 한국 대비 미국, 중국의 강점 연구 분야 비교



3.4 한국 대비 미국과 중국의 강점 키워드

미국은 휴먼(인공지능, 딥러닝), 안테나(자율주행차), 알고리즘(인공지능 헬스) 등에서 강점을 보였으며, 중국은 최적화(인공지능), 학습알고리즘(딥러닝), 무인항공(자율주행차), 진단(인공지능 헬스) 등이 주요 강점 키워드로 나타났다.

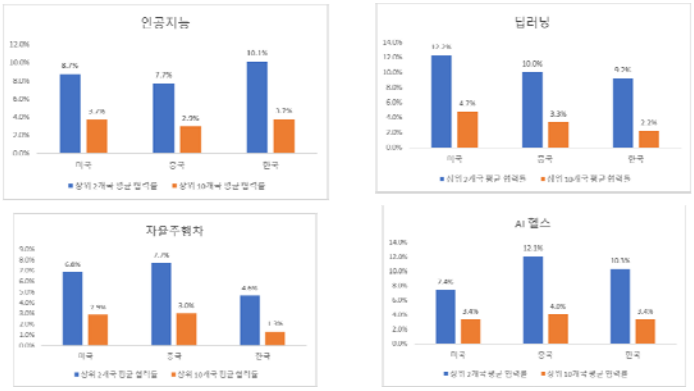
(그림 5) 한국 대비 미국, 중국의 강점 키워드 비교



3.5 국가간 연구협력

한국은 인공지능, 인공지능 헬스 분야에서 국가간 연구 협력률이 상대적으로 높게 나타난 반면, 자율주행차에서 국가간 연구협력률이 상대적으로 낮게 나타났다. 또한 우리나라는 미국과의 연구 협력이 편중되어 있는 것으로 나타났다.

(그림 6) 국가간 연구협력 비교



4. 결론 및 시사점

인공지능 분야에 대한 계량 논문 분석을 통한 지식 생산 구조 및 변동을 요약하면 다음과 같다. 우선, 인공지능 분야 논문 연구는 딥러닝을 중심으로 최근 급증하는 추세를 보이고 있어 딥러닝 분야에 대한 연구가 활발하게 이뤄지고 있음을 알 수 있다. 둘째, 우리나라는 미국과 중국 대비 양적인 측면에서 논문 연구 생산 격차가 여전히 존재하고 있다. 셋째, 미국과 중국은 인공지능 분야에서 논문 연구의 양적인 격차가 크게 나지 않으나, 인공지능 헬스 분야는 미국이 양적인 우위를 점하고 있다. 넷째, 미국은 중국 대비 상대적으로 자율주행차와 인공지능 헬스에서 사회과학 연구가 강점으로 나타났다. 마지막으로 우리나라는 자율주행차 분야에서 미국, 중국 대비 상대적으로 글로벌 연구 협력이 부족한 것으로 나타났다.

이에 우리나라 인공지능 분야 역량 강화를 위한 시사점으로는 우선, 우리나라의 인구수를 감안하면, 인공지능 전 분야에서 미국과 중국에 비해 양적인 연구 생산이 부족한 것은 아니다. 다만, 우리나라가 인공지능 연구 분야와 키워드 모두 특정한 강점을 지니는 분야가 없는 것으로 나타남에 따라 강점 분야 발굴을 위한 선택과 집중의 포지셔닝 전략이 요구된다. 다음으로 우리나라의 연구 협력도 미국이나 중국에 비해 그리 낮지 않은 편이나, 미국에 대한 편중은 다소 높은 편으로 다양한 국가와의 협력적 연구 기반 조성 마련이 필요하다. 특히, 국가간 연구협력률이 낮은 자율주행차에 대한 글로벌 연구협력 강화가 요구된다.

참고문헌

[1] 과학기술정보통신부, 인공지능(AI) R&D 전략, 2018.
[2] 관계부처합동, 인공지능 국가전략, 2019.
[3] IITP, 국내외 AI 활용 현황과 공공 적용, 2018.
[4] KOSTEC, 중국의 인공지능 발전현황 및 시사점, 2018.
[5] NIA, 인공지능을 선도하는 주요국의 핵심전략, 2018.
[6] Stanford University, The AI Index 2018 Annual Report, 2018.

트리거-액션 기반 서비스 매쉬업의 신뢰도를 개선하기 위한 전제 상태 및 목표 상태 검증 방법

김상훈 ○, 고인영
한국과학기술원
seiker@kaist.ac.kr, iko@kaist.ac.kr

Precondition and Goal State Assertion for Improving the Reliability of Trigger-Action Based Service Mashup

Sanghoon Kim○, In-Young Ko
Korea Advanced Institute of Science and Technology

요 약

트리거-액션 기반의 서비스 매쉬업 패러다임은 프로그래밍 경험이 없는 최종 사용자들도 쉽게 매쉬업을 개발할 수 있는 간결함으로 많은 플랫폼에서 사용되고 있다. 하지만 이 간결함은 반대로 사용자들에게 충분한 표현력(Expressive Power)을 제공하지 못하는 단점이 있는데, 이는 물리적인 환경에 직접적으로 작용하는 IoT 서비스의 불확실성과 결합하여 매쉬업의 신뢰도를 떨어뜨리는 한계로써 작용한다. 본 연구에서는 기존 트리거-액션 기반의 서비스 매쉬업의 간결함을 유지하면서 표현력을 개선하기 위한 수단으로 전제 상태 (Precondition) 및 목표 상태(Goal State)의 검증(Assertion)을 추가하는 것을 제안한다. 또한 실제 스마트 홈 IoT 서비스 환경에서 발생할 수 있는 예제 시나리오를 통해 제시된 방법이 기존 트리거-액션 기반의 서비스 매쉬업보다 효과적으로 문제를 예방할 수 있고, 문제가 발생했을 때 사용자들이 더 빠르게 문제를 인식할 수 있음을 확인하였다.

1. 서 론

사물 인터넷 (Internet of Things, IoT) 디바이스들의 보급이 증가함에 따라, IoT 환경이 점차 실현되고 있다. 따라서, 일상생활 속 스마트 홈이나 스마트 오피스와 같은 환경에서 IoT 기기를 활용한 IoT 서비스들이 점차 증가하고 있다. IoT 서비스란 IoT 기기를 활용한 서비스 기능을 네트워크를 통해서 제공하는 것을 의미하며, 일반 사용자들은 필요에 맞게 IoT 서비스를 제공받을 수 있다. 또한 개별 서비스를 사용하는 것에서 더 나아가, 일반 사용자들은 IoT 서비스를 조합해서 자신만의 어플리케이션을 만들고자 한다. 이것은 소프트웨어 공학 분야에서 전통적으로 추구해온 최종 사용자 소프트웨어 개발 (End-User Software Development) 연구와 관련된다.

그동안 개발된 다양한 최종 사용자 소프트웨어 개발 방법 중에서, 서비스 컴퓨팅 (Services Computing) 패러다임을 적용한 연구가 다양하게 논의되었다 [1]. 서비스 컴퓨팅 패러다임에서는 서비스 단위로 어플리케이션을 구성할 수 있고, 사용자들이 자신의 필요에 맞게 서비스를 조합할 수 있다. 따라서 사용자들은 쉽고 가볍게 어플리케이션을 만들어 사용할 수 있는데, 이러한 서비스 조합 방법을 서비스 매쉬업 (Services Mashup)이라고 부른다.

일상생활 속 IoT 서비스를 활용하려는 일반 사용자들의 관심 또한 증가하고 있다. 특히, 사용할 수 있는 IoT 서비스의 조합을 도와주는 매쉬업 플랫폼이 존재하는데, 그러한 플랫폼을 사용하는 일반 사용자의 수가 증가하였다. 예컨대, 서비스 매쉬업 플랫폼 IFTTT에서 단 한 번이라도 공개 애플릿(Applet, IFTTT에서 매쉬업을 부르는 명칭)을 생성한 적이 있는 사용자는 2013년 약 3.5만 명에서 2017년 약 13.5만 명까지 증가하였다 [2].

여기서 IFTTT는 트리거-액션 모델을 기반으로 하는 매쉬업 플랫폼인데, 트리거-액션 모델이란 최종 사용자들을 대상으로 한 개발 방법의 하나로, 트리거-액션 프로그래밍(Trigger-Action Programming)이라고도 불리며 주로 “IF [triggered], THEN [do action]”의 형태를 따른다. 트리거-액션 모델은 직관적이고 단순하여 일반 사용자들이 기존 프로그래밍 경험이 없이도 개발할 수 있다는 장점이 있다.

활용할 수 있는 IoT 서비스의 증가와 쉽고 간편한 매쉬업 플랫폼의 영향으로, IoT 서비스가 조합된 매쉬업 어플리케이션에 대한 사용자들의 의존성은 높아질 것이다. 그러나 물리적인 환경에서 직접적으로 작용해야 하는 IoT 디바이스의 특성상, IoT 서비스가 조합된 매쉬업 어플리케이션은 실제 물리 환경이 가지고 있는 예측 불가능성에 노출되어 있다. 여기서 예측 불가능성(Unpredictability)이란, 사용자가 매쉬업을 구성하는 시점과 매쉬업 어플리케이션이 실제로 작동되는 시점의 물리 환경이 얼마나 다를지 예측하기 어렵다는 것을 의미한다.

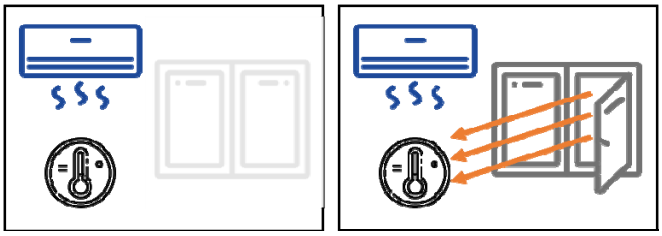


그림 1. 매쉬업을 구성하는 시점(왼쪽)과 매쉬업이 실행되는 시점(오른쪽)의 환경 요소의 차이

예컨대 사용자는 적정 실내 온도의 유지를 위해서 "방 안의 온도가 26도 이상일 경우, 에어컨을 작동한다"는 매쉬업 어플리케이션을 구성할 수 있다([그림 1] - 왼쪽). 하지만 매쉬업이 실제로 실행되는 시점에는 "창문이 열려있는 상태"라는 예측하지 못한 환경 요소의 영향으로 ([그림 1] - 오른쪽) 사용자가 기대했던 결과가 발생하지 않을 수 있다.

이렇게 환경 영향으로 인해 기대하지 않은 결과가 발생했을 때, 사용자들은 이를 방지하기 위해서 우선 매쉬업을 수정한다. 매쉬업을 수정하는 과정에서 오작동에 영향을 준 환경 요소를 검출하고, 그 환경 요소에 대응하기 위한 행위 등을 매쉬업에 추가함으로써 점진적으로 그런 문제를 해결하고 싶어 한다 [3]. 하지만 트리거-액션 프로그래밍 모델의 간결함은 반대로 사용자가 원하는 조건을 추가하는 것이 어렵다는 표현력의 한계로써 작용하고, 사용자가 매쉬업 어플리케이션의 신뢰성을 높이기 어렵게 한다.

본 논문에서는 기존 트리거-액션 프로그래밍 모델이 가지고 있던 표현력의 한계에서 기인하는 신뢰성 문제를 개선하는 방법으로 전제 상태 및 목표 상태의 검증 (Assertion) 방법을 제안한다. 제안하는 방법의 요구 사항은 다음과 같다: **(R1)** 전제 상태 검증을 통해 먼저 환경 조건을 검사하여야 한다. **(R2)** 목표 상태 검증을 통해 매쉬업 액션이 목표에 맞게 정상적으로 작동하였는지를 검사하여 매쉬업 어플리케이션이 환경 변화에서도 신뢰성 있게 (Reliable) 작동하여야 한다. **(R3)** 전제 상태 및 목표 상태를 검증하는 데 기존의 트리거-액션 모델이 가지고 있는 간결함과 직관성을 최대한 유지하여야 한다. 다음 요구 사항을 만족시키기 위해서, 먼저 전제 상태 및 목표 상태의 표현 방법에 대해 정의하였다. 그 다음에는 매쉬업 어플리케이션의 전제 상태 검증 절차, 그리고 매쉬업 액션의 형태에 따른 목표 상태의 검증 절차에 대해 제안한다.

본 논문의 구성은 다음과 같다: 2장에서는 본 연구에서 고려한 IoT 서비스의 전제 상태 관련 연구를 소개한다. 3장에서는 전제 상태 및 목표 상태 검증 방법을 제안한다. 4장에서는 제안된 방법의 요구 사항을 평가하기 위해서 사례 연구를 진행한 결과에 대해 논한다. 마지막으로 5장에서는 본 연구의 결론과 함께 향후 연구 계획에 관해 설명한다.

2. 관련 연구

Liang et al.은 다양한 IoT 서비스가 제공되는 건물 환경에서 트리거-액션 프로그래밍 모델로 정의된 IoT 서비스 매쉬업 어플리케이션의 오류 수정을 위해 전제 상태와 유사한 개념을 사용하였다 [4]. 다만 해당 연구에서는 환경 요소를 고려하기 위함이 아닌, 단순히 사용자가 잘못 생성한 매쉬업 어플리케이션을 수정하기 위한 목적으로써 전제 상태의 개념이 사용되었다.

Zhou and Ye는 사용자가 IoT 서비스를 사용할 때 해당 서비스를 실행하기 전 선행되어야 하는 IoT 서비스를 자동으로 실행하는 방법을 고안하였다 [5]. 선행 서비스를 실행하기 위해서 서비스 간에 경합이 발생할 수 있는데, 이 경합을 처리하는 방법을 2단계 잠금 규약(Two-Phase Locking Protocol)을 활용하여 제안하였다. 그러나 해당 연구는 IoT 서비스 제공자가 서비스 실행 환경에 대해 잘 알고 있으며 특정 IoT 서비스에 대한 선행 서비스의 목록을 사전에 정의해야 한다는 한계점을 갖는다.

3. 전제 상태 및 목표 상태 검증 방법

3.1 전제 상태 및 목표 상태의 형태

전제 상태란 트리거-액션 모델로 정의된 매쉬업 어플리케이션을 정상적으로 실행하기 위한 환경 상태 값들의 집합을 의미하고, 목표 상태란 사용자가 기대하는 매쉬업 어플리케이션의 실행 결과에 의해 변화된 환경 상태 값들의 집합을 의미한다. 환경 상태 값은 IoT 센서들의 측정값을 통해서 $env\_state = \langle sensor : status = \text{'}value\text{'} \rangle$ 로 정의된다.

전제 상태 및 목표 상태는 [그림 2]과 같이 기존 트리거-액션 모델 매쉬업에 추가된다.

그림 2. 전제 상태 및 목표 상태가 추가된 트리거-액션 기반 서비스 매쉬업의 예시

```
1: preconditions ← {env_state, ...}
2: goal_states ← {env_state, ...}
3:
4: ASSERT(preconditions)
5: if triggered then
6:     DO(action)
7: ASSERT(goal_states)
```

[그림 2]의 *preconditions* 및 *goal_states*는 각각 사용자가 정의한 전제 상태 및 목표 상태를 의미한다. 전제 상태 및 목표 상태에 포함된 각 환경 상태 값은 ASSERT 구문을 통해 트리거-액션 매쉬업의 전후에 검증 절차를 거치게 된다.

3.2 전제 상태의 검증

정상적인 트리거-액션 모델 매쉬업 어플리케이션의 작동을 위해서, 어플리케이션이 트리거 작동을 기다리는 동안 전제 상태의 모든 환경 상태 값을 항상 만족하여야 한다. 전제 상태에 포함된 각 상태 값들의 우선순위는 없으며, 만약 하나 이상의 상태 값이 불만족할 경우 (즉, $\langle sensor : status \neq `value' \rangle$ 인 경우) 사용자에게 매쉬업 어플리케이션을 작동할 수 없는 환경 조건이 되었음을 알린다.

3.3 목표 상태의 검증

전제 상태의 검증을 통해서 일차적으로 환경 조건에 대한 검증이 되었더라도, 새로운 환경 요소가 등장하거나 혹은 사용자가 놓친 환경 요소의 영향으로 매쉬업 어플리케이션의 결과가 사용자의 기대를 만족시키지 못할 수 있다. 목표 상태의 검증은 환경 상태 값의 변화를 통해 간접적으로 IoT 서비스 매쉬업 어플리케이션이 물리적인 환경 내에서 사용자의 기대와 일치하게 작동했는지 여부를 확인한다.

매쉬업 어플리케이션에 등록할 수 있는 액션의 형태는 시간적 특성에 따라 3가지로 분류할 수 있는데 [6], 목표 상태의 검증은 사용된 액션의 형태에 따라 다음과 같이 수행된다.

- 즉각적인 액션(Instant Actions)에는 IoT 카메라를 통한 사진 촬영 혹은 사용자에게 알림 전송 등이 포함된다. 이러한 액션은 물리적인 환경 상태에 영향을 주지 않기 때문에, 목표 상태 검증의 대상이 되지 않는다.
- 연장되는 액션(Extended Actions)에는 커피 머신에서 커피 추출 혹은 세탁기에서 세탁 등이 포함된다. 이러한 액션은 작동되는 동안 약간의 시간이 필요하고, 작동이 끝난 뒤 IoT 서비스는 초기 상태로 돌아온다. 따라서 목표 상태 검증은 먼저 매쉬업 액션이 실행되는 동안의 특징(소음 등)이 검출되는지 환경 상태 값을 확인하고, 실행이 끝난 뒤에는 정상적으로 초기 상태로 돌아왔는지 확인해야 한다.
- 유지되는 액션(Sustained Actions)에는 IoT 조명 작동 혹은 난방기 가동 등이 포함된다. 이러한 액션은 액션마다 목표 상태까지 도달하는 데 걸리는 시간이 다르고, 초기 상태로 되돌아오지 않는다. 따라서 목표 상태 검증은 일정한 시간 간격으로 환경 상태 값을 확인하여 목표 상태에 접근하고 있는지 확인해야 한다.

전제 상태의 검증과 동일하게, 목표 상태에 포함된 각 환경 상태 값들의 우선순위는 없으며, 하나 이상의 상태 값이 불만족할 경우 사용자에게 어플리케이션이 사용자 기대와 다르게 작동했음을 알린다.

4. 사례 연구를 통한 평가

본 논문에서 제안하는 전제 상태 및 목표 상태의 검증 방법을 평가하기에는 아직 초기 단계이다. 다만 현재까지 연구를 수행한 결과를 바탕으로 사례 연구(Case Study)를 구성하여 요구사항에 대한 정성적 평가를 진행하였다. 평가에 사용된 매쉬업 시나리오는 다음과 같다.

- 사용자가 생각하는 매쉬업 어플리케이션은 "방의 내부온도가 섭씨 26도를 넘을 경우 방의 공조 장치를 가동하라"이다. 방의 내부 온도는 Nest Thermostat의 *ambient_temperature_c* 값을 통해 측정할 수 있고, 공조 장치는 Samsung Room Air Conditioner가 사용된다고 가정한다.
- 사용자가 매쉬업 어플리케이션을 통해 원하는 것은 방의 내부온도가 적정 수준까지 내려가는 것이다. 본 예제에서는 섭씨 22도가 사용자의 희망 온도라고 가정한다.
- 사용자가 생각하는 매쉬업 어플리케이션의 전제 조건은 "방의 문과 창문이 닫혀있어야 한다"이다. 만약 열려있을 경우 공조 장치가 계속 작동하고 있어도 사용자의 희망온도까지 도달하는 것이 거의 불가능하기 때문이다. 따라서 전제 조건으로 "Wyze Door Contact Sensor 및 Wyze Window Contact Sensor가 닫혀 있는 상태여야 한다"가 포함된다. Wyze Contact Sensor의 닫힘 여부는 *open_close_state* 값을 통해 확인할 수 있다.

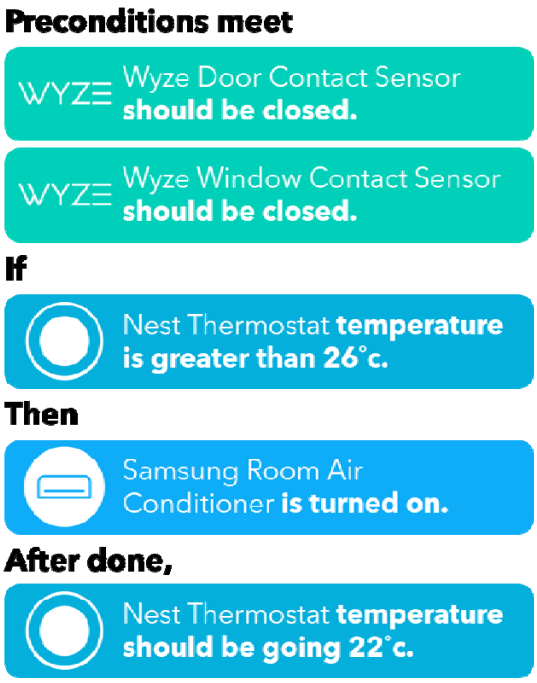


그림 3. 전제 상태 및 목표 상태 검증을 적용한 매쉬업 시나리오의 모형 사용자 인터페이스 예시

그림 4. 전제 상태 및 목표 상태 검증이 적용된 매쉬업 시나리오의 기저 표현

```
1: preconditions ← {⟨WyzeDoorContactSensor : open_close_state = 'close'⟩,
   ⟨WyzeWindowContactSensor : open_close_state = 'close'⟩}
2: goal_states ← {⟨NestThermostat : ambient_temperature_c = 22⟩}
3:
4: ASSERT(preconditions)
5: if ⟨NestThermostat : ambient_temperature_c > 26⟩ then
6:     DO(⟨SamsungRoomAirConditioner : AC_FUN_POWER = 'On'⟩)
7: ASSERT(goal_states)
```

[그림 3]은 본 연구에서 제안한 전제 상태 및 목표 상태의 검증 방법을 적용한 매쉬업 시나리오 어플리케이션의 모형 사용자 인터페이스(Mock-up User Interface)의 예시이고, [그림 4]은 매쉬업 시나리오 어플리케이션에 대한 기저 표현(Underlying Representation)이다. 기저 표현에는 각 IoT 서비스로부터 값을 요청하기 위한 변수명이 포함된다. 각 IoT 서비스에서 제공하는 실제 변수명은 서비스마다 다르기 때문이다. 사용자 인터페이스에서 [그림 3]과 같이 생성된 매쉬업 어플리케이션은 [그림 4]의 기저 표현으로 변환된다.

정성적 측면에서 전제 상태 및 목표 상태의 검증 방법은 두 가지의 장점을 가진다. 첫 번째로, 전제 상태의 검증을 통해서 일차적으로 적절한 환경에서만 작동되며 (R1), 목표 상태의 검증을 통해 실제로 사용자 기대에 맞게 작동되었는지를 확인할 수 있다 (R2). 기존 트리거-액션 모델에서도 트리거-액션의 중첩을 통해 이러한 상태 검증을 시도할 수 있으나, 이 경우 중첩으로 인해 간결함과 직관성을 잃게 된다. 본 논문에서 제시된 검증 방법을 사용할 경우 간결함을 잃지 않으며 표현력을 높일 수 있다 (R3).

두 번째로, 검증 과정에서 실패가 발생하였을 경우 사용자에게 알림을 줌으로써 사용자가 직접 가서 매쉬업 어플리케이션의 실행 결과를 확인하는 것보다 더 빠르고 효과적이다 (R3). 또한, 사용자들은 IoT 서비스의 자동화를 구성할 때 여러 횟수에 걸쳐 점진적인 절차를 통해 개발하는데 [3], 본 방법에서는 실패에 대한 알림을 통해 사용자가 실패를 인지하여 IoT 자동화를 점진적으로 발전시키는 데에 도움이 된다.

5. 결론 및 향후 연구

본 연구에서는 트리거-액션 프로그래밍 모델의 간결함을 유지하되 표현력을 개선할 수 있는 전제 상태 및 목표 상태 검증 방법에 대해 제안하였다. 전제 상태는 배포된 매쉬업 어플리케이션이 정상적으로 실행되기 위해서 항상 만족해야 하는 환경 상태 값들의 집합으로 구성되며, 목표 상태는 서비스 매쉬업이 사용자 기대(User Expectation)에 맞게 정상적으로 실행되었는지를 간접적으로 판단하기 위한 환경 상태 값들의 집합으로 구성된다. 사례 연구를 통해서 최종 사용자들이 본 방법을 통해 개선된 모델을 사용하여 효과적으로 매쉬업 문제를 파악하고 해결하는 데 도움이 될 수 있음을 확인하였다.

향후 연구 계획은 다음과 같다. 첫 번째로, 사용자가 트리거-액션 모델로 서비스 매쉬업을 작성하였을 때, 해당 정보를 바탕으로 전제 상태 및 목표 상태를 추천해주는 시스템에 관해 연구하고자 한다. 두 번째로, 프로그래밍 경험이 없는 실제 최종 사용자들을 대상으로 실증 연구를 진행하여 전제 상태 및 목표 상태 검증 방법의 타당성을 조사하고자 한다.

Acknowledgement

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2019R1A2C1087430).

참고문헌

[1] Zhao, Liping, et al. "User studies on end-user service composition: a literature review and a design framework." *ACM Transactions on the Web (TWEB)* 13.3 (2019): 15.

[2] Mi, Xianghang, et al. "An empirical characterization of IFTTT: ecosystem, usage, and performance." *Proceedings of the 2017 Internet Measurement Conference*. ACM, 2017.

[3] Woo, Jong-bum, and Youn-kyung Lim. "User experience in do-it-yourself-style smart homes." *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*. ACM, 2015.

[4] Liang, Chieh-Jan Mike, et al. "Systematically debugging IoT control system correctness for building automation." *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*. ACM, 2016.

[5] Zhou, Qian, and Fan Ye. "APEX: automatic precondition execution with isolation and atomicity in internet-of-things." *Proceedings of the International Conference on Internet of Things Design and Implementation*. ACM, 2019.

[6] Huang, Justin, and Maya Cakmak. "Supporting mental model accuracy in trigger-action programming." *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2015.

기능 블록 다이어그램에 대한 자동 뮤턴트 생성

Lingjun Liu O, 지은경, 배두환
한국과학기술원
{riensha, ekjee, bae}@se.kaist.ac.kr

Automated mutant generation for function block diagram programs

Lingjun Liu O, Eunkyoung Jee, Doo-Hwan Bae
School of Computing, Korea Advanced Institute of Science and Technology (KAIST)

Abstract

Since function block diagram (FBD) programs are widely used to implement safety-critical systems, effective testing for FBD programs has become important. Mutation testing is an error-based technique. It is highly effective but computationally expensive. To support developers for FBD testing, we propose an automated mutant generator for FBD programs. We designed this tool with the cost and equivalent mutant issues in consideration. We conducted experiments on real industrial examples to present the performance of this tool. The results show that this tool can generate mutants for FBD programs automatically with low probability of equivalent mutants and low cost. This tool can effectively support mutation analysis and mutation-adequate test generation for FBD programs.

1. Introduction

The testing for Programmatic Logic Controller (PLC) programs has become an important issue since the PLCs have been used to implement safety-critical systems. As Function Block Diagram (FBD) is one of the standard PLC programming languages defined in IEC 61131-3 [1], the effective testing of FBD programs is also necessary. Mutation testing is an effective technique to measure fault detection capability of test data and also a way to achieve the high quality required in critical software [2].

We propose a tool called MuGenFBD to automatically generate mutants for FBD programs based on the defined mutation operator [3]. We consider the cost of mutation testing and equivalent mutant raising issues in our approach. This tool can considerably ease the mutation analysis and the generation of mutation adequate test suite for FBD programs.

2. Related work

For FBD testing, some existing studies evaluated effectiveness of test suites [4] and generated mutation adequate test suites [5] by mutation testing method. Shin et al. [4] conducted mutation analysis to investigate the fault detection capability of test suite and defined five mutation operators: Constant Value Replacement (CVR), Inverter Insertion or Deletion (IID), Arithmetic Block Replacement (ABR), Logic Block Replacement (LBR), and Comparison Block replacement (CBR). Eniou et al. [5] proposed mutation-based test suite generation by model checking and defined six mutation operators. The difference between these two mutation operator sets is the additional timer block replacement operator defined in Eniou et al.'s work.

*This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2019R1I1A1A01062946) and the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. 2015-0-00250, (SW Star Lab) Software R&D for Model based Analysis and Verification of Higher-order Large Complex System).

Jee et al. [3] extended Shin et al.'s work and defined 13 mutation operators which includes all the mutation operators defined in the previous work. This mutation operator set covers most of defined functions and function blocks. Considering the misplacement of inputs, this mutation operator set also includes SWitched Inputs (SWI) operator.

3. Mutant generator for FBD programs

3.1. Overall process

The overall process of MuGenFBD is shown in Figure 1. MuGenFBD takes subject FBD programs in XML format and mutation operator selection as input. For each block, MuGenFBD applies the corresponding block replacement operator with number of inputs in consideration if the block replacement operator is selected; MuGenFBD applies the SWI operator with equivalent mutant issues in consideration if the

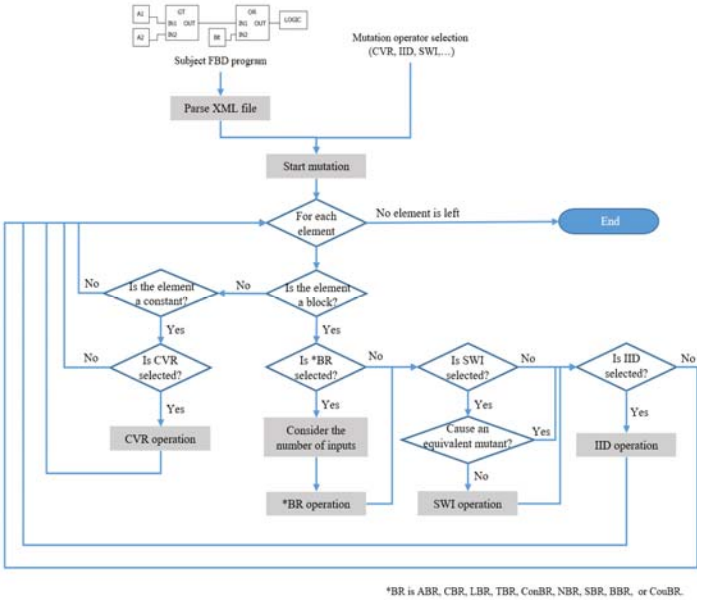


Figure 1. Overall process of MuGenFBD

Table 1. Probabilities of generating equivalent mutants

Subject	#blocks	#inputs	#outputs	#total mutants	Probability
simTRIP	3	3	2	14	0.000 (0/14)
simGRAVEL	3	3	4	8	0.000 (0/8)
LAUNCHER	4	2	1	13	0.077 (1/13)
FFTD	29	12	8	129	0.000 (0/129)
FRTD	29	12	8	129	0.000 (0/129)
VFTD	44	16	8	198	0.015 (3/198)
VRTD	44	17	8	199	0.015 (3/199)
MFTD	47	21	8	211	0.014 (3/211)
HB	19	6	1	114	0.000 (0/114)
combinedTD	437	221	92	1948	0.005 (9/1948)
Average	65.9	31.3	14	296.3	0.013

SWI operator is selected; MuGenFBD applies the IID operator if the IID operator is selected. For each constant, MuGenFBD applies the CVR operator when the operator is selected.

3.2. Issues of block replacement operators

Among all functions, there are some extensible functions. The number of inputs in extensible functions can be increased. The number of inputs is fixed in non-extensible functions. In the same group, there might be some extensible functions and some non-extensible functions, and there might be some different numbers of input or output between blocks. Thus, when we designed the block replacement mutation operators, we also considered whether the block is extensible or not and different number of inputs and outputs between blocks.

3.3. Equivalent mutant raising issues

An equivalent mutant is functionally equal to the original program. Calculating mutation score should exclude equivalent mutants. Thus, the chance for generating equivalent mutants should be limited. We found the SWI operator can possibly generate equivalent mutants. For instance, if we apply this mutation operator to the *AND* block, there's no influence on the logic (behavior). Hence, when applying the SWI operator, we carefully exclude functions that produce equivalent mutants, such as *ADD* (addition), *AND*, *OR*, etc.

4. Empirical evaluation

4.1. Subject programs

We chose our subject programs from the Korean Nuclear Instrumentation and Control System (KNICS) project's BP system [6]. The BP system includes about 20 modules that can be categorized into heartbeat (HB) monitoring modules and five types of trip decision modules: fix-falling (FFTD), fix-rising (FRTD), variable-rate-falling (VFTD), variable-rate-rising (VRTD), and manual-reset-falling (MFTD). To test scalability of the proposed approach, the combinedTD module is developed by combining several modules in the BP system. We designed three more subject programs, which are simTRIP, simGRAVEL, and LAUNCHER, to cover all function block groups. Table 1 shows the size information of each subject program.

4.2. Experiment

To demonstrate the performance of our tool, we applied our tool to subject programs. There are three aspects that we want to show the performance: (1) probability of producing equivalent mutants, (2) mutation operator selection, and (3) time efficiency.

Probability of producing equivalent mutants:

While selecting all the mutation operators, we executed MuGenFBD on all subject programs. As shown in Table 1, in half of cases, no equivalent mutants were found. In average, there is only 1.3 percent of probability of generating equivalent mutants. We utilized the SMT solver to identify equivalent mutants by finding a solution to distinguish the mutant from the original program. Without the automated mutant generation, we cannot evaluate the quality of mutation operator set.

Mutation operator selection:

MuGenFBD can support users freely select their desired mutation operators. First, we selected the original mutation operator set: CVR, IID, ABR, LBR, and CBR [4]. Second, we selected all the implemented mutation operators. In average, the extended mutation operator set generates over 44% more mutants than the original mutation operator set.

Time Efficiency:

To present the efficiency, we selected all the mutation operators and executed MuGenFBD on the large scale program called combinedTD. MuGenFBD took around three minutes to generate up to 1948 mutants. MuGenFBD is considered to provide practically usable performance.

5. Conclusion

This paper proposed an automated mutant generator called MuGenFBD for FBD programs by considering the cost and equivalent mutant issues. MuGenFBD achieved significantly low chance for producing equivalent mutants by only 1.3 percentage. For large scale program, MuGenFBD generated 1948 mutants in around three minutes. According to results, MuGenFBD can ease the mutation analysis and the automated generation of mutation-based test suites for FBD programs. In future work, we plan to develop a tool, which can automatically generate mutation-based test suite, with the help of MuGenFBD.

6. References

[1] International Electrotechnical Commission (IEC), "IEC61131-3: International Standard for Programmable Controllers - Part 3: Programming Languages," 2013

[2] M. R. Woodward, "Mutation testing—its origin and evolution," Information and Software Technology, Vol. 35, No. 3, pp.163-169, 1993

[3] E. Jee, J. Song, and D. H. Bae, "Definition and Applicatio of Mutation Operator Extensions for FBD Programs," Journal of KIISE: Transactions on Computing Practices, Vol. 24, No. 11, pp. 589-595, 2018 (in Korean)

[4] D. Shin, E. Jee E, D. H. Bae, "Empirical evaluation on FBD model-based test coverage criteria using mutation analysis," International Conference on Model Driven Engineering Languages and Systems, pp. 465-479, 2012

[5] E. P. Enoiu, D. Sundmark, A. Causevic, R. Feldt, andP. Pettersson, "Mutation-Based Test Generation for PLC Embedded Software using Model Checking," Proc. of the 28th International Conference on Testing Software and Systems, Lecture Notes in Computer Science, Vol. 9976, pp. 155-171, 2016.

[6] Doosan Heavy Industry & Construction, "Software design specification for the bistable processor of the reactor protection system," KNICS.RPS.SDS231-01, Rev.01, 2006 (In Korean)

딥러닝 기반 버그 추적 기법의 OOV 단어 처리를 위한 형태 정보와 문맥 정보 통합

김영경 O1, 김미수 2, 이은석 2

[1] 성균관대학교 소프트웨어학과, [2] 성균관대학교 전자전기컴퓨터공학과

{agnes66, misoo12, leees}@skku.edu

Integrating morphological and contextual information for handling out-of-vocabulary words in deep learning-based bug localization

Youngkyoung Kim O1, Misoo Kim2, Eunseok Lee2

[1]Department of Software Platform, Sungkyunkwan University,

[2]Department of Electrical and Computer Engineering, Sungkyunkwan University

요 약

딥러닝 기반 버그 추적 기법은 자동으로 결함의 위치를 찾기 위해 제안되었다. 대부분의 기존 기법은 버그 리포트와 소스 파일의 단어들을 기반으로 벡터를 생성하여 버그 추적 모델을 학습한다. 그러나 학습 단계에서 확인되지 않은 단어에 대해서는 벡터 정보를 얻을 수가 없기 때문에, 새로운 버그 추적 시 그 정확도를 보장하기 어렵다. 딥러닝 기반 버그 추적 기법을 위해, 본 연구에서는 단어 형태 및 문맥 정보를 활용한 out-of-vocabulary 단어의 벡터화 방법을 제안한다. 실험을 통해 제안 방법이 문맥 정보만 활용하는 것 대비 버그 추적 기법의 평균 역순위를 30.3% 향상시키는 것을 보였다.

1. 서 론

버그 추적이란 소프트웨어에 발생한 결함 위치를 찾는 활동이다. 개발자는 소프트웨어 사용자가 결함에 대해 기술한 버그리포트를 기반으로 소스 파일 내 결함 위치를 찾는다. 소프트웨어의 규모가 커짐에 따라 버그 추적을 위해 검토해야 하는 소스 파일의 수가 증가한다. 이는 결과적으로 소프트웨어 개발과 유지보수에 소모되는 시간과 비용을 크게 증가시킨다. 위 문제를 해결하기 위해 버그 추적을 자동화하는 것이 필요하다.

정보 검색 기반 버그 추적 기법은 대표적인 자동 버그 추적 기법 중 하나이다. 이 기법은 버그 리포트를 쿼리로 활용하여 소스 파일의 집합에서 결함이 있는 소스 파일을 자동으로 검색하는 기법이다 [1]. 검색 시 문맥에 따른 의미적 정보를 반영하기 위해 딥러닝 기반 버그 추적 기법이 제안되었다 [2-6].

Lam et al.은 과거에 해결된 버그 리포트와 결함 소스 파일 사이의 관계의 딥러닝을 통해, 새로운 버그 리포트에 대한 결함 파일을 추적하는 것을 처음으로 제안했다 [2]. 그 밖에도 다양한 딥러닝 모델을 활용하는 연구들이 제안되었으며, 기존 정보검색 기반 기법 대비 높은 추적 정확도를 보이고 있다 [3-6].

딥러닝 기반 버그 추적 기법은 과거 데이터의 단어 벡터를 기반으로 리포트와 파일 사이의 관계를 학습하기 때문에 학습 시 등장하지 않은 미확인 단어가 필연적으로 존재한다. 즉 out-of-vocabulary(OOV) 단어의 벡터를 획득하기 어렵다. 이를 보완하기 위해 최대한 많은 단어 말뭉치(예: Wiki Corpus, Google News)를 학습하거나[7], 무작위 실수나 단일 상수로 OOV 단어 벡터로 표현한다. 하지만 존재하는 모든 단어를 학습하는 것은 불가능하고, 임의의 벡터는 단어의 특성이 반영되지 않기 때문에, 결과적으로 버그 추적 정확도를 보장하기 어렵다. 즉, OOV 단어를 의미 있는 벡터로 나타내는 방법이 필요하다.

본 연구에서는 딥러닝 기반 버그 추적 모델의 성능 향상을 위한, OOV 단어의 벡터화 방법을 제안한다. 구체적으로 단어의 형태 및 문맥적 정보를 활용하여 OOV 단어를 의미 있는 벡터로 표현한다. 제안 방법의 효과를 검증하기 위해, 4개의 오픈 소스 소프트웨어의 12,295개의 버그 리포트를 사용하여 실험했다. 실험 결과, 문맥 정보만 활용하는 것 대비 딥러닝 기반 버그 추적 기술의 평균 역순위를 30.3% 향상시키는 것을 확인하였다. 즉, 제안하는 방법을 통해 OOV 단어를 의미 있는 벡터로 표현하여 버그 추적 성능을 향상시킬 수 있음을 보였다.

2. 관련 연구

2.1. 딥러닝 기반 버그 추적 모델

일반적으로 텍스트에 딥러닝 적용 시, 문서내의 단어의 순차적 정보를 반영할 수 있는 RNN(Recurrent Neural Network)이 효과적이라고 알려져 있다 [8]. 하지만 RNN은 계산 과정이 복잡해 비용이 높다. CNN(Convolutional Neural Network)은 데이터의 특성

추출에 용이하며 RNN보다 비용이 낮다는 장점이 있다.

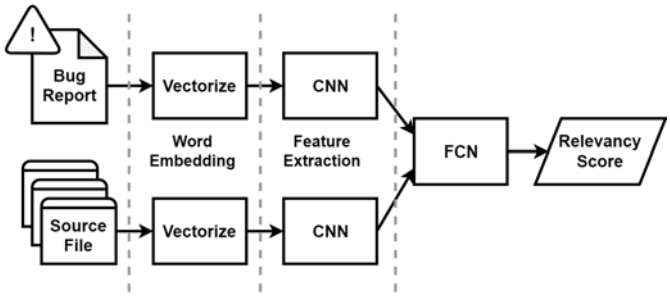


그림 1. CNN 기반 버그 추적 모델

최근 많은 연구들이 CNN기반 버그 추적 기법이 RNN기반 기법보다 버그 추적 정확도가 더 높음을 보였다 [3-6]. 그림 1은 Xiao et al.이 제안한 가장 최근의 CNN 기반 버그 추적 모델이다[6].

제안한 버그 추적 모델은 버그 리포트와 소스 파일들을 입력으로 받아 벡터화 과정을 거친다. 벡터화된 문서는 CNN을 통해 특성을 추출하는데 사용된다. 추출한 특성은 FCN(Fully Connected Network)에 입력되어 입력 값들의 relevancy score를 출력한다. 이 relevancy score를 기반으로 정렬된 소스 파일 목록을 개발자에게 제공한다.

문서 벡터화는 딥러닝 기반 버그 추적 기법의 가장 첫 단계로 가능한 많은 정보를 벡터에 반영하는 것이 필요하다.

2.2. 딥러닝 기반 버그 추적 모델의 문서 벡터화

대부분의 딥러닝 기반 버그 추적 기법은 버그 리포트와 소스 파일의 단어 집합을 문서로 취급하고, 벡터화된 두 문서 간 관계를 학습한다.

문서 벡터화를 위한 첫 번째 단계는 단어를 벡터화하는 것이다. 우선 문서 내 유의미한 단어만 남기기 위해 전처리 과정을 거쳐야 한다. 전처리 단계에서는 1) 각 문서들을 단어의 집합으로 자르고, 2) 전치사나 관사 등 지나치게 빈번히 등장하는 불용어를 제거하고, 3) 필요에 따라 어근화를 통해 단어의 기본 형태를 추출한다. 위 과정을 거친 단어들을 벡터로 표현하고, 문서를 벡터화 하여 버그 추적 모델을 학습한다. 벡터 표현을 위해 워드 임베딩 모델 [9]을 사용한다.

문자의 조합으로 이루어지는 단어는 무한한 집합이기 때문에 모든 단어를 가지는 말뭉치는 존재하지 않는다. 이 때문에 단어에 기반한 문서 벡터화 시 OOV 단어가 발생한다. 기존 연구들은 OOV 단어 처리를 위해 *<UNK>* 토큰을 사용한다. *<UNK>* 토큰은 주로 무작위 실수로 이루어진 벡터나 하나의 상수로만 이루어진 벡터로 표현된다 [2-6]. 이는 단어의 특성과 문서의 의미를 반영하지 못하는 방법으로 딥러닝 기반 버그 추적 모델의 성능을 저하시킬 수 있는 문제 중 하나이다 [3].

문서는 단어의 집합으로 취급할 수도 있지만 문자의 집합으로 취급할 수도 있다. 문자는 언어의 문자체계마다 수는 다르지만 유한한 집합이다. 문서를 단어로 나타내면 OOV 단어가 발생하지 않는다. Xiao et al. 은 버그 리포트와 소스 파일의 문자 집합을 문서로 취급하는 버그 추적 기법을 제안했다 [5]. 이 기법은 OOV 문제가 발생하지 않지만 단어의 문맥적 정보를 활용하지 않는다. 불충분한 정보 활용은 버그 추적 성능 저하의 원인이다. 따라서 단어의 문맥적 정보를 유지하며 OOV 문제를 해결할 수 있는 새로운 접근 방식이 필요하다.

3. 제안 내용

본 연구에서는 기존 딥러닝 기반 버그 추적 모델의 벡터화 단계에서 OOV 단어를 유의미한 벡터로 표현하기 위한 방법을 제안한다. 그림 2는 제안하는 OOV 단어 처리 방법이다.

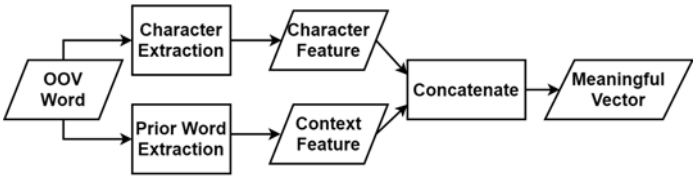


그림 2. 제안하는 OOV 단어 처리

OOV 단어에서 얻을 수 있는 정보는 크게 1) 단어 자체의 형태에서 얻을 수 있는 정보 2) 주변 단어와의 문맥에서 얻을 수 있는 정보 두 가지로 나눌 수 있다.

첫 번째로 단어 자체의 형태에서 정보를 얻기 위해 단어를 문자의 n-gram으로 학습하는 스킵-그램(skip-gram)모델을 사용한다. 단어의 시작과 끝을 나타내기 위해 *'<'*와 *'>'*를 사용한다. 26개의 문자로 이루어진 영어는 $26*27^{N-1}$가지의 토큰이 생길 수 있다. 단순히 문자를 벡터로 치환하는 것 보다 많은 형태적 정보를 포함할 수 있다. 예를 들어 단어 *'hello'* 를 3-gram으로 나타내면, *'<he', 'hel', 'ell', 'llo', 'lo>'* 로 표현된다.

즉 한 단어는 n-gram 토큰화를 통해 $T_1,\ldots,T_N$으로 나타낼 수 있다. 스킵-그램 모델은 식 (1)을 최대화하도록 학습된다.

$$\arg\max \sum_{i=1}^{N} \sum_{T_s \in S_i} \log\left(p(T_s|T_i)\right) \quad (1)$$

S_i는 단어 토큰 T_i를 둘러싸고 있는 토큰 집합이다. 토큰 T_i가 등장했을 때 전후에 T_s가 발견될 확률을 학습한다. 학습이 완료되면, 토큰 $26*27^{N-1}$개에 대한 벡터가 생긴다. K개의 토큰으로 나눠진 OOV 단어의 형태 벡터 V_m는 식 (2)와 같이 나타낼 수 있다.

$$v_m = \sum_{i=1}^{K} T_i \quad (2)$$

식 (2)는 식 (1)을 통해 학습된 T_i의 합을 계산한다.

이를 통해 OOV 단어의 형태 토큰 정보들을 벡터에 반영할 수 있다.

두 번째는 OOV 단어 벡터에 문맥 정보를 반영하는 단계이다. 우선 OOV 단어가 포함된 문장과 문장 내 단어들을 추출하고, 기 학습된 임베딩 모델에 학습된 단어의 벡터를 추출한다. 다음으로 OOV 단어와 각 단어 사이의 거리를 구한다. 마지막으로 각 단어의 벡터 값과 다음으로 각 단어의 벡터 값과 거리의 역수를 곱하여 OOV 단어의 문맥 정보를 벡터로 표현한다. 식 (3)은 이를 계산하는 수식이다. OOV 단어 T의 문맥 벡터 v_c는 식 (3)과 같이 나타낼 수 있다.

$$v_c = \sum_{w \in S} \frac{1}{dist(T,W)} w \quad (3)$$

이 때 S는 OOV 단어가 등장하는 문장의 OOV가 아닌 단어 집합이고, $dist(T,W)$는 OOV 단어와 단어 W 사이의 거리이다.

마지막으로 형태 벡터 v_m과 문맥 벡터 v_c를 통합(concatenate)한다. 통합된 최종 벡터는 버그 추적 시 입력된 OOV 단어의 벡터로 표현된다.

4. 실험

4.1. 실험 설정

실험 데이터: 성능 비교 실험을 위해 4개의 오픈 소스 프로젝트를 사용했다. 이는 기존 연구들이 활용한 데이터와 동일하다 [3-6].

추가적으로 본 연구에서 해결하고자 하는 OOV 문제를 확인하기 위해 각 프로젝트에 OOV 단어 비율을 분석하였다. 임베딩 모델 학습을 위해서는 300만 개의 단어가 있는 Google News 말뭉치를 사용했다. OOV 단어의 비율은 프로젝트의 단어 말뭉치 중 실험 프로젝트에는 등장하지만 Google News에는 등장하지 않는 단어들로, 버그 추적 시 벡터화될 수 없는 단어의 비율을 설명한다. 표 1은 각 프로젝트의 버그 리포트, 소스 파일 수와 OOV 단어 비율을 보인다.

표 1. 실험 프로젝트

프로젝트	버그 리포트 수 (개)	소스 파일 수 (개)	OOV 비율 (%)
Tomcat	1,056	1,552	52.7
AspectJ	593	4,439	47.4
Eclipse UI	6,495	3,454	34.5
SWT	4,151	2,056	29.9
Total	12,295	11,501	36.5

실험 데이터는 총 12,295개의 버그리포트와 11,501개의 소스 파일로 구성되어 있다. Tomcat은 OOV 단어 비율이 52.7%로 가장 높다. OOV 단어의 평균적인 비율은 평균 36.5%이다. 즉, 기존 프로젝트들이 모두 OOV 단어 문제를 내포하는 것을 보여준다.

실험 패러미터 설정: 버그 추적 모델에는 그림 1의 모델을 사용했다. 특징 추출 단계의 CNN은 미리 정의된 필터 사이즈가 필요하다. 본 연구에서는 Xiao et al.의 연구를 참고하여 CNN 모델의 패러미터를 설정했다[6]. 버그 리포트의 특징 추출 CNN은 6x6 단일 사이즈의 필터 100개를 사용했다. 소스 파일 추출 CNN은 3X3, 4X4, 5X5 사이즈의 필터를 사용한다. 단어 형태 정보 추출에는 기존 연구에서 효과적으로 사용된 3-gram을 사용했다 [10].

평가 지표: 기존 검색 연구들이 널리 사용하는 MRR(Mean Reciprocal Rank)을 평가지표로 사용해 성능 향상 정도를 비교했다. MRR은 식 (4)와 같이 나타낼 수 있다.

$$\mathrm{MRR} = \frac{1}{|N|} \sum_{i=1}^{|N|} \frac{1}{rank_i} \quad (4)$$

N은 버그 리포트 집합이고 $rank_i$는 해당 버그 리포트에 대해 가장 높은 순위로 추적된 결함 소스 파일의 순위다.

4.2. 실험 결과

기존의 딥러닝 버그 추적 기법은 대부분 문맥 정보만 활용한다. 제안 기법을 통한 버그 추적 성능 향상 효과를 확인하기 위해, 단어의 문맥 정보만 반영한 추적 성능과 비교했다. 표2는 실험 결과이다.

표 2. 실험 결과

프로젝트	기존 방법	제안 방법	성능 향상(%)
Tomcat	0.14	0.40	65.00
AspectJ	0.42	0.55	23.64
Eclipse UI	0.42	0.71	40.85
SWT	0.39	0.36	-8.33
평균	0.34	0.50	30.29

실험 결과 4개의 프로젝트 중 3개의 프로젝트에서 성능이 향상됨을 확인했다. OOV 단어 비율 약 52.7%로 가장 높은 Tomcat은 성능이 가장 많이 향상되었다. 반면 OOV 단어의 비율이 가장 낮은 SWT는 성능 저하가 있었다. 그러나 평균적으로는 30.3%의 성능 향상을 보일 수 있었다. 따라서 OOV 비율에 따라서 형태 정보와 문맥 정보의 가중치를 조절하는 등 형태 정보와 문맥 정보를 보다 효율적으로 결합하는 방법이 필요하다.

5. 결론 및 향후 연구

본 연구에서는 딥러닝 기반 버그 추적 기법의 OOV

문제를 해결하기 위해, 단어의 형태적 정보와 문맥 정보를 함께 활용해 OOV 단어를 유의미한 벡터로 만들어 처리하는 방법을 제안했다. 제안 방법은 버그 추적 모델의 성능을 평균적으로 30.3% 향상 시키는 것을 확인했다.

단어의 형태적 특징에는 문자의 순서 외에도 여러가지가 있다. 또한 버그 리포트의 자연어와 소스 파일의 코드는 서로 다른 형태적 특징을 가진다. 향후 연구에서는 다양한 형태적 정보를 반영하고, 버그 리포트와 소스 파일 각각의 특성을 반영한 버그 추적 연구를 진행할 것이다.

Acknowledgement

이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2018R1D1A1B07050073, 2019R1A2C2006411).

참고문헌

[1] Saha, R., Lease, M., Khurshid, S., and Perry, D., “Improving bug localization using structured information retrieval,” in Proceedings of the 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE 2013), pp. 345–355, 2013.

[2] Lam, A., Nguyen, A., Nguyen, H., and Nguyen, T., “Combining deep learning with information retrieval to localize buggy files for bug reports (n),” in Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015), pp. 476–481, 2015.

[3] Xiao, Y., Keung, J., Bennin, K., and Mi, Q., “Machine translation-based bug localization technique for bridging lexical gap,” Information and Software Technology, vol. 99, pp. 58–61, 2018.

[4] Xiao, Y., Keung, J., Mi, Q., and Bennin, K., “Improving bug localization with an enhanced convolutional neural network,” in Proceedings of 2017 24th Asia-Pacific Software Engineering Conference (APSEC 2017), pp. 338–347, 2017.

[5] Xiao, Y., and Keung, J., “Improving Bug Localization with Character-Level Convolutional Neural Network and Recurrent Neural Network,” in Proceedings of 2018 25th Asia-Pacific Software Engineering Conference (APSEC 2018), pp. 703–704, 2018.

[6] Xiao, Y., Keung, J., Bennin, K., and Mi, Q., “Improving bug localization with word embedding and enhanced convolutional neural networks,” Information and Software Technology, vol. 105, pp. 17–29, 2019.

[7] Zhou, Z.H., and Feng, J., “Deep Forest”, arXiv preprint arXiv:1702.08835, 2017.

[8] Yin, W., Kann, K., Yu, M., and Schütze, H., “Comparative study of CNN and RNN for natural language processing,” arXiv preprint arXiv:1702.01923, 2017.

[9] Mikolov, T., Chen, K., Corrado, G., and Dean, J., “Efficient estimation of word representations in vector space,” arXiv preprint arXiv:1301.3781, 2013.

[10] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T., “Enriching word vectors with subword information,” Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146, 2017.

소프트웨어 결함 예측 모델을 위한 N-gram 로그 확률 메트릭

원은지○, 남재창
한동대학교 정보통신공학과
21931007@handong.edu, jcnam@handong.edu

N-gram Log Probability Metric for Software Defect Prediction

Eunji Won○, Jaechang Nam
Department of Information Technology, Handong Global University

요 약

소프트웨어 결함 예측은 소프트웨어의 품질 보증을 위해 활발히 연구되는 기술이다. 소프트웨어의 결함을 예측하기 위해서는 예측 모델 훈련을 위해 사용되는 소프트웨어 복잡도 메트릭의 역할이 중요하다. 다양한 방법으로 소프트웨어의 복잡도를 측정하는 메트릭이 발전해왔지만 언어 모델을 이용한 메트릭 연구는 흔치 않다. 2013년도에 처음으로 N-gram 언어 모델을 이용하여 N-gram Log Probability(NGLP)라는 새로운 데이터 기반 복잡도 메트릭이 제안되었지만, NGLP 메트릭이 실증적으로 결함 예측 모델의 성능을 향상시키는지는 연구되지 않았다. 본 논문의 목적은 Allamanis et al.가 제안한 NGLP 메트릭이 결함 예측의 성능 향상에 얼마나 영향을 미칠 수 있는지 조사하는 것이다. 이를 위해 우리는 8개의 프로젝트를 대상으로 NGLP 메트릭을 사용한 것과 사용하지 않은 결함 예측 모델의 성능을 비교하는 실증적 연구를 수행한다. 그 결과 모든 데이터셋에 대해 NGLP를 메트릭을 추가한 결함 예측 모델이 기존의 메트릭만을 사용했을 때 보다 결함 예측 성능을 향상시킨다는 것을 확인할 수 있었다.

1. 서 론

소프트웨어 결함 예측 모델의 성능은 소프트웨어 복잡도 메트릭에 달려있다 [1]. 과거에는 버전 관리 시스템 및 이슈 추적기 등의 소프트웨어 아카이브의 사용이 활발해지면서 결함 예측 메트릭에 대한 연구가 발전해왔다 [3]. 특히 메트릭 연구는 소스코드의 크기 및 복잡도를 측정하는 코드 메트릭과 개발자 수나 소스코드 변경 개수와 같은 프로세스 메트릭의 형태로 나눌 수 있다.

다양한 방법으로 결함을 측정하는 메트릭이 발전해왔지만 언어 모델을 이용한 메트릭 연구는 흔치 않다. 2013년도에 Allamanis et al. [2]이 처음으로 N-gram 언어 모델을 이용하여 N-gram 로그 확률(NGLP)이라는 새로운 데이터 기반 복잡도 메트릭을 제안하였다.

Allamanis et al. [2]이 언어 모델을 이용하여 NGLP 메트릭을 제안하였지만 결함 예측 모델에 NGLP 메트릭이 미치는 영향은 조사하지 않았다. 즉 실증적으로 결함 예측 모델을 만들어 모델의 성능을 측정하지 않았다. Allamanis et al. [2] 연구는 프로젝트 전체의 코딩 관행을 탐구하기 위해 GitHub의 14,807개의 오픈 소스 Java 프로젝트의 10억 개 이상의 토큰을 이용하여 소스 코드에 대한 최초의 기가-토큰 언어 모델(giga-token probabilistic language model)을 만드는 것을 목표로 하였기 때문이다. 또한 Allamanis et al. [2] 연구 이후 NGLP 메트릭을 위한 후속 연구는 진행되지 않았다.

본 논문에서는 Allamanis et al. [2]가 제안한 NGLP 메트릭이 결함 예측의 성능 향상에 얼마나 영향을 미칠 수 있는지 조사하는 것을 목표로 한다. 이를 위해 8개의 프로젝트를 대상으로 NGLP 메트릭을 사용한 것과 사용하지 않은 결함 예측 모델의 성능을 비교하는 실증적 연구를 수행했고, NGLP 메트릭을 추가한 결함 예측 모델이 기존 결함 예측 성능을 향상시킨다는 것을 확인할 수 있었다.

2. 관련 연구

과거 메트릭 연구는 소스코드의 크기 및 복잡도를 측정하는 코드 메트릭과 개발자 수나 소스코드 변경 개수와 같은 프로세스 메트릭의 형태로 발전돼 왔다. 또한 변경/코드 메트릭 천(Churn), 변경/코드 엔트로피, 인기 및 개발자 상호작용과 같은 다양한 메트릭이 제안되었다 [4], [5], [6], [7], [8], [9], [10], [11].

Allamanis et al. [2]은 프로젝트 전체의 코딩 관행을 탐구하기 위해 GitHub의 14,807개의 오픈 소스 Java 프로젝트의 새로운 말뭉치(Corpus)를 사용한다. Allamanis et al. [2]은 3억 5200만 라인으로 구성된 소스코드 말뭉치를 이용하여 N-gram 언어 모델 훈련시킨다. 이 언어 모델은 10억 개 이상의 토큰을 이용하여 훈련되며 소스 코드에 대한 최초의 기가-토큰 확률 언어 모델이다. 또한, Allamanis et al. [2]은 N-gram 로그 확률(NGLP) 메트릭과 식별자 정보 메트릭을 제안한다. 식별자 정보 메트릭은 소스 코드 파일이 얼마나 특정한 도메인에 한정돼 있는지 측정하는 메트릭으로 결함 예측 보다는 코드 재사용과 관련이 있기에 본 연구에서는 NGLP 메트릭에 집중하였다.

Allamanis et al. [2]은 NGLP 메트릭을 분석하기 위해 NGLP와 기존의 복잡도를 측정하는 메트릭인 line of code(LOC)와 McCabe's cyclomatic complexity(CC)의 상관관계를 조사한다. 구체적으로 NGLP와 LOC 그리고 CC 각각 spearman's rank correlation를 검사함으로써 NGLP가 기존의 메트릭들과 관련이 있을 뿐만 아니라 전반적으로 복잡성을 추정할 수 있는 합리적인 메트릭임을 발견했다. 하지만 실증적으로 결함 예측 성능에 어떤 영향이 있는지는 측정하지 않았다.

Allamanis et al. [2]가 제안한 NGLP는 토큰 시퀀스의 로그 확률이다. NGLP 메트릭은 말뭉치를 이용하여 N-gram 모델을 학습시킨 후 각 토큰 시퀀스가 가질 수 있는 확률을 모두 곱한 뒤 로그를 취해 주어진 소스코드의 복잡도를 측정한다. NGLP는 직관적으로 수신기가 토큰 시퀀스를 받았을 때 수신기의

"놀라움"의 척도 정도로 해석할 수 있다. 예를 들어, 어떠한 토큰 시퀀스의 NGLP 값이 크다면 이는 그 토큰 시퀀스가 기존 말뭉치에서 자주 보이지 않았으며 발생할 확률이 높지 않아 새로운 정보이고 복잡한 코드라고 해석될 수 있다.

Shippey et al. [4]은 결함이 있는 자바 소스코드의 특징들을 식별하기 위해 소스코드의 추상 구문 트리(AST) 노드를 이용해 N-gram 모델을 만들어서 사용자에게 잠재적으로 결함이 있는 코드를 식별하는데 도와주는 해법을 제안했고 N-gram 모델이 결함 예측 모델에 도움이 됨을 검증하였다.

본 연구는, Shippey et al. [4]의 연구와 다르게 소스코드 수정정보를 담고 있는 커밋 자체를 토큰화하여 N-gram을 훈련시켜 NGLP 메트릭을 생성한다. 또한 Shippey et al. [4]의 연구에서 사용된 AST N-gram 메트릭은 메소드 단위 결함 예측에서 검증을 하였지만 본 연구에서는 소스 코드 수정 단위의 Just-in-time 결함 예측 모델의 성능에 끼치는 영향을 검증하였다.

3. 실험 설정

3.1 NGLP 메트릭 생성 과정

본 연구에서는 Allamanis et al. [2] 연구와 동일하게 n이 3인 trigram을 사용한다. git show 명령어를 통해 반환되는 커밋 텍스트 정보로 커밋 말뭉치를 생성하고 이를 이용하여 trigram을 학습시켜 토큰 시퀀스에 대한 확률들을 미리 계산한다. NGLP 메트릭 값을 얻기 위해서는 NGLP 메트릭을 구해야 할 커밋을 먼저 토큰화하여 trigram 형태로 만든다. 그 다음 trigram 토큰 시퀀스에 해당하는 확률을 모두 곱해 마지막에 음의 로그를 취한다. 본 연구에서는 NGLP 메트릭을 구하기 위해 single trigram 모델과 master trigram 모델을 학습시킨다. Single trigram model은 한 프로젝트의 기간을 정해 그 사이의 모든 커밋들을 훈련 데이터로 사용하여 trigram을 훈련시키지만, master trigram 모델은 여러 개의 프로젝트 커밋을 학습 데이터로 사용하여 master 모델을 생성한다. 본 연구에서는 8개의 프로젝트에서 수집한 데이터를 훈련 데이터로 사용하여 master trigram 모델을 훈련시킨다. 본 연구에서는 single trigram 모델을 이용하여 얻은 메트릭을 Single NGLP, master 모델을 이용하여 얻은 메트릭을 Master NGLP로 정의한다.

3.2 연구 질문

과거 Allamanis et al. [2]가 제안한 NGLP 메트릭이 결함 예측 성능 향상에 실증적으로 얼마나 영향을 미칠 수 있는지 조사하기 위해 다음과 같은 연구 질문(RQ)들을 설정한다.

- RQ1. Single NGLP 메트릭을 결함 예측 모델에 추가했을 때 예측 성능을 향상시키는가?
- RQ2. Master NGLP 메트릭을 결함 예측 모델에 이용했을 때 예측 성능을 향상시키는가?

RQ1에서, Kamei et al. [1] 이 제안한 메트릭을 사용하여 만든 결함 예측 모델과 그 모델에 Single NGLP 메트릭을 추가시켜 만든 모델의 성능을 비교한다.

RQ2에서, Kamei et al. [1] 이 제안한 메트릭을 사용하여 만든 결함 예측 모델과 그 모델에 Master NGLP 메트릭을 추가시켜 만든 모델의 성능을 비교한다.

3.3 데이터셋

표 1. 8개의 결함 데이터셋

Project	# of instances		# of metrics
	All	Buggy(%)	
Camel	4821	429 (8.17%)	8
Eagle	105	5 (4.55%)	
Flink	2982	902 (23.22%)	
Groovy	1231	173 (12.32%)	
Jena	1264	100 (7.33%)	
Juddi	26	2 (7.14%)	
Metamodel	159	2 (1.24%)	
Nutch	461	41 (8.17%)	

최근 35개월 동안 GitHub에 커밋(change)된 수가 100개 이상이며 지라(Jira)를 통해 이슈를 관리하고 있는 Apache 오픈소스 프로젝트 중 8개의 프로젝트를 뽑아 실험 데이터셋으로 사용한다. 8개의 데이터셋 모두 커밋 단위이며 Kamei et al. [1]에서 제안된 메트릭을 사용하였다. 8개의 메트릭은 변경된 디렉토리 수, 수정된 라인 수, 추가된 라인 수, 삭제된 라인 수, 파일 간 변경된 코드의 분포, 수정된 파일을 고친 개발자의 수, 수정된 파일이 변경된 횟수 메트릭이다.

NGLP 메트릭을 구하기 위해서는, N-gram 모델을 학습시켜야 한다. 우리는 8개의 프로젝트 첫 커밋부터 2016년 12월까지의 모든 커밋들을 모아 N-gram의 학습 데이터로 사용한다. 또한 결함 예측 모델을 만들기 위해서 2017년 1월부터 2019년 6월까지 약 30개월 동안 생성된 커밋들을 모아 훈련 및 검증 데이터셋으로 사용한다.

3.4 예측 모델

기계학습 알고리즘으로 로지스틱 회귀 모델을 사용한다. 우리는 예측 모델에 Weka의 기본 옵션을 적용한다. 우리는 기계 학습 모델을 평가하기 위해 10겹 교차 검증을 사용한다. 10겹 교차 검증 방법은 주어진 데이터 세트를 10개의 폴드로 나눈다. 그 다음 9개의 폴드 부분들을 훈련 데이터로, 나머지 1개의 폴드 부분을 테스트 데이터로 사용한다. 10겹 교차 검증은 100번 반복했다.

3.5 성능 지표

본 연구에서는 결함 예측의 성능을 측정하기 위한 성능 지표로 AUC가 사용된다. AUC는 다른 모델들을 비교할 때 합리적인 성능 지표로 잘 알려져 있다 [12], [13], [14], [15]. AUC는 다양한 예측 임계값들의 리콜과 거짓 양성 비율을 표시하여 계산한다. 따라서 AUC는 다른 모델의 전반적인 예측 성과를 비교하기에 적합한 척도이다.

본 연구에서는 또한 프로젝트 별 모델들의 AUC와 모든 프로젝트들의 평균 AUC 결과가 통계적으로 유의미한 차이임을

확인하기 위해 유의수준 5%에서 Mann-Whitney U-test를 실행한다.

3.6 다양한 설정을 가진 예측 모델들

NGLP 메트릭이 결함 예측 성능 향상에 실증적으로 얼마나 영향을 미칠 수 있는지 조사하기 위해 *Baseline, Single-NGLP, Master-NGLP* 3가지 모델을 설정한다. *Baseline*은 본 연구의 기준선이 되는 모델로 Kamei et al. [1] 이 제안한 메트릭을 사용하여 만든 결함 예측 모델이다. *Single-NGLP*는 *Baseline* 모델에 각 프로젝트 기반으로 학습한 N-gram 모델로부터 얻은 Single NGLP 메트릭을 추가하여 만든 결함 예측 모델이다. *Master-NGLP*는 *Baseline* 모델에 모든 프로젝트를 학습한 master N-gram 모델로부터 얻은 Master NGLP 메트릭을 추가하여 만든 결함 예측 모델이다.

4. 결과

표 2는 각 데이터셋에 대해서 *Baseline*, *Single-NGLP* 그리고 *Master-NGLP* 모델의 평균 AUC 값을 보여준다. 본 연구에서는 로지스틱 회귀 모델을 분류기로 사용하여 결함 예측 모델을 구축한다. 또한 유의수준 5%에서 Mann-Whitney U-test를 통해 두 모델의 차이가 통계적으로 유의미한지 검증한다. 표 2의 다섯 번째 열부터 마지막 열까지는 프로젝트 별 모델들의 AUC 성능 차이 및 통계검증 후 얻은 p-value값을 보여준다. 또한 모든 프로젝트들의 평균으로 통계검증한 후 얻은 p-value값은 마지막 Avg.행의 괄호 안에 표기한다. p-value값이 0.05보다 작아 통계적으로 유의미하게 차이가 나는 경우 밑줄로 표시한다.

표 2. 로지스틱 회귀 모델을 분류기로 사용한 *Baseline*, *Single-NGLP*, *Master-NGLP* 모델 실험 결과

Project	*Baseline* (A)	*Single-NGLP* (B)	*Master-NGLP* (C)	AUC Difference		
				B - A (p-value)	C - A (p-value)	C - B (p-value)
Camel	0.581	0.594	0.608	0.013 (0.187)	0.027 (0.304)	0.014 (0.768)
Eagle	0.524	0.591	0.598	0.068 (0.000)	0.075 (0.000)	0.007 (0.760)
Flink	0.591	0.64	0.644	0.048 (0.000)	0.053 (0.000)	0.005 (0.859)
Groovy	0.594	0.632	0.636	0.038 (0.000)	0.042 (0.000)	0.004 (0.987)
Jena	0.593	0.621	0.625	0.028 (0.001)	0.032 (0.001)	0.004 (0.914)
Juddi	0.581	0.612	0.616	0.031 (0.000)	0.035 (0.000)	0.004 (0.977)
Metamodel	0.594	0.624	0.627	0.030 (0.000)	0.033 (0.000)	0.003 (0.997)
Nutch	0.476	0.505	0.532	0.029 (0.000)	0.056 (0.000)	0.027 (0.223)
Avg.	**0.567**	**0.602**	**0.611**	**0.036** (0.038)	**0.044** (0.005)	**0.008** (0.574)

4.1 RQ1: Single NGLP 메트릭을 결함 예측 모델에 추가했을 때 예측 성능을 향상시키는가?

표 2의 두 번째 열과 세 번째 열은 *Baseline* 모델과 *Single-NGLP* 모델의 평균 AUC 값을 각각 보여준다. 표 2에서 8개의 모든 데이터셋에 대해 *Single-NGLP*가 *Baseline* 모델보다 성능이 높다는 것을 관찰한다. 예를 들어, Eagle의 경우 *Baseline* 모델의 평균 AUC 값은 0.524이며 *Single-NGLP* 모델은 0.591이다. 또한 그 모델의 성능 차이는 0.068이다. *Baseline* 모델과 *NGLP* 모델의 성능 차이는 0.013에서 0.068로 다양하다. *Baseline* 모델의 8개 데이터셋의 평균 AUC 값은 0.567, *Single-NGLP* 모델은 0.602로 그 둘의 성능 차이의 평균은 0.036이다. 두 모델의 성능은 Camel의 경우를 제외하고 모든 프로젝트에서 통계적으로 유의미하게 차이가 난다. 또한 모든 프로젝트의 평균으로 통계검증한 경우에도 p-value가 0.038이므로 통계적으로 유의미한 차이가 난다. 그러므로 *Single-NGLP* 모델이 결함 예측 성능 향상에 긍정적인 영향이 있음을 확인하였다.

4.2 RQ2: Master NGLP 메트릭을 결함 예측 모델에 이용했을 때 예측 성능을 향상시키는가?

표 2의 두 번째 열과 네 번째 열은 *Baseline* 모델과 *Master-NGLP* 모델의 평균 AUC 값을 각각 보여준다. 표 2에서 8개의 모든 데이터셋에 대해서 *Master-NGLP* 모델이 *Baseline* 모델보다 성능이 높다는 것을 관찰한다. 예를 들어, Nutch 경우 *Baseline* 모델의 평균 AUC 값은 0.476이며 *Master-NGLP* 모델은 0.532이다. 또한 그 모델의 성능 차이는 0.056이다. *Baseline* 모델과 *Master-NGLP* 모델의 성능 차이는 0.027에서 0.075로 다양하다. *Baseline* 모델의 8개 데이터셋의 평균 AUC 값은 0.567, *Master-NGLP* 모델은 0.567로 그 둘의 성능 차이의 평균은 0.044이다. 두 모델의 성능은 Camel의 경우를 제외하고 모든 프로젝트에서 통계적으로 유의미하게 차이가 난다. 또한 모든 프로젝트의 평균으로 통계검증한 경우에도 p-value가 0.005이므로 통계적으로 유의미한 차이가 난다. 그러므로 *Master-NGLP* 모델이 결함 예측 성능 향상에 긍정적인 영향이 있음을 확인하였다.

5. 논의

5.1 NGLP 메트릭이 결함 예측 모델 성능에 긍정적인 영향을 끼치는 이유

NGLP 메트릭은 말뭉치를 이용하여 N-gram 모델을 학습시킨 후에 각 토큰 시퀀스가 가질 수 있는 확률을 모두 곱한 뒤 음의 로그를 취해 주어진 소스코드의 복잡도를 측정한다. 이때 NGLP 값이 크다는 것은 각 토큰 시퀀스가 가질 수 있는 확률이 작다는 의미다. 이는 정보이론 관점에서 확률이 작기 때문에 정보량이 커지게 되고 새로운 정보라고 여겨진다 [2]. NGLP 메트릭의 본질은 새로운 정보일수록 소스코드가 더 복잡할 것이라고 가정에 기반을 두고 있다. NGLP 메트릭이 결함 예측 모델의 성능을 향상시키는 것을 실증적인 연구를 통해 살펴보았듯이, 정보이론 관점에서의 새로운 정보가 복잡한 코드다 라는 가정이 잘 작동한다고 이해할 수 있다. 이런면에서 NGLP 메트릭이 소프트웨어의 복잡도를 잘 측정하여 결함 예측 모델의 성능을 향상시킬 수 있었다고 판단된다.

5.2 *Single-NGLP*와 *Master-NGLP* 모델의 예측성능 차이

N-gram 모델을 훈련시킬 때 데이터가 많을수록 결함 예측 성능 향상에 도움이 되는지 알아보기 위해 *Master-NGLP* 모델과 *Single-NGLP* 모델의 성능을 비교한다. 표 2의 네 번째 열과 세

번째 열은 *Master-NGLP* 모델과 *Single-NGLP* 모델의 평균 AUC 값을 각각 보여준다. 그 결과 8개의 모든 데이터셋에서 *Master-NGLP* 모델의 AUC가 *Single-NGLP* 모델의 AUC 보다 높다는 것을 관찰했다. 예를 들어, *Master-NGLP* 모델과 *Single-NGLP* 모델의 성능 차이는 0.003에서 0.027로 다양하다. *Master-NGLP* 모델의 8개 데이터셋의 평균 AUC 값은 0.532, *Single-NGLP* 모델은 0.505로 그 둘의 성능 차이의 평균은 0.008이다. 하지만 통계검증 후 어떠한 프로젝트의 p-value값도 0.05보다 작지 않고 모든 프로젝트들의 평균으로 통계검증한 경우에도 p-value가 0.574이므로 통계적으로 유의미하게 차이가 나지 않았다. 본 연구에서 Master NGLP 메트릭을 구하기 위해 8개의 프로젝트로 훈련한 master N-gram 모델을 이용하였는데, 더 많은 프로젝트 데이터를 이용한 N-gram모델을 이용할 경우 의미있는 성능 향상이 있을지 추가 검증이 필요하다.

6. 결론 및 향후연구

우리는 소프트웨어 메트릭을 발굴하기 위해 과거 Allamanis et al. [2]가 제안한 NGLP 메트릭이 결함 예측 성능의 성능 향상에 얼마나 영향을 미칠 수 있는지 조사하는 것을 목표로 하였다. 이를 위해 8개의 오픈소스 프로젝트를 대상으로 NGLP 메트릭을 사용한 것과 사용하지 않은 결함 예측 모델의 성능을 비교하는 실증적 연구를 수행하였다. 그 결과 모든 데이터셋에 대해서 NGLP를 메트릭으로 추가한 결함 예측 모델이 기존의 메트릭만을 사용했을 때 보다 결함 예측 성능을 향상시킨다는 것을 확인할 수 있었다. 향후 더 많은 프로젝트 및 다양한 머신러닝 알고리즘을 이용하여 동일한 결과를 얻을 수 있을지 추가 검증할 계획이다. 본 연구에서 NGLP가 유용한 메트릭임을 확인했기 때문에 향후 GitHub 프로젝트들의 커밋으로 master N-gram 모델을 만들어 NGLP 값을 구해주는 NGLP generator와 같은 오픈소스 서비스를 구현할 계획이다.

※ 본 연구는 2019 년도 정부(과학기술정보통신부)의 재원으로 한국연구 재단의 지원을 받아 수행된 연구임(No. 2018R1C1B6001919).

참고 문헌

[1] Y. Kamei *et al.*, "A large-scale empirical study of just-in-time quality assurance," in *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 757-773, June 2013.

[2] Allamanis, M & Sutton, C, Mining source code repositories at massive scale using language modeling. in Mining Software Repositories, 2013.

[3] J. Nam and S. Kim, "CLAMI: Defect Prediction on Unlabeled Datasets(T)," *2015 30th IEEE/ACM International Conference on Automated Software Engineering(ASE)*, Lincoln, NE, pp. 452-463, 2015.

[4] Thomas Shippey, David Bowes, Tracy Hall, Automatically identifying code features for software defect prediction: Using AST N-grams, Information and Software Technology, Volume 106, Pages 142-160, 2019.

[5] A. Bacchelli, M. D'Ambros, and M. Lanza, "Are popular classes more defect prone?" in Proceedings of the 13th International Conference on Fundamental Approaches to Software Engineering, ser. FASE'10. Berlin, Heidelberg: Springer-Verlag, pp. 59-73, 2010.

[6] C. Bird, N. Nagappan, B. Murphy, H. Gall, and P. Devanbu, "Don't touch my code!: Examining the effects of ownership on software quality," in Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, ser. ESEC/FSE '11. New York, NY, USA: ACM, pp. 4-14, 2011.

[7] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: a benchmark and an extensive comparison," Empirical Software Engineering, vol. 17, no. 4-5, pp. 531-577, 2012.

[8] A. E. Hassan, "Predicting faults using the complexity of code changes," in Proceedings of the 31st International Conference on Software Engi- neering, ser. ICSE '09, pp. 78-88, 2009.

[9] T. Lee, J. Nam, D. Han, S. Kim, and I. P. Hoh, "Micro interaction metrics for defect prediction," in Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, 2011.

[10] R. Moser, W. Pedrycz, and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction," in Proceedings of the 30th international conference on Software engineering, ser. ICSE '08, pp. 181-190, 2008.

[11] N. Nagappan and T. Ball, "Use of relative code churn measures to predict system defect density," in Proceedings of the 27th international conference on Software engineering, ser. ICSE '05, pp. 284-292, 2005.

[12] E. Giger, M. D'Ambros, M. Pinzger, and H. C. Gall, Method-level bug prediction, In Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 171-180, 2012.

[13] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings, IEEE Transactions on Software Engineering 34, 4(July 2008), 485-496, 2008.

[14] Foyzur Rahman, Daryl Posnett, and Premkumar Devanbu, Recalling the "Imprecision" of Cross-project Defect Prediction, In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering(FSE '12). ACM, New York, NY, USA, Article 61, 11 pages, 2012.

[15] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, A General Software Defect-Proneness Prediction Framework, IEEE Transactions on Software Engineering 37, 3(May 2011), 356-370, 2011.

비밀번호 복구를 위한 사진 문맥 데이터 기반 스마트폰 사용자 인증 접근법

송성한* 김순태 ○김태영
전북대학교
shak5643@gmail.com, stkim@jbnu.ac.kr, rlaxodud1200@gmail.com

Smartphone User Authentication Approach using Context based Data of Photograph for Password Recovery

SeounghanSong* SuntaeKim○ TaeyoungKim
Jeonbuk National University

요 약

아이디와 비밀번호를 이용한 사용자 인증방식은 사용 편리성과 인증시스템 구축 비용의 저렴함으로 여전히 사회 전반에서 널리 사용되고 있다. 하지만, 발견된 보안상 취약점으로 인해 사용자에게 더욱 어려운 비밀번호의 설정과 잦은 변경을 요구하고 있다. 그 결과 사용자들은 익숙하지 않은 비밀번호를 기억하지 못하는 경우가 흔히 발생하여 비밀번호를 복구하기 위한 절차로써 다시 지식기반 또는 소유기반의 인증수단을 사용하여 사용자를 인증한다. 이는 사용자에게 인증을 위해 기억해야 할 지식에 대한 기억유지 부담을 가중시켜 여러 서비스에 동일한 지식을 인증 수단으로 사용하도록 유도한다. 이러한 경우, 지식의 유출로 인한 특정 서비스의 인증 시스템이 붕괴되었을 때 다른 서비스의 시스템 또한 무력화될 가능성이 높다. 따라서, 본 논문은 사용자의 지식 및 경험을 기반으로 서비스 계정의 비밀번호 복구를 위한 사용자 인증을 진행하며 사용자의 지식 기억유지 부담을 줄일 수 있는 인증 방법으로써 사용자의 스마트폰의 사진을 이용한 문맥 데이터 기반 비밀번호 복구 접근법을 제시한다. 또한, User Study를 진행하여 제시하는 경험 기반의 사용자 인증 방법이 기존의 인증방법에 비하여 필요한 지식에 대한 기억유지 부담이 경감된다는 결과를 확인할 수 있다.

1. 서 론

현대 사회에서 제공되는 대다수의 서비스는 개인의 식별 및 특정 정보로의 접근제어를 위해 개인 인증을 실시하고 있다. 우리가 흔히 사용하는 오프라인 개인 인증 방식으로는 주민등록증이나 운전면허증을 제시하는 것이고, 온라인 개인 인증 방식으로는 특정 서비스의 계정을 만들 때 사용했던 아이디(ID)와 패스워드(Password)가 있다. 이러한 방법들은 한 개인이 개인 자신임을 확인시키는 수단으로, 일반적으로 이를 '인증'이라고 지칭한다.

2000년대에 이르러 스마트폰의 보급이 활성화된 후 현재 한국의 스마트폰 보급률은 91%에 도달하고 스마트폰의 사용량이 급격하게 증가함에 따라 스마트폰을 이용한 인증 방식 또한 발달하게 되었다.[1]

현재 모바일 환경에서 주로 사용되는 인증 방식들은 인증 과정에서 사용되는 수단에 따라 생체 기반, 소유 기반, 지식 기반의 인증의 3가지로 구분될 수 있다. 각 인증 기법은 사용성, 보안성, 구축비용의 측면에서 다음과 같은 특징을 가진다.

첫째로, 생체 기반의 인증방식은 개인의 고유 생체 정보를 인증 도구로 사용하여 높은 인식률과 높은 사용 편의성으로 확장성이 뛰어나다. 하지만 해당 생체 정보는 변경 불가능한 정보로써 외부에 노출되었을 때 그 피해가 막대하고 인증시스템의 구축 비용이 높은 단점이 있다.[2]

둘째로, 소유 기반의 인증방식은 사용자에게 공인된 인증기관에서 발급된 토큰과 같은 별도의 장비를 소지하게 하여 토큰 소지 사실을 기반으로 사용자를 인증하여 보안성을 높였다. 하지만 인증시스템의 구축이 어렵고 토큰의 발급 방식이 까다로우며, 인증을 위해서 물리적인 토큰을 별도로 소지하고 있어야 하는 조건이 있어서 사용 편의성이 낮다는 단점이 있다.[3]

세번째, 지식 기반의 인증방식은 사용자가 계정과 관련한 특정 지식을 알고 있는지 여부를 확인하여 사용자를 인증하는 방식이다. 이는 타 인증방식에 비해 구축비용이 저렴하고 사용하기 편리한 장점이 있어 널리 쓰인다. 하지만 패스워드 무작위 대입 공격 및 어깨너머 훔쳐보기 공격과 같은 계정 해킹 방법에 달리 대응이 없다.[4]

또한 현대의 지식기반 인증방식에서는 사용되는 지식의 보안을 높이기 위해 점점 더 어려운 지식을 사용자에게 기억하도록 요구하고 있다. 하지만 이로 인해 해당 지식을 일정 기간 후에 사용자가 기억하지 못하는 문제가 흔히 발생하고 있다. 이러한 이유로 사용자들은 동일한 지식을 여러 시스템의 인증에 사용하게 되었고 이는 한 시스템의 지식이 노출되었을 때 타 시스템의 보안이 무력화되는

문제점을 야기하여 타 인증 방식에 비해 보안강도가 가장 낮다고 평가된다.[5]

그럼에도 불구하고 아직까지 현대 사회에서는 사용자의 사용 편리성이 높고 구축비용이 저렴한 이유로 지식기반 인증방식을 최우선 인증방식으로 채택하고 있다. 따라서 기본 지식기반 인증의 사용 편의성 및 보안성 등의 장점은 유지하되 지식의 기억유지 어려움 및 지식 노출로 인한 보안 취약점 발생 문제를 해결해야 할 필요가 있다.

이에 따라 다음과 같은 Research Question을 정의한다.

1. 기존 지식기반 인증방식에 비하여 인증에 사용되는 지식에 대한 기억유지 부담이 경감되는가?

본 연구는 안전한 사용자 인증 방법으로써 사용자의 스마트폰으로 촬영된 사진의 문맥 데이터 기반 사용자 인증 접근법을 소개하고 계정을 복구하기 위해 주로 사용되는 질문/답을 이용한 지식 기반의 인증 기법과의 기억유지 부담에 대한 사용자 비교를 목표로 한다. 본 연구에서는 사용자가 모바일 환경에서 촬영한 사진의 위치를 나타내는 GPS 데이터, 촬영된 시간을 나타내는 시간 데이터, 사진으로부터 추출한 사람의 얼굴 데이터 등을 사용한다.

연구에 사용된 사진 데이터는 사용자가 직접 자신의 휴대폰으로 촬영한 사진으로 제한하고, 그 중 GPS 및 촬영 시간 데이터를 포함한 사진과 그렇지 않은 사진을 구분하기 위해 별도의 전처리를 과정을 거친다. 또한, 사진으로부터 사람의 얼굴을 식별하고 추출하기 위해 OpenCV 오픈소스 라이브러리를 사용한다.

본 논문은 다음과 같은 순서로 진행된다. 다음의 2장에서는 제안하는 사용자 사진 기반의 스마트폰 사용자 인증 기법에 대한 개요와 인증 진행 방식을 설명한다. 3장에서는 본 연구에서 실시한 실험 환경(실험 대상, 방식) 및 실험 결과에 대한 평가를 진행한다. 4장에서 관련연구를 소개하고 5장에서 결론을 내린다.

2. 사진 문맥 데이터를 이용한 사용자 인증 접근법

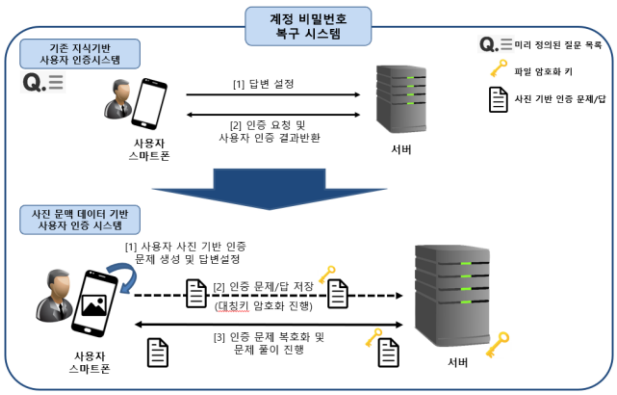


그림 1. 사진 문맥 데이터 기반 사용자 인증 접근법

먼저 위의 그림 1과 같이 새로운 비밀번호 복구 기법의 경우 기존 계정 비밀번호 복구를 위한 사용자 인증방식에서 흔히 사용되는 고정적 질문과는 달리 사용자가 직접 촬영한 사진의 위치 정보와 촬영 시점 정보 데이터 그리고 사진에 존재하는 사람 얼굴 데이터를 추출하여 사용자마다 다른 질문을 생성한다. 질문 생성 프로토타입은 웹 기반으로 구현되어 사용자는 아래의 과정을 통해 사용자 인증에 사용할 질문을 선택할 수 있다. 이 과정은 그림 2와 같다. 하지만 아래의 그림에서 보여지는 선택지 중 사람이 아닌 다른 물체 또한 인식된 사실을 알 수 있다. 이는 사용된 OpenCV 오픈소스 기반 얼굴인식 모듈의 성능 문제로써 본 논문에서는 문제삼지 않는다. 하지만 이에 따른 사용자 불편을 최소화하기 위해 사람이 인식되지 않은 사진에 대해서 건너뛰기 버튼을 통해 다른 사진을 선택할 수 있도록 한다.

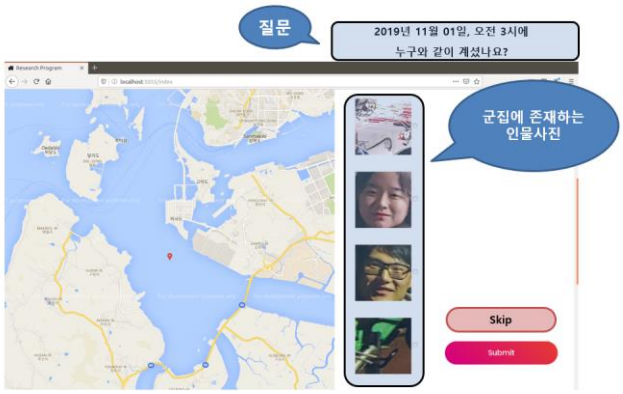


그림 2. 사용자 인증 질문 확인 화면

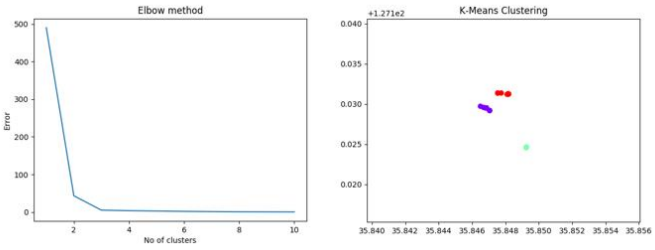


그림 3. 위치 기반 사용자 사진 군집화

첫째로 위의 그림 3과 같이 사진의 위치 정보에 따른 군집화를 진행하여 사용자가 어느 위치에 얼마만큼의 정량적 빈도로 방문했는지를 파악한다.

본 논문은 방문 빈도수가 상대적으로 적은 위치의 군집에 포함된 사진일수록 해당 사용자만의 독특한 경험일 것이라고 가정하기 때문에 사용자를 구별할 수 있는 문제의 생성을 위해 사진의 위치 정보에 따른 군집화를 진행한다.

또한 사진에서 사람으로 인식되는 이미지를 추출하여 인물별 군집화를 진행하고 위치 기반으로 분류된 군집에 어떤 인물이 포함되는지를 매핑한다.

이는 사용자 인증 과정에서 정답 외 선택지 생성 시 사용자가 헷갈릴 수 있는 선택지를 배제하기 위함이다.

이를 통해 사용자는 질문에서 사진이 촬영된 위치, 시간, 동행한 사람의 얼굴 등의 데이터를 동시에 확인이 가능하여 경험에 대한 연상이 더욱 쉽게 일어날 수 있다. 이 때, 동일한 경험이 없는 대상은 문제를 보게 되었을 때 문제에 대한 답을 쉽게 하지 못할 것이다. 시각적인 개인적 경험은 쉽게 외부로 유출 및 전수될 수 없기 때문이다.

둘째로, 생성된 질문과 답의 정보는 서버가 보관하는 키를 이용해 대칭 암호화 진행 후 서버에 저장하여 관리한다. 이를 통해 위의 비밀번호 복구 접근법에서 사용되는 사용자 인증 수단인 사진과 그로부터 생성된 질문의 유출을 막을 수 있다.

셋째로, 서버가 사용자로부터 인증 요청을 받아 문제를 반환하여 사용자 인증을 진행할 때 그림 4와 같이 사용자에게 서버의 키로 복호화 된 문제를 풀게 하여 인증을 진행한다. 이 과정에서 여러 인증 단계를 거쳐 한 장소에서 동일한 경험을 가진 타인의 인증 시도록 인한 계정 비밀번호 탈취 가능성을 낮출 수 있다.

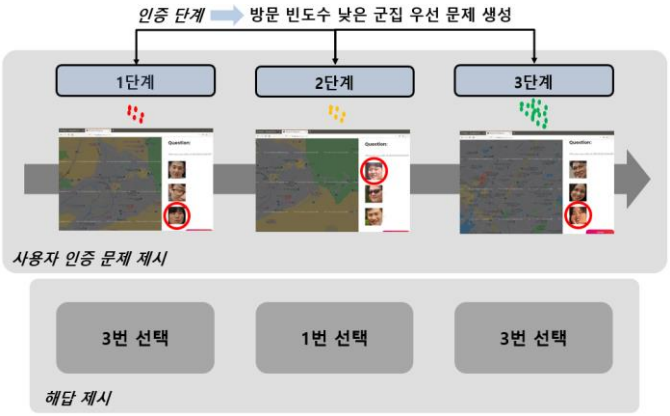


그림 4. 사진 문맥 데이터 기반 사용자 비밀번호 복구 접근법의 사용자 인증 진행 방식

3. 실험 및 평가

3.1 실험 환경

본 논문에서 제안하는 사용자 사진의 문맥 데이터 기반 비밀번호 복구 접근법에 대한 효용성 실험을 진행하기 위해 총 6명의 참가자를 모집하였다. 아래의 표 1과 표 2에 나타난 것과 같이 참가자의 연령대가 20대와 50대에 주로 분포하고, 평균적으로 2017년부터 2019년 말 사이에 촬영된 총 3년의 사진 데이터를 제공하였다.

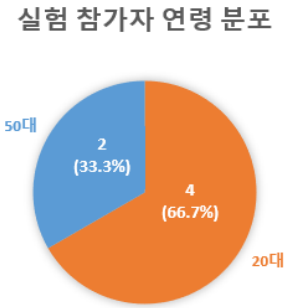


표 1. 실험 참가자 연령 분포

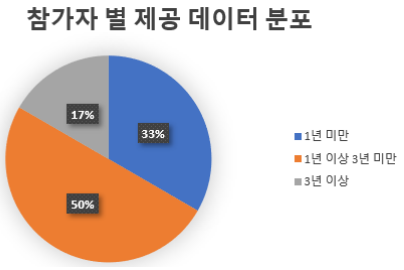


표 2. 참가자 별 제공 데이터 분포

참가자들은 자신의 스마트폰으로 촬영된 사진을 시스템에 제공하고 웹 기반으로 구현된 인증 문제 생성 프로그램을 통해 총 3개의 문제를 선택하였다. 또한, 선택된 문제 정보는 서버에 암호화된 형태로 저장하여 관리한 후 1주의 시간이 흐른 뒤 참가자들에게 앞의 그림 3과 같이 문제를 풀게 하였다.

또한 사용자 자신은 아니지만 국지적으로 동일한 경험을 가지는 타인이 인증을 진행하였을 때 인증 성공 확률을 조사하기 위해 서로의 활동 반경이 자주 겹치는 3명의 참가자들에게 서로의 문제를 풀어보도록 하는 교차 문제 풀이를 진행하였다.

3.2 실험 결과 및 평가

	사용자	경험 공유자	타인
인증1단계(%)	83.33	0	0
인증2단계(%)	100.00	0	0
인증3단계(%)	100.00	50.00	0
인증 결과	O	X	X

표 3. 인증 요청자에 따른 인증 단계별 인증 성공률

사용자가 인증을 위한 문제를 선택한 후 1주가 지나고 다시 문제를 풀어보도록 하였을 때, 위의 표 3과 같이 6명의 참가자는 인증 단계별 평균 94.44%의 확률로 해당 문제를 풀 수 있었다.

또한 동일한 경험을 가진 참가자들 사이에 진행된 교차 문제 풀이 결과 총 3단계의 인증 단계 중 동일한 경험을 나타내는 문제를 제외한 인증 문제의 풀이에 실패하였다. 이는 국지적으로 경험이 겹치더라도 인증에 성공할 확률이 낮다는 것을 의미한다.

또한 아래의 표 4는 사진 맥락 데이터 기반 비밀번호 복구 접근법에 사용되는 사용자 인증 방법이 기존의 지식기반 사용자 인증 방법에 비하여 기억유지 노력을 경감시켰다는 사용자 인식 조사 결과를 나타낸다. 본 실험은 1주간의 짧은 연구 기간을 가지고 진행된 실험으로 장기적인 기억유지 노력에 대한 결과는 포함하고 있지 않다. 하지만 짧은 실험 기간에도 불구하고 대부분의 실험 참가자들로부터 기존의 인증 시스템에 비하여 지식 기억유지에 대한 노력이

경감되었다는 답변을 얻을 수 있었다.

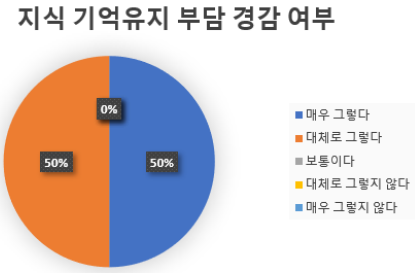


표 4. 지식 기억유지 부담 경감 여부

4. 관련 연구

4.1 이미지 기반 사용자 인증 방법

먼저 시각적인 이미지를 기반으로 사용자 인증을 진행하는 방법에 대한 연구가 진행되었다.[6][7] 그 중 이미지 인지 능력을 이용한 사용자 인증 연구에서는 사용자에게 스마트폰 인증에 사용될 이미지(패스 이미지)를 선택하게 한 후, 인증 시에 여러 다른 사진을 나열한 뒤 패스 이미지의 선택 여부를 통해 사용자를 인증한다. 진행 방식은 아래의 그림 5과 같다.

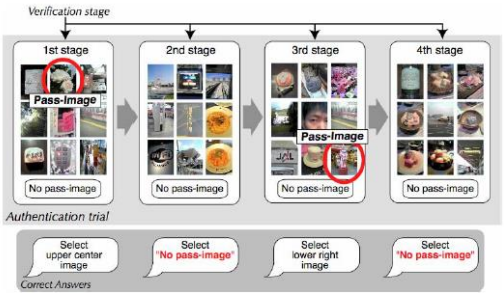


그림 5. 인지 기반 이미지 사용자 인증 방법

이러한 접근 방법은 최대 16주간 인증에 사용된 사진을 기억할 수 있다는 결과를 나타낸다.[7]

하지만 이미지 저장소에서 관리되는 패스 이미지에 대한 암호화 처리가 되지 않아 해당 이미지 저장소의 공격으로 인한 패스 이미지의 탈취가 가능하며 어깨너머 훔쳐보기 공격에 대응을 위한 패스 이미지 변경 후, 이전에 사용되던 이미지의 영향으로 인해 변경된 패스 이미지를 선택하지 못할 가능성이 있다. 이를 통해 인증을 위한 패스 이미지를 변경한 횟수가 많아질수록 새로운 이미지를 기억하기 위한 부담이 가중될 것이라고 예상된다.

4.2 행동 기반 사용자 인증 방법

서론에서 언급한 것과 같이 기존의 지식기반 인증방법에서의 지식 기억은 사용자에게 큰 부담으로 다가올 수 있다. 이를 해결하기 위해 사용자 일상 행동에서 패턴을 분석하고 추출하여 타인과 구분하려는 연구가 진행되고 있다.[8] 하지만 타인과 개인의 구분을 위해 제공되어야 할 데이터의 종류와 양이 많다. 또한 패턴 인식을 통해 생성된 확률을 통해 정상 사용자와 비정상 사용자를 구분하기 때문에 타 인증 방법에 비해 사용자 인증 정확도가 높지 않다.

5. 결 론

본 논문은 계정의 복구에 사용되는 지식기반 사용자 인증 방법에서 사용자에게 가중되는 지식의 기억유지 부담을 해결하기위해 사진 문맥 데이터 기반의 사용자 인증 접근법을 제시하였다. 또한 제시하는 접근법의 타당성을 입증하기 위해 User Study를 진행하여 기존 인증 방식의 보안성은 유지한 채 사용자의 지식의 기억유지 부담을 경감시킬 수 있다는 연구 결과를 얻었다. 이러한 방식이 실제 사용되기 위해서 사용자의 사진에 GPS 위치 데이터, 시간 데이터가 필수적으로 존재해야 한다는 조건이 있지만 그럼에도 불구하고 실험 참가자들로부터 추후 사용 필요성에 관한 긍정적인 답변을 얻을 수 있었다.

6. Acknowledgment

본 논문은 정부(정보통신기술진흥센터)의 재원으로 대학 ICT 연구센터 육성지원 사업의 지원을 받아 수행된 연구임 (No.IITP-2019-2017-0-01628)

7. 참고 문헌

[1] Laura silver, “Smartphone Ownership Is Growing Rapidly Around the World, but Not Always Equally”, Pew Research Center, Feb.2019.

[2] Yongjin Wang, “Face Based Biometric Authentication with Changeable and Privacy Preservable Templates”, Biometrics Symposium, 2007

[3] Dhanashri Ghosalkar, Shweta Patil, “OTP over SMS: Time Delay Issues and Causes”, IJIRCCE, Vol. 7, Issue 1, January 2019.

[4] Herley C., “So long, and no thanks for the externalities: the rational rejection of security advice by users”, In Proc. Of SACMAT, 2009.

[5] Lamport, Leslie. “Password authentication with insecure communication.” Communications of the ACM 24.11: 770-772. 1981

[6] Dhamija, Rachna, and Adrian Perrig. “Déjà vu-A User Study: Using Images for Authentication.” USENIX Security Symposium. Vol. 9. 2000.

[7] Takada, Tetsuji, Takehito Onuki, and Hideki Koike, “Awase-e: Recognition-based image authentication scheme using users’ personal photographs.” In 2006 Innovations in Information Technology, pp. 1-5. IEEE, 2006

[8] Eiji H., Saivik Das, Shahriyar A., Jason Hong and Ian Oakley, “CASA:Context-Aware Scalable Authentication”, SOUP, 2013.

개인지원 로봇의 머신러닝 기능을 위한 리스크 감소 프로세스

이연재 ○, 한혁수
상명대학교 컴퓨터과학과
yjflore34@gmail.com, hshan@smu.ac.kr

A Risk Reduction Process for Machine Learning Functions of Personal Care Robot

Yeonjae Lee ○, Hyuksoo Han
Dept. of Computer Science, Sangmyung University

요 약

개인지원 로봇 (Personal Care Robot)은 사람과의 물리적 접촉과 상호작용이 필수적이기 때문에, 사고가 나면 부상 등 인적 피해를 가져올 수 있는 안전 필수 시스템이다. 개인지원 로봇의 안전성 확보를 위해 개발된 ISO 13482는 로봇 개발에서 준수해야할 해저드 분석과 리스크 완화 방안을 위한 기본 틀을 제시하고 있다. 특히, 해저드에 대한 대처 방안으로는, ISO 12100의 리스크 감소 3단계인 설계상의 대책 (Inherently Safe Design), 안전장치 적용 (Protective Measure), 사용 정보 (Information for Use)를 기준으로 안전지침을 명세하고 있다. 하지만, 최근 각 분야에서 적용되고 있는 머신러닝 기법에 해당하는 부분은 자율 의사결정 지침에서 상위 수준의 가이드만 제시하고 있어 실무자들이 적용하기에 어려움이 있다. 본 논문에서는 ISO 13482의 안전지침 구성을 기반으로, 개인지원 로봇의 머신러닝 기능에 대한 해저드 분석과 리스크 감소 프로세스를 제안한다. 그리고 머신러닝 기법 중 로봇의 특성에 적합하다고 생각되는 강화 학습을 기반으로 식당의 서비스 로봇 안전 기능에 적용해 보았다.

1. 서론

안전 필수 시스템(Safety-Critical System) 내 전기, 전자 및 소프트웨어의 비중이 높아지면서, 제어 소프트웨어의 결함 및 시스템 컴포넌트들 간의 상호작용으로 인한 사고의 위험이 커지고 있다[1]. 이러한 사고를 예방하기 위해, 각 산업계에서는 기능 안전(Functional Safety)의 메타 모델인 IEC 61508을 기반으로 각 분야의 특성에 맞는 기능 안전 표준을 제정하였다[2].

이동형 도우미 로봇(Mobile Servant Robot), 신체보조로봇(Physical Assistant Robot), 탑승형 로봇(Person Carrier Robot)등을 포함하는 개인지원 로봇(Personal Care Robot)은 사람과의 물리적 상호작용이 주된 인터페이스이기 때문에, 사소한 오작동도 사람에게 큰 피해를 가져올 수 있는 안전 필수 시스템이다. 그러므로, 개인지원 로봇의 개발에 있어서도 철저한 안전성 확보 노력이 필요하며, 이를 위해 개인지원 로봇의 안전성 표준으로 ISO 13482가 개발되었다. 그림 1은 ISO 13482의 안전 활동들을 보여준다[3].

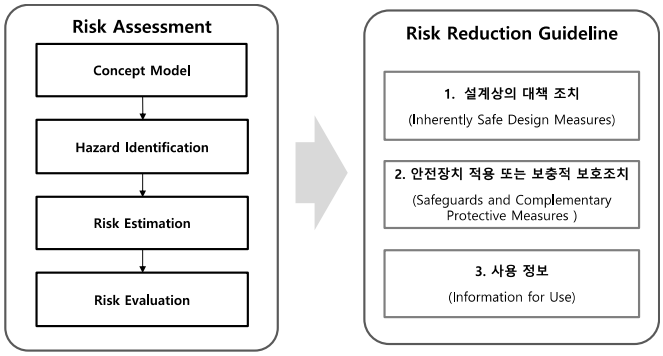


그림 1. ISO 13482의 안전 활동

안전 활동의 중심에는 해저드 분석 활동과 리스크 평가 활동 그리고 안전 요구사항 도출 활동이 있다.

ISO 13482는 각 해저드에 대한 대처 방안으로 리스크 감소 3단계를 기준으로 각 단계의 안전 지침을 명시하고 있다. 리스크 감소를 위한 3단계는 다음과 같다[3].

(1) 설계상의 대책(Inherently Safe Design)

리스크 감소의 가장 중요한 첫번째 조치이다. 설계를 할 때 안전을 고려해서 본질적으로 해저드를 제거 또는 완화시키는 것이다. 예를 들면, 충격이 덜 가는 소재를

쓰고, 피부에 닿아도 괜찮은 외장을 사용하는 것 등이 있다.

(2) 안전장치 적용(Protective Measure)

설계상의 조치만으로 해결할 수 없는 경우들에 대한 방안을 제공한다. 예를 들어, 주변 장애물로 인한 해저드는 설계만으로 대책을 세울 수 없다. 주로, 제어 기능을 활용해야 하는 경우가 많다.

(3) 사용 정보(Information for Use)

설계와 제어기능으로도 해결되지 않은 해저드에 대해서는 사용자에게 안전 관련 정보를 제공한다.

최근 머신러닝은 다양한 분야에 적용되고 있으며, 눈에 띄는 성공 사례들이 나오고 있다[4]. 로봇 분야에서도 자율성 부분에 머신러닝 기법이 적용되고 있다[5,6]. 이에 따라, 머신러닝 기반 시스템의 안전성에 관한 우려가 증가하고 있으며, 안전성 확보를 위한 많은 연구들이 진행되고 있다[7,8].

개인지원 로봇과 같은 안전 필수 시스템들은 허용 가능하지 않은 리스크에 대한 잠재적 해저드를 파악하고, 이들을 제거하거나 완화하기 위한 대책을 마련해야한다. 하지만, 개인지원 로봇의 안전 요구사항 표준인 ISO 13482는 머신러닝 기법에 해당하는 부분에 대한 자율적 의사결정 지침에서 상위 수준의 가이드만 제시하고 있어 실무자들이 적용하기에 어려움이 있다.

본 논문에서는 ISO 13482의 안전 지침 구성을 기반으로, 개인지원 로봇의 머신러닝 기능에 대한 해저드 분석과 리스크 감소 프로세스를 제안한다. 제안된 프로세스의 타당성을 분석하기 위해 머신러닝 기법 중 강화 학습을 기반으로 식당 서비스 로봇의 자율기능에 적용해 보았다.

2. 관련 연구

2.1 머신러닝(Machine Learning)

머신러닝(Machine Learning)은 명시적인 프로그래밍 없이 컴퓨터가 학습할 수 있도록 하는 기법이다. 머신러닝 기법의 종류에는 학습에 사용되는 데이터의 형태와 해결하려는 문제에 따라 지도 학습(Supervised Learning), 비지도 학습(Unsupervised Learning), 강화 학습 (Reinforcement Learning) 등이 있다.

이 중 강화 학습은 지능을 갖춘 에이전트(Agent)가 주어진 상황에서 최적의 행동을 선택하기 위한 일련의 순서가 있는 의사결정 문제를 다루기 때문에 자율 주행, 로봇, 게임 등 자율(Autonomous) 시스템에 폭 넓게 활용되고 있다. 다른 머신러닝 기법과 달리, 그림 2와 같이 명시적인 학습 데이터없이 에이전트가 환경(Environment)과 반복적인 상호작용을 통해 데이터를 수집한다. 에이전트는 센서 등을 통해 관측(Observation)한 자기 자신 및 환경의 상태(State)와 정책(Policy)을 기반으로 다음에 어떤 행동을 수행할지 결정하며, 이에 따른 보상(Reward)을 받는다. 이에 따라, 에이전트는 환경과의 상호작용에 따른 보상의 기대치를 최대화하여 목표를 달성하기 위한 최적의 정책을 학습한다[5].

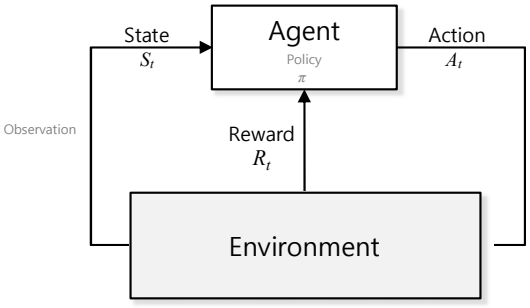


그림 2. 강화 학습의 기본 개념도

2.2 ISO 13482의 자율 활동(Autonomous Action) 안전지침

ISO 13482에서는 5.12절에서 자율 활동에 대한 지침을 제공하고 있다. 자율적 의사결정(Autonomous Decision)과 활동(Action)을 수행하는 로봇은 잘못된 결정(Wrong Decision)과 부정확한 행동(Incorrect Action)이 허용가능하지 않은 리스크(Unacceptable Risk)를 야기하지 않도록 해야한다고 규정한다. 이러한 해저드에 대한 대처방안으로 다음과 같이 3단계 리스크 감소 방안을 제시하고 있다.

(1) 설계상의 대책(Inherently Safe Design)

- 부정확한 활동으로 인한 리스크를 줄이기 위해 운용 시나리오(Operational Scenario)를 규제한다.
- 안전성과 관련된 객체, 이동경로등에 대해 고유식별자를 사용한다.

(2) 안전장치 적용(Protective Measure)

- 결과가 불확실성이 높은 의사 결정은 대안 방식을 기반으로 재평가 되어야 하며, 외부의 도움을 찾아야 하고, 보호 멈춤 장치가 시작되어야 한다.
- 위험한 상황(Hazardous Situation)을 야기 할 수 있는 의사결정들에 대해서는 타당성 체크를 수행해야 한다.

(3) 사용 정보(Information for Use)

- 사용 정보를 통해, 로봇이 보유한 센싱(Sensing)과 의사결정 능력에 대해, 잘못된 행동과 의사결정으로 인한 재난을 피할 수 있도록 지침을 제공해야 한다.

2.3 기존 연구 분석

머신러닝은 많은 분야에서 효과를 나타냄에 따라 적용 분야가 넓어지고 있지만, 머신러닝이 적용된 시스템의 안전성에 대한 염려도 커지고 있다[5,9]. Russel과 Dewey는 자율형 시스템의 활용이 늘어갈수록 시스템의 견고성과 안전성이 중요해진다는 것을 강조하면서, 검증(Verification), 확인(Validation), 보안(Security), 제어(Control) 등 4가지 영역에서의

대처방안을 제안했다[4]. Wesel 등은 NASA 보고서에서 머신러닝 알고리즘이 매우 복잡하고, 검증하기가 어렵다는 것을 주장하고, 검증 분야에서 해결해야할 연구 주제들을 제안하였다[9]. Faria는 머신러닝 기반의 자율기능의 안전성을 확보하기 위한 연구들을 조사하고, STAMP의 Control Structure를 기반으로 제어기의 프로세스 모델은 Markov Decision Process Model로, 제어 알고리즘은 에이전트가 택하는 정책(Policy)에 맵핑시키는 방식으로 자율 제어 기능의 잠재적 문제점을 지적했다. 프로세스 모델과 실제 상태와의 불일치로 인한 문제점은 머신러닝 에이전트 모델 불일치로, 알고리즘 문제는 부적절한 정책으로 유사성을 부여했다[9]. Pecka와 Svoboda는 자율 로봇의 안전 행위에 대한 연구를 수행했는데, 특히 알려지지 않은 상태로 Exploration을 수행할 때에도 안전함을 유지하면서도, Exploration의 장점을 살릴 수 있는 프레임워크를 제안하였다[5].

3. 개인지원 로봇의 머신러닝 기능을 위한 리스크 감소 프로세스

3.1 제안 프로세스

본 논문에서는 ISO 13482를 기반으로 머신러닝 기법을 활용하는 자율기능의 리스크 감소 프로세스를 제안한다. 이는 그림 3과 같이 3단계로 구성되어 있다.

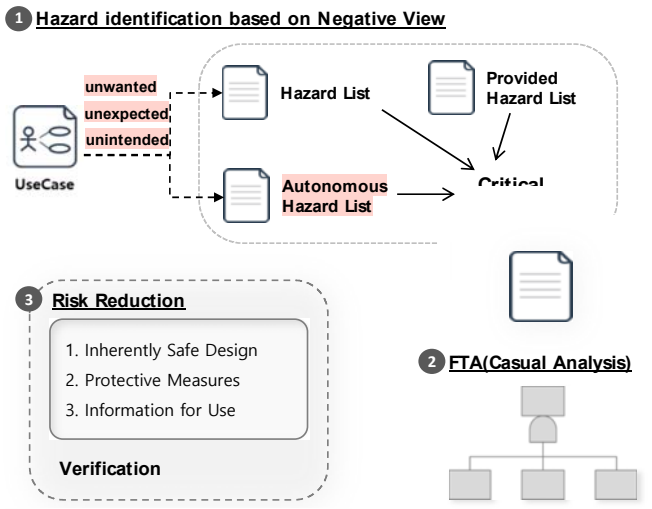


그림 3. 머신러닝 기능을 위한 리스크 감소 프로세스

3.1 Negative View 기반의 해저드 분석

이 단계의 목표는 자율기능의 잠재적인 해저드를 다양한 관점에서 빠짐없이 식별하는 것이다. SysML의 다이어그램들을 활용한 모델링이 끝나고 나면, 각 다이어그램이 제공하는 관점에 따라 비정상적이고 부정적인 상황들을 해저드로 추출한다.

본 논문에서는 자율기능의 특징을 반영하여 <Unwanted>, <Unexpected>, <Unintended>를 가이드 워드로 도출하였다.

3.2 FTA 기반의 원인 분석

이 단계에서는 FTA 기법을 활용하여 치명적인 해저드들의 원인을 도출한다. FTA 기법은 시스템에 문제를 발생시키는 특정 해저드에 대해 예상 가능한 근본 원인을 식별하고, 그에 대한 논리적 구조를 규명하는 분석 기법이다[1].

3.3 리스크 감소방안을 기반으로 대처방안 도출

각 해저드에 대한 대응책을 마련하기 위해 ISO 13482가 제시한 리스크 감소 체계를 기반으로 가이드라인을 제시한다.

(1) 설계상의 대책 조치(Inherently Safe Design)

잘못된 학습으로 인하여 야기된 부정확한 활동으로 인한 리스크를 줄이기 위해 환경 및 운용 조건을 보수적으로 규제한다.

(2) 안전장치 적용(Protective Measure)

머신러닝 기법에 따라 그 특성을 반영하여 지침을 제공해야 한다. 예를 들어, 지도 학습과 비지도 학습의 경우에는 데이터의 중요성을 강조하고, 강화 학습의 경우 보상(Reward)과 정책(Policy)에 중점을 둔다.

(3) 사용정보(Information for Use)

자율의 한계에 대한 정보, 이슈 발생시 대처 방안을 제공한다. 예를 들어, 로봇의 정상 작동을 위해 피해야할 환경 조건 등을 설명하고, 위급 상황에서 로봇을 제어할 수 있는 방안을 설명한다.

4. 사례 적용 : 식당의 서비스 로봇

본 논문에서 제안한 프로세스의 효용성을 확인하기 위해, 식당의 서비스 로봇의 주요 기능에 적용해 보았다.

식당 서비스 로봇은 매장 내 테이블 배치, 고객과 직원들의 동선 등을 고려해 음식을 서빙 할 수 있는 자율 주행 서비스 로봇이다. 적용 예제의 요구사항을 기반으로 제안 프로세스에 따라, UseCase 다이어그램을 작성하고, 주요 기능 중 자율적 의사결정이 포함된 “UC06:배터리 충전” UseCase에 대해 자율기능의 특징을 반영한 가이드워드를 적용하여 표 1과 같은 해저드 리스트를 도출하였다.

표 1. 배터리 충전 기능의 해저드 리스트 예

UC06 : 배터리 잔량이 일정 수준 이하일 경우, 충전기로 이동한다.	
Guide Word	Hazard
Unwanted Unexpected Unintended	H1. 배터리 잔량이 일정 수준 이하이나, 이동하지 않음
	H2. 충전기로 이동하려 하나, 목적지를 찾지 못함
	H3. 이동 중 의도치 않게 멈춤
	...

그림 4는 도출된 해저드 중 치명적이라고 판단된 해저드에 대해 FTA를 활용하여 원인 분석을 실시한 결과를 나타낸다.

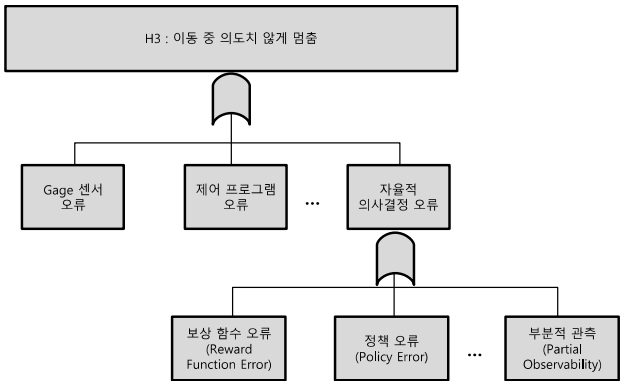


그림 4. 해저드 H3의 원인 분석 결과

파악된 원인들을 기반으로 자율적 의사결정 오류에 대한 리스크 감소 방안 일부를 그림 5와 같이 도출하였다.

UC06 : 배터리 잔량이 일정 수준 이하일 경우, 충전기로 이동한다.			
Hazard	Cause	Measure	Verification
H3. 이동 중 의도치 않게 멈춤	자율적 의사결정 (Autonomous Decision) 오류	(1) 설계상의 대책(Inherently Safe Design) - 중복 센서를 사용해야한다. - 안전성에 위협이 되는 환경 및 운용 조건을 규제해야한다. ... (2) 안전장치 적용(Protective Measure) - SL_M1 학습데이터가 편향적이지 않아야한다. - RL_M1 보상에 대한 사전 점검을 수행해야한다. - RL_M2 정책에 대한 사전 점검을 수행해야한다. - RL_M3 Exploit & Explore trade-off 점검을 수행해야한다. - RL_M4 Fail-Safe 기능을 제공해야한다. ... (3) 사용정보(Information for Use) - 로봇의 기능 중 학습을 통한 자율 기능을 사용자에게 제공해야한다. - 각 자율 기능의 의도된 정상적인 행위들에 관한 정보를 제공해야한다. ...	F. 소프트웨어 검사 (Examination of Software) - 정형검증 (Formal Verification)
* SL : Superivsed Learning, RL : Reinforcement Learning			

그림 5. 자율적 의사결정 오류에 대한 리스크 감소 방안 예

5. 결론

안전 필수 시스템인 개인지원 로봇 (Personal Care Robot)의 안전성 확보를 위해 개발된 ISO 13482는 로봇 개발에서 준수해야할 해저드 분석과 리스크 완화 방안을 위한 기본 틀을 제시하고 있다.

하지만, 머신러닝 기법에 해당하는 자율적 의사결정과 활동에 관한 부분은 상위 수준의 가이드만 제시하고 있어 실무자들이 적용하기 위해서는 머신러닝 기법이 반영된 프로세스와 지침이 요구된다.

본 논문에서는 ISO 13482의 메커니즘을 기반으로, 개인지원 로봇의 머신러닝 기능에 대한 해저드 분석과 리스크 감소 프로세스를 제안한다.

제안한 프로세스의 장점은 다음과 같다.

첫째, SysML 다이어그램 중 UseCase를 기반으로 명세하기 때문에, 개발자들과 안전담당자들 간에 원활한 소통을 지원한다.

둘째, 자율기능에 적합하다고 판단되는 가이드워드인 Unwanted, Unexpected, Unintended를 활용하여 해저드를 도출하는 방식을 제공하고, 기존 해저드 분석 방법과의 통합 체계를 제공하였다.

셋째, ISO 13482 국제 표준에 부합하는 해저드 식별 방법 및 리스크 감소 체계를 구축하였다.

향후에는 지도학습(Supervised Learning)과 비지도학습(Unsupervised Learning) 그리고 추천시스템(Recommendation System) 등에 대한 해저드 분석 기법들을 연구하고, 이를 기반으로 안전 요구사항들을 도출하고자 한다.

6. 참고 문헌

[1] Leveson, N., "A new accident model for engineering safer systems," Safety Science, Vol.42, No.4, pp. 237-270, 2004.

[2] IEC 61508 : Functional Safety of Electrical/ Electronic/Programmable Electronic Safety- related Systems, 2nd Ed., 2010.

[3] ISO, 13482 Robots and robotic devices — Safety requirements for personal care robots, 2014.

[4] Russell, Stuart, Daniel Dewey, and Max Tegmark. "Research priorities for robust and beneficial artificial intelligence." Ai Magazine 36.4, 105-114, 2015.

[5] Pecka, Martin, and Tomas Svoboda. "Safe exploration techniques for reinforcement learning–an overview." International Workshop on Modelling and Simulation for Autonomous Systems. Springer, Cham, 2014.

[6] ICRC, "Autonomy, artificial intelligence and robotics: Technical aspects of human control." ICRC(International Committee of the Red Cross), 2019.

[7] Zhang, Du, and Jeffrey JP Tsai. "Machine learning and software engineering." Software Quality Journal 11.2, 87-119, 2003.

[8] Faria, José M. "Machine learning safety: An overview." Proceedings of the 26th Safety-Critical Systems Symposium, 2018.

[9] Van Wesel, Perry, and Alwyn E. Goodloe. "Challenges in the verification of reinforcement learning algorithms." 2017.

자동화된 소프트웨어 가변성 추출을 위한 정의/사용 정보 기반 제품군 공유 코드 정렬

김태영, 이지현
전북대학교 소프트웨어공학과
wareengineer@gmail.com, jihyun30@jbnu.ac.kr

Shared Source Code Arrangement of a Software Family based on Define/Use Information for Automated Software Variability Extraction

Taeyoung Kim, Jihyun Lee
Department of Software Engineering, Jeonbuk National University

요 약

본 연구에서는 클론-앤-오운(clone-and-own) 방식으로 개발된 유사 소프트웨어 제품들을 소프트웨어 프로덕트 라인 개발 체계로 마이그레이션하고자 할 때, 소스 코드로부터 소프트웨어 가변성을 효과적으로 추출하기 위한 전처리 기법으로써 소스 코드 정렬 방법을 제안한다. 코드 정렬은 특정 변수를 정의하는 문장과 해당 변수가 사용되는 문장을 가능한 가까이 연속적으로 배치시킴으로써 향후 가변성을 식별하는 과정에서 가능한 큰 단위로 가변성이 식별될 수 있도록 만드는 과정이다. 평가를 위해 클론-앤-오운 방법으로 개발된 오픈소스 게임인 ApoGames에 코드 정렬 방법을 적용하였으며, 평가 결과 제안한 방법이 일부 상황에서 가변 소스 코드의 그룹핑에 긍정적으로 작용함을 확인했다.

1. 서 론

클론-앤-오운(clone-and-own) 개발 방법은 기존 소프트웨어 제품과 유사한 새로운 소프트웨어 제품 개발 시 레거시 소프트웨어 제품의 코드를 복제(copy)하고 수정(modify)하여 새로운 소프트웨어 제품을 개발하는 방법이다[1]. 클론-앤-오운 방법은 빠르게 새로운 소프트웨어 제품을 개발할 수 있게 해주지만 클론한 소프트웨어 제품들 간의 관계 정보가 없기 때문에 재사용한 코드 부분의 결함 등으로 인한 수정, 변경이 필요할 때 유지보수에 심각한 어려움이 있다[2, 3].

추출식 SPL(Extractive Software Product Line) 접근법은 기존의 유사한 소프트웨어 제품들로부터 공통된 부분과 상이한 부분을 추출하여 제품 개발 방식을 SPL로 마이그레이션하는 방법이다. 이 방법은 유사한 소프트웨어 제품들의 소스코드, 설계, 분석 등의 자산으로부터 공통성과 가변성 정보를 추출하여 제품라인 자산을 구축하고 이로부터 각 소프트웨어 제품군의 멤버제품을 생산하는 방법으로, 클론-앤-오운 방법으로 개발된 제품들을 SPL로 마이그레이션하기에 용이한 방법이다.

클론-앤-오운 방법을 포함하여 단일 제품 개발 방식으로 개발된 제품군으로부터 SPL로 마이그레이션하는 기존의 관련 연구는 소스코드로부터 공통 부분과 가변 부분을 찾아내는 데에 중점을 두고 있다[1, 4]. 그리고 이렇게 찾아낸 가변 부분들은 동일 소프트웨어 가변성[1]임에도 여러 가변 부분으로 나뉘어져 있는 경우가 종종 발견된다. 이러한 경우 가변성들 간의 관계가 설정되어야 하기 때문에 자동화된 제품라인의 공통성 및 가변성 분석 및 관리가 복잡해질 수 있다는 문제점을 가지고 있다.

본 논문에서는 이를 해결하기 위한 시작으로 먼저 각 소스 파일 내에서 변수를 정의하는 라인과 해당 정의가 영향을 미치는 라인을 하나의 코드 집합으로 그룹화하는 정의/사용(define/use) 기반 정렬 및 그룹핑 방법을 제안한다. 이 정렬 과정을 통해 관련성이 높은 코드 라인들이 하나의 그룹으로 표현되므로 라인 단위의 의존성을 블록 단위로 관리할 수 있게 된다. 이는 향후 SPL에서 관리해야 할 의존 관계가 줄어드는 것을 의미한다. 결과적으로 본 논문에서 소개하는 코드 그룹핑 방법은 SPL 가변성 분석을 용이하게 하는데 기여함으로써 복잡도 증가로 인해 발생할 수 있는 문제를 최소화하고 향후 SPL 유지 및 관리에서 상당한 이점을 제공할 것으로 기대된다.

2. 관련 연구

L. Linsbauer 등[6]과 W. Fenske 등[7]은 SPL로의 마이그레이션을 위해 일단의 유사 소프트웨어 제품군으로부터

[1] 소프트웨어 가변성은 소프트웨어 시스템 또는 산출물을 효율적으로 확장, 변경, 수정 또는 구성하는 능력을 의미[5]하는 용어로 제품라인의 멤버제품이 서로 어떻게 다른지를 의미하는 제품라인 가변성과는 구분되는 개념임. 이하 가변성은 소프트웨어 가변성을 의미함.

피처를 추출하는 방법을 제안하였다. E. Ghabach 등[8]은 클론-앤-오운 개발 방법으로 개발된 제품군을 기반으로 새로운 제품 변형을 도출할 때 가능한 시나리오와 추정되는 비용 정보를 소프트웨어 엔지니어에게 제공함으로써 파생 제품의 개발을 지원하는 방법을 제안하였다. T. Ishio et al.[9]은 Minwise 해싱 기술을 활용하여 소스 파일 집합을 입력으로 받아 유사한 파일과 구성 요소를 추출하는 코드 검색 방법을 제안한다. R. Lapeña et al.[1]은 새로운 제품에 대한 자연어 요구사항이 주어졌을 때 자연어 처리를 통해 요구사항과 관련성이 높은 레거시 제품 요구사항을 순위화하여 제공한다. 그렇지만, 이들 연구는 SPL 기반이 아닌 제품군으로부터 공통 또는 유사 코드 부분이나 요구사항을 도출하고 있으며 소프트웨어 가변성을 추출하는 방법에 대해서는 논의하고 있지 않다.

3. 정의/사용 기반 코드 그룹핑

본 논문에서 사용한 코드 그룹핑 방법은 특정 변수의 정의와 사용에 기반을 둔다. 따라서 주어진 원본 소스 코드를 분석하여 특정 변수가 정의되는 라인과 정의된 변수가 사용되는 라인을 식별하고, 이들 간의 의존성에 따라 적절한 순서로 정렬한다. 그림 1은 본 논문에서 소개하는 정렬 방법의 세부 단계를 나타낸다. 그리고 그림 2는 제안한 정렬 방법을 통해 정렬된 코드에 대한 예시를 보여준다.

그림 1. 소스 코드 정렬 과정

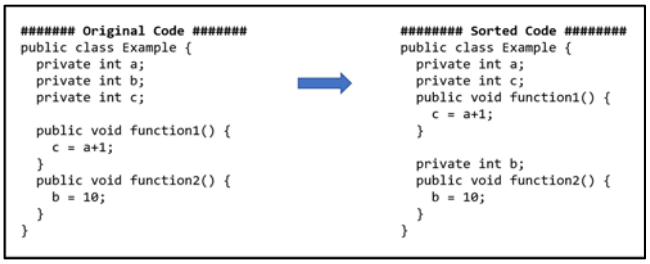


그림 2. 소스 코드 정렬 결과 예

3.1 토큰화 (Tokenization)

토큰화 단계에서는 원본 소스코드를 읽어 주석과 공백 문자를 제거하고, 유형에 따라 토큰을 식별하는 단계이다. 이 단계에서는 토큰을 예약어(keyword), 상수(constant), 식별자(identifier), 기호(symbol) 등 4가지 유형으로 분류한다.

3.2 의미 분석

의미 분석 단계에서는 특정 프로그래밍 언어의 특성에 따라 식별한 토큰의 의미를 파악하는 단계이다. 이 단계의 주요 목적은 해당 토큰이 어떠한 변수와 대응되며, 대응하는 변수가 소스 코드 내에서 어떠한 용도로 쓰이는지 파악하는 것이다. 본 논문에서 정의한 변수의 용도는 정의(Definition), 할당(Assignment), 사용(Use)으로 총 3가지이다. 정의(Definition)는 메모리 공간을 할당하고 해당 메모리 공간을 참조하기 위해 사용자가 지정한 이름을 부여하는 것을 의미한다. 대입(Assignment)은 정의된 변수의 메모리 공간에 실제 사용될 값을 저장하는 것을 의미하며, 사용(Use)은 변수 이름을 매개로 변수에 저장된 값을 참조하는 것을 의미한다.

3.3 구조화

구조화 단계에서는 용도가 파악된 토큰을 코드 라인과 코드 블록으로 조립하여 트리(Tree) 형태로 표현하는 단계이다. 코드 라인은 토큰 또는 코드 블록의 조합으로 구성되며, 하나의 코드 라인에 속하는 내부 요소는 정렬이 불가능하다는 특징을 가진다. 비록 소스 코드 내에서 다수의 라인으로 표현될지라도 하나의 코드 라인으로 간주해야 하는 경우가 존재한다. 예를 들어 *ELSE* 또는 *IF-ELSE*를 이끄는 조건문은 순서가 바뀌거나 서로가 분리되어서는 안되므로 하나의 코드 라인으로 간주되어야 한다. 코드 블록은 코드 라인의 집합이며 코드 블록 내에서는 코드 라인간 의존성을 고려하여 정렬하는 것이 허용된다. 또한 각각의 코드 라인과 코드 블록은 그들 내부에서 쓰인 변수의 이름과 용도를 기억하고 있다. 그림 3은 구조화 단계의 결과물인 코드 트리의 구조를 보여준다.

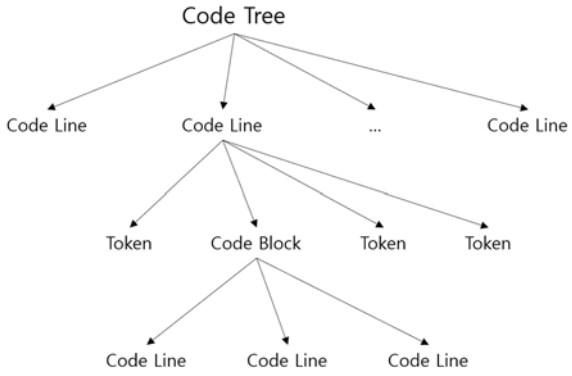


그림 3. 코드 트리의 구조

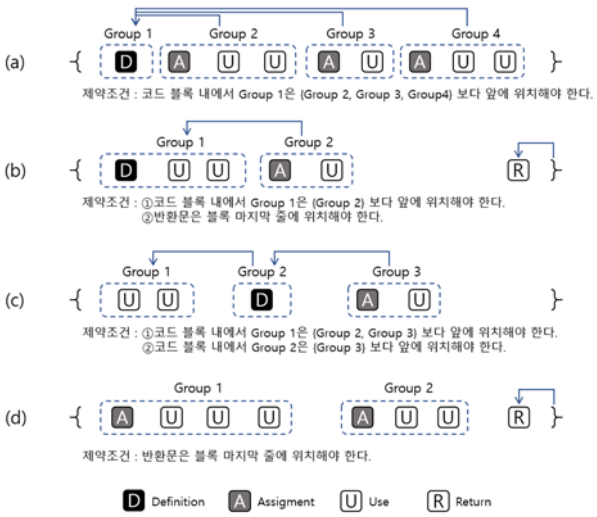


그림 4. 그룹화 기준 및 제약 조건

3.4 코드 정렬

코드 정렬은 각 변수에 대한 정보를 기반으로 관련된 여러 코드 라인을 하나로 엮는 과정이다. 이 과정은 코드 트리 내부의 모든 코드 블록에서 수행되며, 트리의 말단 노드에서 상위 노드로 순차적으로 수행된다. 코드 블록 내부에서의 정렬 과정은 정의된 규칙에 따라 수행된다. 그림 4는 코드 정렬 과정에서 특정 변수와 연관된 코드 라인을 그룹화하는 기준과 그룹간 제약 조건을 시각적으로 나타낸다.

4. 실험 및 평가

4.1 실험

제안한 코드 정렬 방법의 효과를 확인하기 위해 클론-앤-오운 개발 방법으로 개발된 ApoGames에 제안 방법을 적용하였다. ApoGames는 2006년 Dirk Aporius가 오픈소스 게임으로 처음 개발하여 클론-앤-오운 개발 방법으로 계속 진화시켜 온 제품군이다[2].

우리는 ApoAgmes의 자바 프로젝트 20개에 대하여 코드 정렬 방법을 적용하고, Diff Algorithm을 사용해 클론-앤-오운 개발 방법으로 작성된 클래스들 간의 공통성과 가변성을 추출하였다. 그리고 추출된 공통성과 가변성은 조건부 컴파일 구문을 사용해 하나의 소스 파일로 통합하였다.

조건부 컴파일 구문은 소스 코드 내에서 가변성을 표현하는 대표적인 방법으로 이렇게 통합된 소스파일 내에서 조건부 컴파일 구문에 의해 분할되는 코드 조각은 추출식 SPL에서 관리되어야 할 가변성과 대응된다.

표 1 클론 클래스에 대한 가변성의 수

파일명	정렬 전	정렬 후
\org\apogames\ApoApplet	3	3
\org\apogames\ApoComponent	163	**153**
\org\apogames\ApoComponentBufferedStrategy	211	**198**
\org\apogames\ApoConstants	**21**	22
\org\apogames\ApoIO	**14**	41
\org\apogames\ApoMain	23	**21**
\org\apogames\ApoMainBufferedStrategy	**41**	55
\org\apogames\ApoNewThread	12	12
\org\apogames\ApoScreen	**104**	142
\org\apogames\ApoSubGame	60	**54**
\org\apogames\ApoThread	**175**	176
\org\apogames\ApoTimer	6	6
\org\apogames\entity\ApoAnimation	**110**	137
\org\apogames\entity\ApoButton	114	**72**
\org\apogames\entity\ApoEntity	**72**	79
\org\apogames\entity\ApoNewTextfield	107	**66**
\org\apogames\help\ApoFileFilter	21	**19**
\org\apogames\help\ApoHelp	**144**	162
\org\apogames\help\ApoHighscore	**93**	212
\org\apogames\image\ApoImage	12	12
\org\apogames\image\ApoImageFromValue	143	**107**
\org\apogames\image\ApoImageScale	12	12
\org\apogames\image\ApoRawScale3x	9	9
\org\apogames\input\ApoKeyboard	**60**	93
\org\apogames\input\ApoMouse	28	28
\org\apogames\sound\ApoMP3Sound	**2**	4
\org\apogames\sound\ApoSounds	36	**31**
\org\apogames\sound\AudioPlayer	30	**18**

이 실험에서는 코드 정렬 방법의 적용 전과 후에 대하여 조건부 컴파일 구문에 의해 분할되는 코드 조각의 수를 비교함으로써 코드 정렬 방법이 가변성 식별에 미치는 영향을 분석하고자 한다. 공정한 비교를 위해 주석 제거 및 스타일을 일치시키는 과정은 코드 정렬 전과 후 모두 동일하게 적용하였다.

표 1은 ApoGames의 제품군 내의 변종 게임들 간의 가변성을 포함하는 28개의 자바 파일에 대한 실험 결과로 정렬 전후 컴파일 구문에 의해 분할되는 코드 조각의 개수 변화를 보여준다. 실험 결과 가변성을 포함하는 28개의 클론 파일 중 10개의 파일에서 가변성 개수가 감소하였으며, 다른 11개의 파일에서는 가변성 개수가 증가하였다. 그리고 나머지 7개의 파일에 대해서는 가변성 개수의 변화가 없었다. 본 연구에서 제안한 공유 코드 정렬 방법의 소스 코드는 깃허브에서 다운로드 받을 수 있다[3].

4.2 평가

우리는 가변성 변화의 원인을 면밀히 파악하기 위해 코드 정렬 과정에서 발생하는 대표적인 유형을 4가지로 분류하고 유형별로 가변성의 수에 어떠한 영향을 미치는지 분석하였다. 그림 5는 각 유형에 대한 예시를 보여준다.

(1) 가변성 감소 요인

코드 그룹핑을 위한 정렬 과정에서 가변성의 개수가 감소하는 대표적인 유형은 그림 5의 유형 (a)이다. 유형 (a)는 서로 다른 자바 소스 파일에서 일부 다른 변수가 정의되고, 해당 변수가 사용되는 블록의 헤더 또한 다른 경우이다. 따라서 정의된 변수와 변수 사용 블록이 하나로 그룹화됨으로써 가변성 개수가 줄어들게 된다. 이러한 유형은 가변성을 관리하는 측면에서 명백한 이점을 제공한다.

(2) 가변성 증가 요인

코드 그룹핑을 위한 정렬 과정에서 가변성의 개수가 증가하는 대표적인 유형은 그림 5의 유형 (c)이다. 특정 변수와 관련된 블록이 해당 변수와는 무관한 다른 블록과 의존 관계를 가지며 서로 연속하여 위치할 때 발생하는 경우로 정렬 과정에서 두 블록의 연속성이 사라지면서 가변성의 개수가 증가하게 된다. 이 경우 가변성의 수가 증가하는 방향으로 작용하지만, 이러한 가변성 증가가 부정적인 것만은 아니다. 이는 변수와의 관련성 측면에서 무관한 부분을 사전에 분리함으로써 보다 견고한 가변성 관리를 유도할 수 있다.

(3) 가변성 불변 요인

그림 5에서 유형 (b)와 (d)는 정렬 후에 가변성 개수에 영향을 미치지 않는 것으로 나타난다. 하지만 우리는 2가지 경우에 대해 고민해 볼 필요가 있다.

유형 (b)의 경우에는 서로 다른 변수가 사용되지만 동일한

[2] https://bitbucket.org/Jacob_Krueger/apogamessrc

[3] https://github.com/WareEngineer/KCSE2020

그림 5. 코드 정렬 유형

블록의 헤더를 가지는 경우이다. 이러한 경우 서로 다른 변수의 사용은 향후 더 많은 가변성을 발생시킬 가능성을 내포하고 있다. 그러므로 블록의 헤더와 일부 내부 코드가 중복되더라도 유형(a)와 같이 더 큰 범주로 그룹화는 것이 효과적일 수 있다.

유형 (d)의 경우에는 사용된 변수가 없으므로 정렬의 대상이 되지 않는다. 그리고 Diff Algorithm의 특성상 순차 비교를 통해 가변성을 식별하기 때문에 동일한 메소드의 구성일지라도 순서에 의해 가변성으로 인지될 수 있다. 따라서 이러한 상황을 방지하기 위해서는 메소드에 대하여 일정한 그룹핑을 위한 정렬 규칙을 적용할 필요가 있다.

5. 결론

추출식 SPL을 수행할 때 전처리 없이 단순 코드 비교를 통해 가변성을 식별하는 과정은 가변성의 단위가 라인 단위로 식별될 여지가 많다. 따라서 본 논문에서는 추출식 SPL을 수행하기 전, 보다 큰 단위로 가변성을 식별할 수 있도록 전처리 방법 중 하나로써 코드 정렬 방법을 소개하였다. 코드를 정렬하는 과정은 4단계 세부 과정을 거쳐 수행되며, 코드에서 식별된 변수를 기준으로 해당 변수와 관련이 있는 코드 라인을 하나로 엮음으로써 이루어진다.

코드 정렬 방법에 대한 실증적 평가를 위해 클론-앤-오운 방법으로 개발된 ApoGames에 해당 방법을 적용하고 적용 전과 후에 식별된 가변성의 개수 변화를 관찰하였다. 그리고 가변성의 개수 변화의 요인을 파악하기 위해 대표적인 정렬 유형을 4가지로 분류하고 그 원인을 분석하였다. 이러한 분석을 통해 코드 정렬 방법이 가변성을 식별함에 있어 일부 긍정적인 영향을 미치는 것을 확인할 수 있었다.

그러나 본 논문에서의 검증은 자바 프로젝트에 한해서 수행되었으며, 이러한 검증 결과가 다른 모든 프로그래밍 언어에 일반적으로 적용된다고 단언할 수 없다. 향후 연구에서는 정의/사용 이외의 정렬 기준을 제안하고 이를 기반으로 한 가변성 분석 효과를 포함하여 다양한 프로그래밍 언어에 방법 적용해 보고자 한다.

Acknowledgement

본 연구는 2019년도 중소벤처기업부의 기술개발사업 지원에 의한 연구임 (S2796409)

참고문헌

[1] R. Lapeña, M. Ballarin, and C. Cetina, "Towards clone-and-own support: locating relevant methods in legacy products," in *SPLC*, pp. 194-203, 2016.

[2] Y. Dubinsky, J. Rubin, T. Berger, S. Duszynski, M. Becker, and K. Czarnecki, "An exploratory study of cloning in industrial software product lines," in *CSMR*, pp. 25-34, 2013.

[3] E. Ghabach, "Supporting Clone-and-Own in software product line," 2018.

[4] C. Lima, I. do Carmo Machado, E. S. de Almeida, and C. von Flach G Chavez, "Recovering the product line architecture of the apo-games," in *SPLC*, pp. 289-293, 2018.

[5] K. Pohl and A. Metzger, "Software Product Lines," in *The Essence of Software Engineering*: Springer, Cham, pp. 185-201, 2018.

[6] L. Linsbauer, R. E. Lopez-Herrejon, and A. Egyed, "Variability extraction and modeling for product variants," in *Software & Systems Modeling*, vol. 16, no. 4, pp. 1179-1199, 2017.

[7] W. Fenske, J. Meinicke, S. Schulze, S. Schulze, and G. Saake, "Variant-preserving refactorings for migrating cloned products to a product line," in *SANER*, pp. 316-326, 2017.

[8] E. Ghabach, M. Blay-Fornarino, F. El Khoury, and B. Baz, "Clone-and-Own software product derivation based on developer preferences and cost estimation," in *RCIS*, pp. 1-6, 2018.

[9] T. Ishio, Y. Sakaguchi, K. Ito, and K. Inoue, "Source file set search for clone-and-own reuse analysis," in *MSR*, pp. 257-268, 2017.

Enhancing Patch Validation in APR by Introducing Test Case Prioritization and Sampling

Venugopal Yazhini[1,*], Quang-Ngoc Phung[2], Eunseok Lee[3,*]
[1,2] Electrical and Computer Engineering, Sungkyunkwan University, South Korea
(yazhiniv, ngocpq@skku.edu)
[3,*] College of Software, Sungkyunkwan University, South Korea (leees@skku.edu)

Abstract

Automatic Program Repair(APR) is an active research topic in the area of automatic debugging. In general, APR techniques require test suites to validate the automatically generated patches. However, the test suites used for patch validation may contain a large number of test cases. Running these test suites for every program variant makes the validation process not only time-consuming but also expensive. To mitigate this issue, we introduce a) MPTPS(Modification Point-aware Test Prioritization and Sampling) that iteratively records test execution and performs prioritization and sampling to reduce the test execution time; and b) A concrete fitness function that refines the existing fitness function to improve repair efficiency. We implemented our approach MPPEngine in Astor workspace by extending jGenProg. The experiments on the Defects4j benchmark against jGenProg show that on average jGenProg consumes 79.27s to validate one program variant where MPPEngine takes only 33.70s results in 57.50% of validation time reduction. Also, MPPEngine outperforms jGenProg by finding patches for 6 more bugs than jGenProg.

1. Introduction

The increase in software complexity often results in high debugging and maintenance costs. A recent study on software depicted that the debugging process of the complex software is time-consuming and tedious. In addition, it takes up to 50% of overall project expenses [1]. Bug fixing is one of the main factors that cause this high expense. To overcome this problem Automatic Program Repair (APR) has been introduced.

APR provides a solution to the buggy program by debugging the bugs automatically without any human intervention. So, APR plays a vital role in automated debugging [2]. There are two main approaches to APR. 1) Generate and Validate approach: It modifies the original buggy program by applying a set of change operators. And as the name suggests, it generates the patch and validates it against the test suite; and 2) Semantic-driven approach: It formally or explicitly encodes the buggy program, for instance into a formula that produces solutions which are expected to fix the bug [3].

Initially, the fault localizer will be fed with a buggy program and its test suite as an input. It identifies suspicious statements in the program and each suspicious statement will be given a suspicious score. Then the patch is generated using the Generate & Validate approach or Semantic-driven approach. Once a patch has been generated, the validation process takes place. In the validation process, the generated patch is evaluated using test cases to ensure that the patch does fix the bug and does not introduce any new issue(s). If the patch passes all the test cases in the test suite then it is called a test suite adequate patch. Otherwise, it is considered as an incorrect patch. Le Goues et al.[4] introduced GenProg which is one of the first APR tools based on genetic programming. It is a state-of-art tool developed for C program automated repair. Martinez et al.[8,9] proposed jGenProg as part of Astor which is a java version of GenProg. Java programs can be repaired automatically with the help of jGenProg.

Unfortunately, chances are that the validation test suite includes a large number of test cases. So, it consumes a lot of time for test execution resulting in high cost. As Rothermel et al.[5] reported in their paper, one of their products took seven weeks to execute the entire test suite against a software bug. On the other hand, APR produces a lot of invalid patches that should be identified and filtered out using test suites in patch validation. If the patch encounters a large test suite, it has to spend a significant amount of time to validate until it produces a patch. Sometimes, without executing the entire test suite the patch validation process ends due to timeout where we cannot trace the failing test cases. Qi et al.[6] proposed a tool named TrpAutoRepair that includes Fault Recorded Testing Prioritization(FRTP) to reduce the cost of testing. It was the first time to introduce insights of prioritization in automated program repair. Again, Qi et al.[10] used the prioritization techniques[6] in RSRepair to identify the invalid patches early in the validation process. In both approaches, the same test suite is prioritized and used for validating all the program variants. So, we decided to improve the patch validation by identifying invalid patches in the early stage of

validation through modification point aware fault-based test case prioritization. Thus, it speeds up the whole validation process in an effective way. Previous techniques run all the test cases in the test suite even if there are failed test cases. The number of failing test cases is used to calculate the fitness score. But in our approach, we decided to implement test case sampling to run test cases in small subsets. Test case subsets are then executed one by one until it encounters a failure with the set. Based on the test case failing count and executed test case information, we introduced a new fitness function. This will help in improving the repair efficiency of our approach.

This paper includes the following contributions to enhance the patch validation in APR,

1) We implemented MPTPS(Modification Point-Aware Test Prioritization and Sampling) to reduce test execution time.
2) A new enhanced fitness formula to regulate the already available fitness function in Astor.
3) Also, the initial experiments of the proposed method conducted in the Defects4j benchmark against jGenProg.

The rest of this paper is arranged as follows. Section 2, introduces our proposed approach, MPPEngine in detail. Section 3 presents our experiments and results. Section 4 gives insights into our future research direction. In Section 5, we conclude this paper by summarizing our contribution.

2. Proposed Approach

Patch validation in APR is often time-consuming and expensive due to large test suites. It becomes more serious when the software is large and complex. Also, the fitness function in Astor is calculated by adding up the number of failed test cases in the test suite. Our approach's objective is to reduce the time of patch validation in APR by prioritizing and sampling the validation test suite for every program variant with the help of modification point information. Also, presenting a concrete fitness function to improve repair effectiveness. Although many kinds of research have been going on prioritization in regression testing, it is not common in APR patch validation. Our approach will help to identify invalid patches at the early stage of validation. Therefore, our contributions in this paper help in reducing the test execution time while improving the repair efficiency.

2.1. Modification Point-Aware Test Prioritization and Sampling(MPTPS)

Figure 1 Clearly explains the overview of our proposed method MPTPS(Modification Point-Aware Test Prioritization and Sampling).

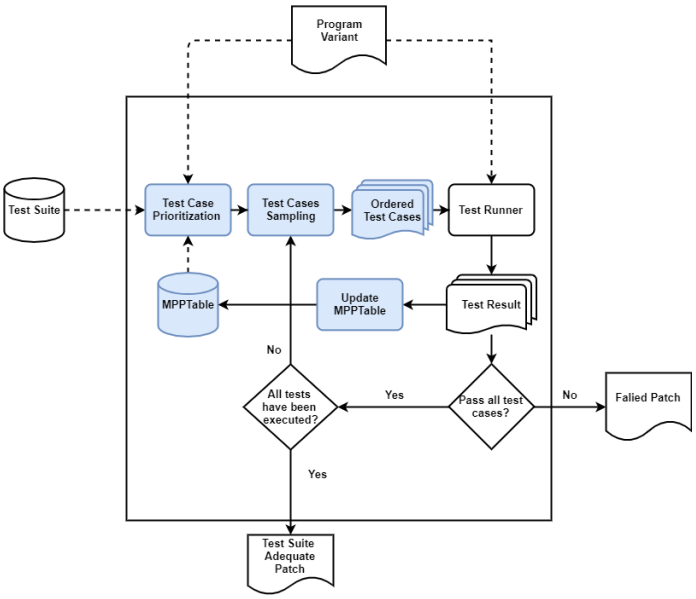


Figure 1. MPPEngine: MPTPS Overview

Initially, *Test Runner* takes *Program Variant* and *Ordered Test Cases* from the *Test Suite* as an input for the validation. From the *Test Results*, we extract failing test case information and *Update MPPTable*. After the *MPPTable* is updated with the test execution results, *Test Case Prioritization* takes place by prioritizing patch killing count in descending order from the table(explained detail in the next section). Prioritized test cases then undergo *Test Cases Sampling* phase to split the test suite into multiple subsets with each having an equal number of test cases. These *Ordered Test Cases* subsets are then executed one by one until the last subset, in case if there are no failing tests. If all the subsets pass, it is delivered as a *Test Suite Adequate Patch* to the developer. Otherwise, the *Test Runner* will be stopped after completely executing the subset with a failing test case(s). In this case, the outcome is considered *Failed Patch*. Subsequently, the fitness score is calculated for the patch using test case execution information and based on the score, the *Failed Patch* might undergo modification once again.

Modification Point-Aware Test Prioritization. We implement the modification point aware prioritization by extending the FRTP approach[6]. During validation, for every program variant, modification point information, related test cases and its failing count(known as patch killing count) are mapped together. Patch killing count is nothing but a number of times a test case makes a program variant fail. These details will be updated and stored in a table format(MPPTable) for each program variant. By monitoring the test cases that make a program variant fail, we record that information to calculate the patch killing count by adding(increasing) value '1' to every failing test cases in the

patch killing count column of the MPPTable. The prioritization steps are as follows,

1) MPPTable is updated every time whenever a new program variant is generated for the modification point(s).
2) Test cases in the table are prioritized based on the patch killing count of the test case in a descending order i.e. placing the test cases with largest to smallest patch killing count.
3) If the test cases have the same patch killing count, it will be ordered randomly among themselves.

Therefore for every program variant, a different prioritized test suite is used.

2.2. Concrete Fitness Function

In Astor, whenever a patch encounters a failed test case, it sums up the failed test case count and set the fitness value. Following the same fitness function in our approach is not considered accurate and efficient as we run test cases in subsets from the prioritized test suite. For example, in our approach, 1 failure out of 50 test cases is different from 1 failure out of 100 test cases. So, we are introducing a new fitness function to alleviate the fitness function efficiency. The fitness function will be calculated based on the following formula,

$$F_s = \frac{T_E}{T_T} - \frac{T_F}{T_E}$$

Where,

T_T – Total Test Case

T_E – Executed Test Case

T_F – Failed Test Case

Lemma 1. Let's assume there exist two patches: p1, p2 which has the same number of test cases. Patch 'p1' provides the best fitness, if p1(T_E)>p2(T_E) and p1($\frac{T_F}{T_E}$)<p2($\frac{T_F}{T_E}$)).

As the most likely to fail test cases are expected to be present in the initial subsets of the prioritized test suite, calculating fitness based on this formula would help in improving repair efficiency.

3. Evaluation

In this section, we describe our experimental setup, results and address some of our research questions.

3.1. Experimental Setup

As our proposed approach MPPEngine is implemented by extending jGenProg[4] in Astor[8,9] workspace, we chose to evaluate our approach on the Defects4j[13] benchmark to compare the experimental results with jGenProg. We conduct our experiments on 4 subject programs(Chart, Lang, Math and Time) with 224 bugs from the Defects4j benchmark as mentioned in Table 1. In Astor, the fault localization tool used is GZoltar[11] and it comes

Table 1. Defects4j benchmark for experiments[13]

Subjects	Bugs	KLOC	Test KLOC	No.of Test cases
JFree Chart	26	96	50	2205
Commons-Lang	65	22	6	2245
Commons-Math	106	85	19	3602
Joda-Time	27	28	53	4130
Total	224	231	128	12182

with the fault localization algorithm, Ochiai [12]. We use the same in our approach as well.

We run experiments on a virtual machine with Ubuntu OS, Intel Core-i5 processor, CPU with @3.50 GHz, and RAM of 4GB. The time limit for all the experiments set to 180minutes, the total generation set to 10000 generations in maximum and sampling size to 20(each sampled subset will have 20 test cases) for every bug. Also, to reduce randomness, we ran all the experiments 3 times and took the average value for the results.

3.2. Experimental Results

RQ1: To what extent our approach is effective in reducing test execution time comparing jGenProg?

We investigated the effectiveness of MPPEngine against jGenProg in terms of reducing test execution time as jGenProg is a base of java based generate and validate the approach. We ran experiments on the Defects4j[13] dataset used in Astor[8,9] to compare the results more effectively. Then we filtered out 48 bugs for which the APR tools in Astor generated patches. These bugs are from 4 project subjects of Defects4j(Chart, Math, lang and Time) to perform the experiments.

Table 2. Time taken to validate a program variant by jGenProg and MPPEngine.

Technique	Avg time for validating 1 program variant	Time reduction
jGenProg	79.27s	57.50%
MPPEngine	33.70s	

In terms of test case reduction with respect to time, MPPEngine is better for 42 bugs(42/48) and jGenProg is better for 6 bugs(6/48). Table 2 shows that on average jGenprog takes up to 79.27seconds to validate one program variant whereas MPPEngine consumes only 33.70 seconds. So, the time consumption on patch validation is reduced to 57.50%. This shows our approach is better in case of reducing test execution time.

RQ2: Is there any new patch generated for bugs by MPPEngine for which jGenProg does not produce any solution?

In RQ2, in fact, we focus only on the bugs repaired by both the approach in the given time limit and maximum generation. Figure 2 compares the patches generated by both approaches.

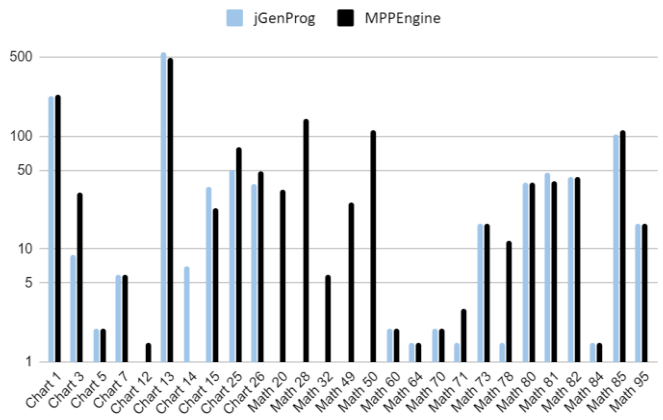


Figure 2. Number of patches generated per bug: jGenProg, MPPEngine

There are patches generated for 27 bugs in which (20/27) bugs repaired by both the approach in common. While MPPEngine generates patches for (26/27) bugs except for Chart 14 in the given time limit. Whereas jGenProg generates patches for this bug(Chart 13). On the other hand, MPPEngine produces patches for the following 6 bugs: Chart 12 and Math 20, 28, 32, 49 and 50 which have not been repaired by jGenProg.

It is clear that MPPEngine performs as good as jGenProg. But, in most cases, MPPEngine(26/27) outdo jGenProg(21/27). As a result, the repair effectiveness of jGenProg is 77.8% whereas MPPEngine is about 96.3% for the 27 bugs. Here, again MPPEngine outperforms jGenProg.

4. Future work

Even though our approach generates test adequate patch it can still be an overfitting patch. As some studies indicate stronger test suites can produce lesser overfitting patches[7], we plan to increase the test coverage by generating additional test cases using Dynamic Symbolic Execution tool to reduce the overfitting problem.

5. Conclusion

In this paper, We built MPPEngine including MPTPS and a new fitness function that reduces validation time and improves repair efficiency. The experiments on 48 bugs from 4 different Defects4j subjects proved to reduce average validation time to 57.50% reduction comparing jGenProg. As of repair effectiveness, our MPPEngine generates patches in most cases(26/27) and outperforms jGenProg(21/27). The experimental results clearly depict that MPPEngine outperforms jGenProg. So, if our method improves repair effectiveness and efficiency against jGenProg, implementing our method against other generate and validate APR tool would provide similar results as well.

Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT). (No. 2019R1A2C2006411)

References

[1] Undo Software: Increasing software development productivity with reversible debugging, Technical report white paper (2014)

[2] Zeller, A: Automated debugging: Are we close, Computer, vol. 34, no.11, pp. 26-31 (2001)

[3] Gazzola, L., Micucci, D., & Mariani, L.: Automatic Software Repair: A Survey, IEEE Transactions on Software Engineering, 45(1), pp. 34-67 (2019)

[4] Le Goues, C., Nguyen, T. V., Forrest, S., Weimer, W.: GenProg: A generic method for automatic software repair, IEEE Transactions on Software Engineering, 38(1), pp. 54-72 (2012)

[5] Elbaum, S., & Rothermel, G.: Prioritizing Test Cases for Regression Testing, IEEE Transactions on Software Engineering, 27(10), pp. 929-948 (2000)

[6] Qi, Y., Mao, X., & Lei, Y.: Efficient automated program repair through fault-recorded testing prioritization, IEEE International Conference on Software Maintenance, ICSM, pp. 180-189 (2013)

[7] Xiong, Y., Liu, X., Zeng, M., Zhang, L., & Huang, G.: Identifying Patch Correctness in Test-Based Program Repair, in Proceedings of the 40th International Conference on Software Engineering (2017)

[8] Martinez, M., & Monperrus, M.: Astor: Exploring the Design Space of Generate-and-Validate Program Repair beyond GenProg, Journal of Systems and Software, Elsevier (2019)

[9] Martinez, M., & Monperrus, M.: ASTOR: Evolutionary Automatic Software Repair for Java, Cornell University Library (2014)

[10] Qi, Y., Mao, X., Lei, Y., Dai, Z., Wang, C.: The strength of random search on automated program repair, in Proceedings of 36th International Conference on Software Engineering, pp. 254-265 (2014)

[11] Gzoltar Homepage (2017). http://www.gzoltar.com

[12] Abreu, R., Zoeteweij, P., van Gemund, A.J.C.: On the accuracy of spectrum-based fault localization, in Proceedings of IEEE the Testing: Academic and Industrial Conference Practice and Research Techniques, pp. 89-98 (2007)

[13] Just, R., Jalali, D., Ernst, M.D.: Defects4J: a database of existing faults to enable controlled testing studies for Java programs, in Proceedings of ACM the International Symposium on Software Testing and Analysis, pp. 437-440 (2014)

PVFS의 보안성 향상을 위한 Temporal Logic 기반의 허위 보고서 판단 방법

안정섭 O, 조대호
성균관대학교 전자전기컴퓨터공학과 O
성균관대학교 소프트웨어학과
sc4217@skku.edu O, thcho@skku.edu

Temporal Logic Based on False Report Detection Method for PVFS Security Improvement

Jung-sub Ahn O , Tae-ho Cho
Sungkyunkwan University, Department of electrical and computer engineering O
Sungkyunkwan University, Department of computer science and engineering

요 약

무선 센서 네트워크(Wireless Sensor Networks)는 도메인에 배치된 수많은 센서 노드들이 상호 통신을 통해 네트워크를 자가구성(self-organization)하고 필드 상황을 모니터링한다. 센서 노드는 광활한 대지에 배치되어 개별적인 노드 관리가 힘들다는 취약점을 갖는다. 따라서 공격자는 훼손시킨 노드를 통해 허위 보고서 주입 공격을 시도할 수 있다. 이러한 응용계층의 공격을 막기 위해 Probability Voting based Filtering Scheme과 같은 여러 보안 프로토콜들이 제안되었으며, 메시지 인증 코드 기반의 보고서 인증을 하며 높은 보안율을 보여주었다. 하지만 메시지 인증 코드 기반의 프로토콜들은 모든 메시지 인증 코드가 허위 이벤트 데이터를 가지는 강도 높은 공격에서는 보안을 달성할 수 없다는 취약점이 남아 있다. 따라서 본 논문에서는 메시지 인증 코드로 보안을 할 수 없는 강도 높은 공격상황에서 Temporal Logic과 시뮬레이션 기반의 통계기반 보안모델을 제안한다. 이를 통해 베이스 스테이션에서는 이벤트 내용을 기반으로 한 보고서 진위탐지를 실행하여 허위 보고서 공격을 방어할 수 있다.

1. 서 론

WSN은 필드에 배치된 센서 노드들을 이용하여 전쟁상황, 재난상황, 홈 네트워크 시스템 등에서 정보 수집을 위해 사용된다. 센서 노드는 네트워크의 비용을 절약하기 위하여 제한된 성능을 가진다. 특히, 센서 노드들은 공개된 환경에 배치되어 있으며 공격자에 의해 쉽게 훼손될 수 있다. 네트워크 공격자는 이러한 취약점을 이용하여 허위 보고서 삽입 공격이나 잘못된 메시지 인증 코드(Message Authentication Code; 이하 MAC)를 주입하는 등 네트워크를 마비 시킬 목적으로 다양한 종류의 공격을 시도할 수 있다. 이러한 공격은 실제로 발생하지 않은 이벤트를 보고서에 주입해 노드들의 불필요한 에너지 소비를 유발하거나 관리자에게 잘못된 정보를 전달한다.

이러한 공격을 막기 위해 여러 보안 프로토콜들이 제시되었다[1-3]. 이 보안 프로토콜들은 공통적으로 대칭 키 보안 기법인 MAC을 통해 보고서의 무결성(integrity)을 검사하며 전달 노드에서 실시하는 보고서 검증 과정을 통해 보고서가 거짓으로 판별할 경우 보고서를 조기 폐기하는 방법으로 네트워크 보안과 노드의 에너지 효율성을 높인다. 또한, 기존 보안 프로토콜들은 보고서에 포함되는 모든 MAC 검증을 보고서 수집 기관인 기지국(Base Station; 이하 BS)에서 하기 때문에 보고서에 포함된 모든 MAC이 공격당하지 않는다면 정상적인 필터링이 가능하므로 허위 알람을 피할 수 있다. 하지만, 만일 허위 보고서에 포함된 모든 키 값이 탈취되어 정상 MAC으로 인식하는 허위 MAC들을 가진다면 BS에서의 2차 검증은 무효화되며, 허위 이벤트 내용에 대한 부적절한 대응(예: IoT 기기 오작동, 허위 알람 등)과 같은 문제를 야기할 수 있다.

허위 보고서를 막기위해 제안된 MAC기반 보안 프로토콜들은 전체 MAC 훼손에 대한 보안을 할 수 없으며 이러한 광범위한 노드 탈취 상황에서는 응용계층에서 진행하는 허위보고서 주입공격 이외의 공격을 감행할 것이라고 명시하고 있다[4]. 그러므로 여전히 전체 MAC 탈취 공격은 발생될 수 있으며, 이를 고려한 보안 해결책이 필요하다. 따라서 본 논문에서는 기존의 보안 프로토콜에서 검증을 할 수 없는 상황에 적용할 3차 검증 기법으로 Temporal Logic (TL) 기반의 통계모델을 적용하여 전체 MAC 탈취 상황에서의 허위보고서 공격에 대한 보안 모델을 제안한다. TL을

이용한 통계모델 기반의 허위보고서 보안이란 WSN 환경과 동일한 가상 시뮬레이션 환경을 구축하여 시뮬레이션 내에서 처리되는 각 이벤트 진행시간 별 노드들이 가질 수 있는 상태 값들에 대한 통계를 데이터베이스에 저장, 정상적인 상태전이만을 표현한 TL룰과 상태변이에 대한 통계와 순위(ranking)를 적용하여 노드들의 상태가 정상인지 비정상인지를 파악하는 모델이다. 상태변이에 대한 통계 기반 순위 알고리즘의 도입은 지능적인 공격의 탐지 정확도를 향상시키고 정상 노드에 대한 오탐지율을 개선하기 위해 적용한다.

2. MAC기반 보안 프로토콜 한계점

기존 보안 프로토콜에서는 중간 여과와 기지국의 최종 여과 단계를 통해 허위 보고서를 필터링한다. 허위 보고서를 검증하기 위해서는 보고서에 포함된 MAC의 무결성을 검사한다. MAC기반의 보안 프로토콜은 대부분 En-route 필터링을 사용하며 배치되는 노드의 모든 키를 BS의 키풀에서 관리하기 때문에 보고서에 포함된 모든 MAC이 훼손되지 않는다면 최종 여과 단계에서 정상적인 필터링이 가능하다. 하지만 보고서에 포함된 MAC이 훼손된다면 공격자는 최종 여과 단계에서도 탐지할 수 없는 허위 보고서를 작성할 수 있으므로 보안의 한계가 존재한다. 이러한 취약점은 이벤트 내용에 의존적인 제어 응용 시스템에 잘못된 동작을 일으키므로 이러한 문제를 해결하는 것은 중요하다.

본 논문에서는 MAC 기반 보안 프로토콜에서 보고서의 모든 MAC이 훼손된 상황을 고려하여 통계모델을 통해 상황 인식을 하고 허위 보고서를 탐지하는 방법을 제안한다.

3. Temporal Logic

시공간을 나타내는 Temporal Logic은 시간 개념과 공간 개념을 동시에 포함하고 있는 논리이다. Temporal Logic은 시간 흐름에 따라 변화하는 요소들을 탐지하고 복합적인 추론을 한다. 이는 기존에 존재하는 사건, 조건, 공간개념으로만 다루어진 논리보다 시간 개념이 확장되어 시간에 따른 구체화된 시공간형태로 표현이 가능하여 복잡한 시스템에서 새로운 관계 지식을 추론할 수 있다. 표 1은 Temporal Logic에서 사용되는 심볼들 중 일부를 나타낸다.

표 1 Temporal Logic 심볼 종류 [4]

Character	Symbolic	Explanation
N φ	○ φ	*다음(Next): 바로 다음 번에는* φ 가 참이다
F φ	◇ φ	*가능한 미래(sometime in Future): 결국 언젠가는* φ가 참이다
G φ	□ φ	*확정된 미래(Always in future): 항상 언제나* φ는 참이다

4. 확률적 투표기반 필터링 기법 (PVFS)

네트워크에서 발생하는 허위 보고서 공격을 막기 위해 다양한 응용 계층 보안 프로토콜이 제안되었다[5]. 그 중 확률적 투표 기반 필터링 기법에서는 이벤트를 탐지한 센서 노드가 이벤트 정보에 대한 보고서를 작성하는데 멤버 노드들로부터 보고서의 진위여부를 가리는 MAC을 삽입한다.

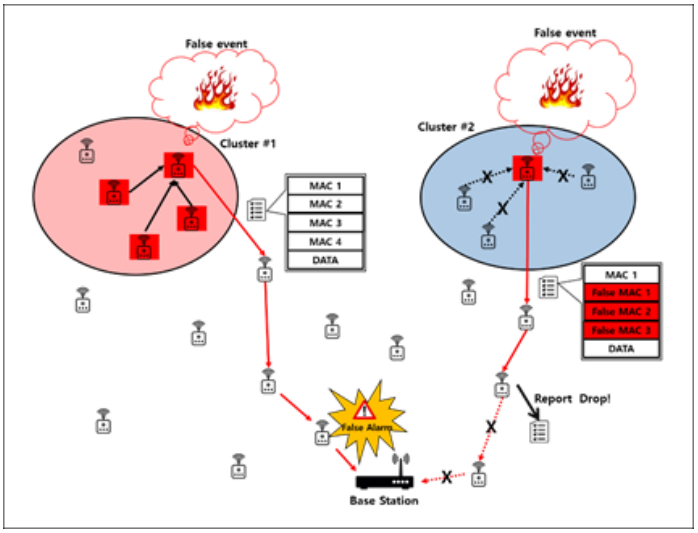


그림 1 다수의 훼손된 노드를 이용한 허위 보고서 삽입 공격

만일 보고서가 거짓된 보고서라면 삽입된 MAC들은 비정상 적인 값을 갖게 되어 En-route filtering 과정에서 허위보고서로 판별되어 보고서 조기 폐기된다. 작성된 보고서에는 네트워크 관리자가 사전에 정의된 임계 값만큼 MAC을 포함시킨다. 하지만 그림 1과 같이 네트워크 공격자는 지역적으로 여러 노드를 탈취하는 클러스터 단위 공격을 감행할 수 있으며 보고서에 포함되는 모든 MAC들이 공격자로부터 훼손된 노드로부터 부착되거나 MAC을 구성하는 키 정보가 노출되어 있다면 해당 보고서에는 모두 정상적인 MAC이 부착되어 거짓 이벤트 정보를 담은 허위보고서가 정상적인 보고서로 판별되어 En-route filtering을 통과하고 BS에서도 제대로 된 검증이 이뤄지지 않는다. 그 결과 BS에서는 허위 알람이 발생하거나 잘못된 행동을 하게 된다. 이러한 내용은 여러 보안 프로토콜의 약점으로 지적되었다[1].

이를 해결하기 위해 필드의 상황을 인식하여 보안을 수행하는 상황인식 아키텍처가 제안되었다[6]. 하지만 해당 아키텍처는 상황인식을 위한 다량의 센서 노드의 추가배치가 필수적이며, 이에 따른 비용추가와 공격상황의 인식은 되지만 정확한 훼손된 노드의 파악은 할 수 없다는 단점이 있다. 또한, 클러스터 헤드(Cluster Head; 이하 CH)노드로 사용하는 고사양의 노드가 필요하며, 이러한 경우, 네트워크 사용자는 CH와 맴버 노드를 적절히 배치하여 클러스터를

구성해야 하기 때문에 노드의 배치를 랜덤하게 할 수 없다는 단점이 존재한다.

5. Temporal Logic 탐지 방법

우리는 광범위한 노드 훼손 공격에서 허위 보고서 탐지와 훼손된 노드 및 클러스터를 찾기 위해 TL과 통계 기반의 노드 비정상 행동 탐지 알고리즘을 제안한다. 제안 방법에서는 TL을 기반으로 하여 비정상 행동 패턴을 감지하는 1차 탐지를 하고 지능화된 공격을 감지하기 위한 확률 기반의 2차 탐지를 진행한다.

1차 탐지에 사용되는 TL기반의 노드 비정상 행동 탐지 알고리즘이란 노드가 탐지하는 이벤트들의 정상적인 상태 전이에 대한 TL 규칙을 작성하고 각 노드들이 BS로 보내는 보고서의 내용이 TL 규칙에 어긋나는 상태 전이를 보이면 해당 보고서를 전송하는 노드를 공격당한 노드로 의심하는 알고리즘이다. TL 규칙에 의거하여 정상적이지 않은 노드의 변화를 탐지하여 올바르지 않은 행동 전이를 일으키는 보고서를 지식 베이스를 통해 이를 탐지한다. 그러므로 기존 보안기법에서는 이벤트 내용을 고려하지 않기 때문에 이를 탐지할 수 없다. 하지만 이 알고리즘은 광범위한 센서 노드 훼손 상태의 허위 이벤트 삽입 공격을 막는 것에 효과적이지만, 공격자는 논리적이지 못한 상태전이를 일으키지 않고 지능적인 공격을 할 수 있기 때문에, 이러한 상황에 대한 대비책으로 시뮬레이션 기반의 통계기반 랭킹 알고리즘을 도입하여 지능형 공격을 막는 알고리즘을 추가로 도입하였다. 통계기반 랭킹 알고리즘이란 이벤트가 발생한 후, 각 단위시간마다 노드가 보일 수 있는 여러 상태변이들에 대한 통계 기반의 순위를 정하여 가장 낮은 순위의 상태전이를 보이는 노드를 공격 의심 노드로 선정하여 필터링 하는 알고리즘이다.

Algorithm 1: Compromised node detection

```
M ← MAX_UT,
N ←Number of nodes, MAX_UT initialize by network field
for n←0 to N do
    set node[n].T to 0
end for
for i ←0 to M do
    for n←0 to N do
        if node[n].state is abnormal state for i
            set node[n].tag to true
        else
        if node[n].probability is abnormal state
            set node[n].T to T+1
        else continue
    end for
        if node[n].tagcount > MAX_UT
        set node[n].compromised to true
    end for
```

Algorithm 1은 훼손된 노드를 탐지하기 위해 적용된 TL 검사 알고리즘의 슈도 코드를 나타낸다.

상태전이에 대한 룰은 TL룰을 사용하였으며, 제안기법에서는 각 단위시간마다 상태전이에 대한 확률을 적용하여 공격 탐지율을 향상시키고 정상 노드에 대한 오 탐지율을 낮췄다.

표 2에서 보여주는 Temporal Logic 규칙은 본 연구의 제안 방법의 화재의 이벤트 탐지 시나리오로써 작성되었다. 각 노드는 필드의 온도를 기반으로 정상, 화재진행 중, 연소 끝과 같은 3개의 상태를 갖게 된다. 이와 같은 상태는 온, 습도 센서 및 기타 여러 센서 모듈을 통해 판단할 수 있는 값이다. 화재 변이에 대한 규칙은 다음과 같다.

표 2 화재 시나리오 TL 규칙

no	Definition	Explanation
1	□(On fire → ○ Done)	현재 상태가 on fire이면 바로 다음 상태는 무조건 Done 이다.
2	Normal→ ◇ Done	Normal 상태에서 언젠가는 Done이 된다.
3	□(Done→ ○ Done)	현재가 Done이면 이후는 꾸준히 Done이다.
4	□(Normal→ ○ (Normal∨ On fire))	현재가 정상 상태이면 다음상태는 정상이거나 on fire 이다.

정상적인 화재는 작성된 TL 규칙에 따라 상태 전이를 할 것이며 공격자가 랜덤하게 발생시키는 상태전이는 제안된 TL룰에 의해 필터링 될 것이다.

6. 통계기반 랭킹 알고리즘

공격자는 센서 노드의 취약점을 이용하여 다양한 형태의 공격을 감행할 수 있다. 따라서 단순한 패턴의 상태

전이를 반복하는 탐지하기 쉬운 형태의 공격에서 나아가 TL 규칙을 거스르지 않는 비정상 행동을 보일 수 있다. 이러한 공격은 기존의 TL 규칙 만으로는 탐지할 수 없다. 따라서 제안기법에서는 이러한 지능적인 공격을 막기 위해 TL 규칙에 진행의 경과 시간 별 상태변이 확률에 대한 ranking을 적용하여 공격 탐지율을 향상시켰다.

시뮬레이션으로 만들어진 경과 시간 별 각 노드의 상태값과 상태전이 확률을 측정하고 이에 대한 정보를 BS에 저장한다. 이러한 보안 모델을 통해 공격 의심 노드를 탐지할 수 있다.

BS는 각 노드의 단위시간별 상태 전이 값과 상태전이에 대한 ranking 값을 가지므로 해당 정보를 통해 실시간으로 들어오는 각 노드의 보고서 데이터를 비교하여 노드의 공격을 탐지한다

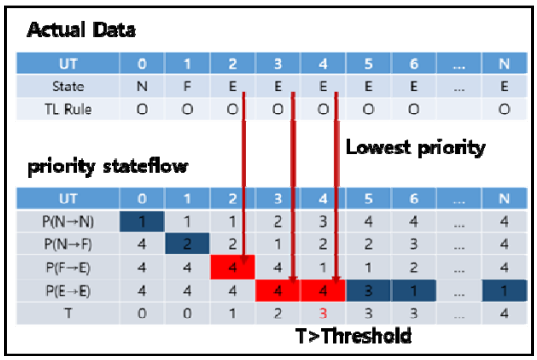


그림 2 랭킹 기반 알고리즘 동작과정

그림 2에서 보여주는 시나리오는 ranking 기반의 비정상 노드 상태변이의 확률이 가장 낮은 동작을 진행을 보여주며, UT가 4일 때 T값이 설정한 임계 값보다 커지게 되어 해당 노드는 공격 노드로 간주된다.

제안기법에서는 TL 규칙을 거스르는 동작은 개별적인 임계 값을 설정하지 않고 즉시 공격 노드로 간주한다.

제안기법에서는 각 단위시간마다 센서 노드가 실제로 보이는 상태변이와 시뮬레이션 기반의 상태변이에 대한 ranking, TL룰을 비교하여 정상적인 상태 변이인지 공격인지를 탐지한다. 또한 통계기반의 탐지는 정상적인 상태변이에 대한 공격으로의 판단 실수가 있을 수 있으므로 임계 값을 설정할 필요가 있다. 만일, 너무 낮은 임계 값을 설정한다면 정상 노드가 보이는 비정상 행동을 너무 빨리 공격으로 간주하여, 해당 노드를 더 이상 사용하지 못해 이벤트 탐지율이 낮아질 수 있기 때문이다. 5장과 6장에서 소개된 두가지의 추가적인 단계를 통해 광범위한 공격의 허위보고서를 탐지할 수 있다. 하지만 이 알고리즘은 BS에서 추가적인 오버헤드를 가진다. BS는 센서 노드와 다르게 자원이 한정적이지 않으며 처리 성능이 뛰어남으로 오버헤드에 대한 부담을 허용할 수 있다. 또한 본 제안 기법을 통해 얻어진 결과를 참고하여 클러스터의 상황에 맞는 보안 강도를 재설정할 수 있다.

7. 결론

본 논문에서는 기존의 응용계층 보안 프로토콜을 통해 보안을 할 수 없는 광범위한 공격 상황에서 TL 규칙과 시뮬레이션을 통해 통계 모델과 랭킹 알고리즘을 적용해 보고서의 내용을 기반으로 상황인식과 보안을 달성하는 방법에 대해 소개하였다. 제안 방법은 보고서의 검증 요소들이 모두 훼손된 상황에서 기존의 보안 프로토콜들이 갖는 공통적인 취약점을 보완하였으며, 중간 노드들의 추가적인 에너지 소모가 불필요하다는 장점이 있다. 본 제안 기법은 데이터에 민감한 응용 환경인 재난 상황, 전장 상황 등과 같은 다른 상황의 모니터링을 위해 WSN 필드에도 적절한 TL룰과 통계 수치를 적용할 수 있다는 장점이 있다.

ACKKNOWLEDGEMENT

이 논문은 2019 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.NRF-2018R1D1A1B07048961)

참고문헌

[1] Ye, Fan, et al. "Statistical en-route filtering of injected false data in sensor networks." *IEEE Journal on Selected Areas in Communications* 23.4 (2005): 839-850.

[2] Yu, Zhen, and Yong Guan. "A dynamic en-route scheme for filtering false data injection in wireless sensor networks." *Proceedings of the 3rd international conference on Embedded networked sensor systems*. ACM, 2005.

[3] Liu, Zhixiong, et al. "A Cluster-Based False Data Filtering Scheme in Wireless Sensor Networks." *Adhoc & Sensor Wireless Networks* 23 (2014).

[4] Fisher, Michael. *An introduction to practical formal methods using temporal logic*. Vol. 82. Hoboken, NJ: Wiley, 2011.

[5] Kumar, Alok, and Alwyn Roshan Pais. "En-route filtering techniques in wireless sensor networks: a survey." *Wireless Personal Communications* 96.1 (2017): 697-739.

[6] Nam, Su Man, and Tae Ho Cho. "Context-aware architecture for probabilistic voting-based filtering scheme in sensor networks." *IEEE Transactions on Mobile Computing* 16.10 (2016): 2751-2763.

A Model Projection Technique for Compositional Verification using Model Checking

Dong-Ah Lee[O], Junbeom Yoo
Department of Computer Science and Engineering, Konkuk University
{ldalove, jbyoo}@konkuk.ac.kr

모델 체킹을 활용한 구성 검증을 위한 모델 투영 기법

이동아[O], 유준범
건국대학교, 컴퓨터공학과

Summary

Model checking of large-scale software is hardly possible because the software is too complex and large. Model checking of a component which is a small part of the software is acceptable to avoid the state explosion problem where the component is not huge and too complex. This paper introduces a novel technique, called model projection, for compositional verification using model checking. The model projection is an activity to identify proper parts of a whole system with respect to verification purposes in order to apply model checking to the large-scale software.

1. Introduction

Model checking is one of formal verification techniques, which checks that a model meets its specification with the aid of automated support [1]. The technique is possible to model a system at any level of the system hierarchically, and to make properties along the level of the system hierarchy. The checking, however, restricts the model not only to have a finite number of states but also to have the size and complexity which a checking algorithm is able to search state space of the model exhaustively. The state explosion problem [2] may arise, as a model is too huge and complex.

Verification at a component level using model checking is acceptable to avoid the state explosion problem where a single component is not huge and complex. Model checking of a component checks that the component meets local properties at the same level of detail. Just because results of the checking show that a model of the component satisfies the properties does not mean that the component correctly works in the global system. It depends on how the component affects the whole system. Furthermore, model checking compositionally requires analysis of relations and influence between components at a system level in order to assert that the results are valid at the system level.

This paper introduces a novel technique, called *model projection*, for compositional verification using model checking techniques. The model projection is an activity to identify proper parts of a whole system with respect to verification purposes. The technique projects small models (formal models at a component-level) from a large one (a formal model at a system-level) in order to verify the proper parts using model checking. The model projection reveals relations and influences between the identified parts and the whole system.

2. The model projection technique

The model projection is a technical process to identify relevant parts simulating a formal model at a system-level with scenarios derived from a verification requirement. <Fig. 1> describes a brief process of the technique. Modeling FM_s (formal model at a system-level) refers software requirement or design specifications and generating S_i (i_{th} simulation scenario) refers verification requirements. Because of the size and complexity, the FM_s is usually much abstracted, which does not have specific information. Simulation of FM_s with S_i identifies P_i (parts projected by the i_{th} simulation scenario) which are running parts of the system with respect to the S_i.

The P_i is a target for the compositional verification using model checking. Traceability analysis assists identifying specific information for modeling with specific information. The analysis traces the P_i to relevant parts of source codes, C_i. Finally, we have a formal model, fm_i,

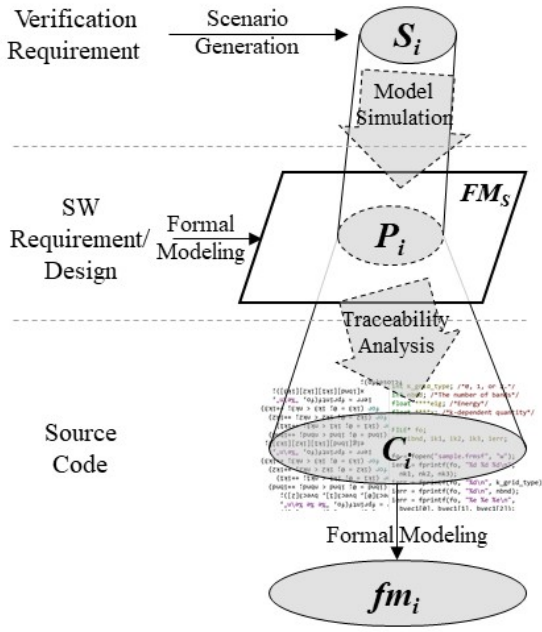


Figure 1. The model projection by simulation

which includes running parts by simulation of FM_s with S_i. Verification properties for model checking of fm_i can be both local properties of P_i and global properties of S_i. The model checking of fm_i is another phase of the compositional verification, which requires effort on selection of a model checker, property specification, analysis of its results and so on. Because the model checking itself is not a main idea in the paper, we omitted the process in detail.

The one of easiest ways to project the parts (P_i) is to model the FM_s with a tool which can model and simulate a system at the system-level, for instance, Statemate [3]. Statemate is a reasonable tool for modeling and verifying large and complex system at a system-level, since it is possible to model hierarchies of the system using various types of graphical languages, such as a Module-chart, an Activity-chart, a Flow-chart, a State-chart, etc. Statemate indicates active parts (Pi) through coloring the active parts during a simulation. The indicating active parts are a starting point of traceability analysis to identify relevant parts in source codes (C_i) with a simulation scenario (S_i).

3. Case study

Qplus-AIR [4] is a real-time operating system complying the ARINC 653 specification [5], which ETRI developed for avionics. Formal modeling and simulation of Qplus-AIR uses Statemate and the modeling refers a software design specification (SDS) of Qplus-AIR. <Fig. 2> shows an overall model of Qplus-AIR at a system-level of the operating system.

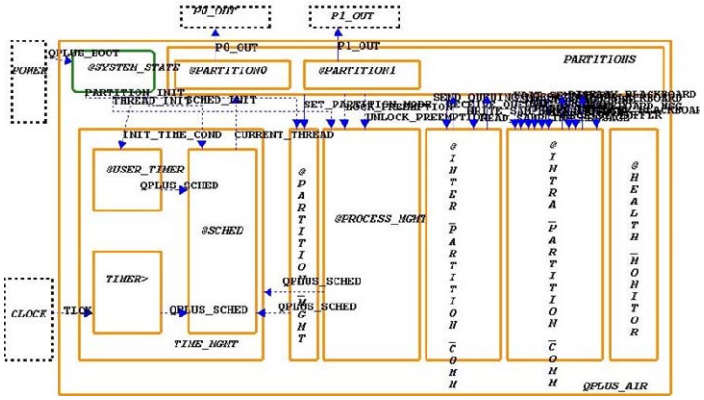


Figure 2. The formal model of a system level of Qplus-AIR with Statemate

Generation of simulation scenarios refers ARINC 653 specification, as the Qplus-AIR should comply requirements in the specification. Simulation and compositional verification in this paper focus on transitions of partition modes—the partition is one of key services in ARINC 653 to execute one or more avionics applications independently. There are 4 operating modes and transitions as described in <Fig. 3>.

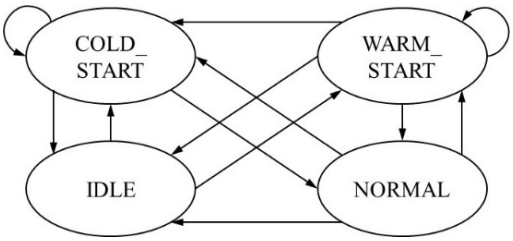


Figure 3. Partition modes and transitions in ARINC 653

Simulation scenarios only include the transitions between modes. Results of the simulation indicated small parts of the model color activated charts in violet. We found 9 functions and one major variable are related with the simulation scenarios through traceability analysis with the SDS, the formal model with Statemate, and source codes.

Model checking uses CBMC [6], which is a bounded model checker for programs written in C or C++ programming language. CBMC is available for Qplus-AIR because it is written in the C. We put an assumption statement and an assertion statement in source code to check the possibility that the mode transition from IDLE to NORMAL. The assumption means pre-condition of a partition's mode, which is IDLE in the case. The assertion

specifies post-condition of the mode and a return value for the transition function, NORMAL and no error. The assertion statement in the source code below:

```
assert((partition_mode == NORMAL)
       && (return == QPLUS_ERRNO_OK));
```

If the conditions in the assertion evaluate to true, then the mode transition from IDLE to NORMAL is possible. <Fig. 4> shows a screen dump of the verification result whether the conditions evaluate to true or false. “VERIFICATION SUCCESSFUL (no false evaluation)” at the bottom of the result means that IDLE to NORMAL without errors is possible in source code of Qplus-AIR.

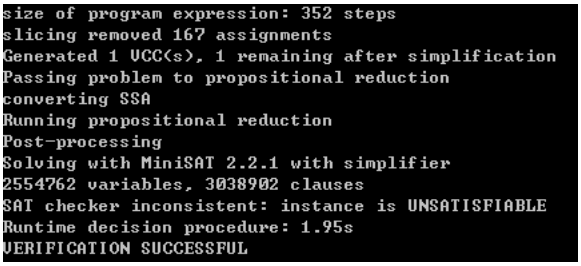


Figure 4. The verification result of a mode transition from IDLE to NORMAL by CBMC

Not only software development documents of Qplus-AIR but also ARINC 653 does not specify anything about the result. ARINC 653 does not give a result of the transition, while other invalid requests, such as a transition from COLD_START to WARM_START, are in requirements. Nevertheless, scheduling a partition without initialization is not normal behavior, and undefined mode transitions should not be allowed even if they are not possible to occur in current situation. It must be only possible within a strong assumption that the operating system, applications, or any kind of components in aircraft do not request the API, SET_PARTITION_MODE, to change a partition’s mode to NORMAL when it is IDLE.

4. Conclusions

The paper introduced the model projection technique to identify relevant parts with verification purposes. Compositional verification without systematic analysis about relation and influence between components at a system level is easy to lose confidence that results of the verification are still valid at the system level. The model projection identifies the relations and influence through simulation of a formal model of a whole system and reveals the relevant parts.

We are now developing an elaborate model and a tool for traceability analysis to make projection much easier and quicker. Since the traceability in hand requires a large amount of effort on finding traces between documents, models, and source codes. If all information is modeled properly, tracing between the information requires less effort. Furthermore, an assistant tool we plan may let the traceability analysis be semi-automatic or full-automatic.

Acknowledgement

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation (NRF) of Korea funded by the Ministry of Science, ICT (NRF-2017M3C4A7066479) and Basic Science Research Program through NRF funded by the Ministry of Education (NRF-2017R1D1A1B0303 0065).

References

[1] E. Clarke, *et al.*, Model Checking, MIT Press, 2018.
[2] E. Clarke, *et al.*, Progress on the state explosion problem in model checking., Informatics, pp. 176–194, 2001.
[3] David Harel, *et al.*, Statemate: A working environment for the development of complex reactive systems, IEEE Transactions on software engineering, Vol. 16, No. 4, pp. 403-414, 1990.
[4] T. Kim, *et al.*, Qplus/Esto-AIR: DO-178B Level A Certified RTOS and IDE for Supporting ARINC 653, Communications of the Korean Institute of Information Scientists and Engineers, Vol. 30 No. 9, 65-70, 2012.
[5] Airlines Electronic Engineering Committee, Avionics Application Software Standard Interface ARINC Specification 653 Part 1, Aeronautical Radio Inc., 2006.
[6] E. Clarke, D. Kroening, and F. Lerda, “A Tool for Checking ANSI-C Programs,” Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004), LNCS 2988, 168-176, 2004.

Javadoc 대상 테스트 요구사항 추출 기법의 정확도 평가

김지웅 홍신
한동대학교 전산전자공학부
{jeewoong, hongshin}@handong.edu

Evaluation of Test Requirement Extraction Techniques for Javadoc Description

Jeewoong Kim Shin Hong
School of Computer Science and Electrical Engineering, Handong Global University

요약

본 논문은 Javadoc과 같은 주석으로부터 테스트 오라클(test oracle)의 요구사항 명세를 추출하는 tComment, Toradocu, Jdoctor 기법을 오픈소스 프로젝트인 JFreeChart 내 총 360개 메소드에 적용하여 테스트 요구사항 기법의 두 가지 정확도 척도, 즉 실제 Javadoc 문서에서 테스트 요구사항을 내포하는 구문을 얼만큼 정확히 인식하는지, 인식한 요구사항 내포 구문으로부터 요구사항 명세를 얼만큼 정확히 추출하는 지 실험적으로 평가하는 사례연구를 수행했다. 실험 평가 결과, 세 기법은 실제 테스트 요구사항 명세를 포함하는 Javadoc 구문 중 43.2%에 대해서 정확하게 요구사항을 추출함을 확인하였다. 반면, 56.8%에 대해서는 요구사항 인식이나 정확한 추출에 실패함을 확인하였는데, 특히 이러한 요구사항에는 추상적, 복합적, 범위표현, 타입검사에 대한 특징이 있음을 구체적으로 파악할 수 있었다.

1. 서론

퍼징(fuzzing)[1], 심볼릭 테스팅(symbolic execution)[2] 등 테스트 입력을 자동으로 생성하는 기술이 실제 소프트웨어 개발 현장에서 널리 적용됨에 따라, 테스트 오라클(test oracle), 즉 테스트 실행 결과를 요구사항에 비추어 자동으로 검사하기 위한 명세의 자동 생성에 대한 기술적 수요가 증가하고 있다. 테스트 오라클 자동 생성 기술은, 높은 수요에도 불구하고, 현재까지 매우 한정적인 기법만 개발된 상황이며, 테스트 입력 생성 기술에 비하여, 발전 속도도 제한적인 수준이다[3]. 테스트 입력 생성 기법의 경우, 기존에 개발되어 온 정적/동적 프로그램 분석 기법을 활용해 테스트 대상 프로그램 자체에 내포된 구조 정보와 의미 정보를 테스트 입력 생성에 효과적으로 활용할 수 있는 반면, 테스트 오라클 생성에는 테스트 대상 코드의 직접적 활용이 어려우며, 테스트 대상 코드와 별개로, 개발자가 의도한 테스트 요구사항을 추정하는 새로운 형태의 분석 기법이 요청되는데 상황이다.

테스트 오라클 자동 생성에 활용도가 높을 것으로 기대되는 새로운 프로그램 분석 기법으로 개발자가 작성한 Javadoc과 같은 자연어 주석으로부터 테스트 요구사항을 추출하는 기법이 개발되고 있다[4][5][6]. 개발자가 자연어로 기술한 코드 주석은 테스트 대상 프로그램 코드와는 별개로 개발자가 의도한 코드의 기능과 요구사항에 대한 고차원적 설명이 내포되어 있다. 또한, 테스트 대상 프로그램에는 결함이 존재하더라도, 주석에는 해당 코드의 정확한 동작에 대한 정보가 존재할 수 있으므로, 테스트 실행을 검사하는 올바른 조건을 발견할 수 있는 정보를 내포하는 경우가 많다. 코드 주석으로부터 테스트 요구사항이 정확하게 파악되고 추출된다면 자동 테스트 생성[7][8], 테스트 케이스 추가 및 향상에 다양하게 활용될 수 있을 것으로 기대된다.

본 연구는 정부의 재원으로 한국연구재단의 지원을 받아 수행되었음.
(NRF-2017R1C1B1008159, NRF-2017M3C4A7068179)

코드 주석으로부터 테스트 오라클에 활용할 수 있는 요구사항 명세를 자동으로 추출하는 기술은 다음의 두 가지 기능의 정확하고 효과적인 수행이 필요하다: (1) **요구사항 구문 식별**: 어떠한 구문이 테스트 요구사항에 대한 정보를 포함하고 있는지를 확실하게 식별하는 기능과 (2) **요구사항 추출**: 식별된 구문으로부터 해당 요구사항 명세를 정확하게 추출(번역)하는 기능이 필요하다.

코드 주석으로부터 요구사항을 추출하는 기법에 대한 대부분의 연구[4][5][6]는, 각 기술이 집중하는 요구사항 구문 패턴에 한정하여, 해당 패턴의 요구사항에 대한 보다 정확한 추출에 보다 집중하고 있다. 반면, Javadoc과 같은 실제 코드 구문에 어떠한 종류의 요구사항이 존재하며, 현재 제안된 기법들이 이들을 포괄적으로 식별하는 지에 대한 평가는 매우 제한적인 실정이다.

예를 들어, Javadoc을 대상으로 한 요구사항 추출 기법인 tComment[4]와 Toradocu[5]는 파라미터로 `null`이 주어지는 경우와 Exception발생에 대한 요구사항 패턴에 특화되어 있으며, Jdoctor[6] 역시 파라미터의 값 비교 등 제한된 요구사항 패턴에 한정하여 요구사항을 추출한다. 반면, 실제 Javadoc에는 기존 기법이 고려하지 않은 보다 일반적인 형태의 요구사항을 내포하는 구문이 존재한다. 한 가지 예로, 아래의 주석은 JFreeChart의 `XYSeries` 클래스의 `getMinX` 메소드에 대한 주석 중 일부다:

```
/** Returns the smallest x-value in the series,
  * ignoring any Double.NaN values.
  * This method returns Double.NaN if there is
  * no smallest x-value
  * (for example, when the series is empty).
```

이 주석 구문 중 음영으로 표시한 부분은 `getMinX` 메소드가, `series`가 비어 있는 경우, `Double.NaN`을 반환해야 한다는 기능적 요구사항을 표현하고 있으며, 이는 테스트 요구사항으로 직접적으로 활용될 수 있다. 반면, 앞서 언급한

세 가지 기법 모두 메소드 서술 영역(description block)에 작성된 요구사항을 추출하지 못하였으며, 이와 같이 Javadoc에는 다양한 형태로 요구사항이 존재함에도 현재까지 제안된 기술들은 요구사항을 추출하기에는 한계가 존재한다. 현재까지 제안된 기술을 테스트 오라클 생성 기술 요소로 기반하여 새로운 기법을 개발하기 위해서는, 현재까지 제안된 요구사항 추출 기술의 정밀도(precision)는 물론 요구사항 구문 식별 기술의 재현도(recall)에 대한 구체적이고 객관적인 평가가 필요하다.

본 논문은 실제 오픈소스 Java 프로젝트의 Javadoc 주석구문을 대상으로 최근에 개발된 세 가지 기법 tComment, Toradocu, Jdoctor를 적용한 후, 수작업으로 파악한 Javadoc 요구사항 명세에 관련된 구문 중 얼마만큼 정확하고 포괄적으로 탐지하는 지, 또한 추출(번역)된 명세가 정답(ground truth)과 얼마만큼 일치하는 지를 평가하였다. 오픈소스 Java 프로젝트인 JFreeChart의 총 360개 메소드 주석(총 11개 클래스)을 대상으로 한 평가 결과, 총 250개의 요구사항 명세 구문 중, 3개 기법이 총합 111개를 올바르게 식별(탐지)하였으며, 이 중 108개는 정확하게 요구사항이 추출되었다. 본 논문은 3개 기법이 탐지하지 못한 139개 요구사항의 특징(4.1절), 3개 기법이 정확하게 탐지한 111개와 정확하게 추출한 108개 요구사항 구문의 종류를 분석한 결과(3.2절)를 소개한 후, 향후 연구 주제를 논의한다(4.3절).

2. Javadoc으로부터 요구사항을 추출하는 기법

본 연구의 주제가 되는 tComment[4], Toradocu[5], Jdoctor [6]는 Javadoc의 특정 패턴의 구문을 탐지한 후, 특정한 규칙에 따라 요구사항을 추출한다. 본 장에서는 각 기법이 추출하는 Javadoc 구문 대상의 특징과 추출하는 요구사항의 특징, 그리고 동작 방식을 개관한다.

1) tComment. tComment[4]는 메소드에 대한 Javadoc 주석 중 특정 파라미터 P에 대한 `@param` 태그 주석 구문과 특정 Exception E에 대한 `@throws` 주석 구문에 `null`이 포함된 경우를 탐지한 후, 특정 파라미터에 `null`값이 들어올 경우 특정 Exception을 발생(throw) 시켜야 한다는 형태의 요구사항을 "P == null => E" 형태 탬플릿에 맞추어 추출한다. 예를 들어, 한 메소드에 "`@param key the key null not permitted`"와 "`@throws IllegalArgumentException if key is null`"가 주석으로 기재되어 있을 경우, tComment는 "key==null => IllegalArgumentException"을 추출하는데, 이 요구사항 명세는 해당 메소드 호출 시 `key` 파라미터에 `null`값이 올 경우, `IllegalArgumentException`이 반드시 발생해야 한다는 요구사항을 뜻한다. 추가로, `@param` 혹은 `@throws`태그 주석에 "`null`"과 "`not`"이 동시에 나타난 경우에는 "key != null"을 Exception의 조건식으로 하여 요구사항을 추출한다.

2) Toradocu. Toradocu[5]는 메소드에 대한 Javadoc 주석 중 `@throws` 또는 `@exception` 주석 구문에 특정 어휘적(lexical) 패턴이 일치할 때 특정한 패턴의 요구사항을 생성하도록 정의된 규칙에 따라 작동한다. Toradocu가 탐지하는 구문 패턴과 요구사항을 추출하는 템플릿은 다음과 같다:

- Exception E에 대한 `@throws` 혹은 `@exception` 구문에 "X is/are positive" 형태의 표현이 나오고 X에 어휘적으로 직접 대응되거나 어휘적으로 유사한 변수/메소드 X'가 있을 경우 "X' > 0 => E"을 추출함. "X is/are negative" 경우도 대칭적으로 "X' < 0 => E"을 추출함.
- Exception E에 대한 `@throws` 혹은 `@exception` 구문에 "X is/are < 1" 형태의 표현이 나오고 X에 어휘적으로 직접 대응되거나 어휘적으로 유사한 변수/메소드 X'가 있을 경우 "X' < 1 => E"을, "X <= 0"의 경우는 "X' <= 0 => E"을 추출함.
- Exception E에 대한 `@throws` 혹은 `@exception` 구문에 "X is/are true" 형태의 표현이 나오고 X에 어휘적으로 직접 대응되거나 어휘적으로 유사한 변수/메소드 X'가 있을 경우, "X' == true => E"을 추출함. "X is/are false"의 경우는 "X' == false => E", "X is/are null"의 경우는 "X == null => E"를 추출함.

예를 들어, "`@throws IndexOutOfBoundsException if the dataset is empty`"라는 구문이 Javadoc에 있다면, "is empty"와 가장 유사한 이름의 메소드인 "`isEmpty()`"를 대응시켜서 "dataset.isEmpty() == true => E"를 요구사항으로 추출한다.

3) Jdoctor. Jdoctor[6]는, Toradocu와 유사하게, 메소드에 대해 작성된 Javadoc 주석 중 `@param`, `@return`, `@throws` 태그로 작성된 구문에 대해 정의된 구문 탐지 패턴, 요구사항 추출 템플릿에 의해 요구사항을 추출한다. Jdoctor가 사용하는 요구사항 추출 규칙은 Toradocu에 정의된 규칙들과 관련 연구[9][10]에 정의된 규칙을 활용한다. 다만, Jdoctor는, Toradocu 그리고 ALICS[9]와는 달리, 구문 패턴 인식에 어휘적(lexical) 패턴 매칭을 사용하는 대신, 어휘를 Word2Vec 방식으로 임베딩한 후, 구문패턴과의 거리(distance)가 가까운지를 판별하는 방식으로 구문패턴을 탐지한다.

앞서 Toradocu에서 소개한 것 외에 Jdoctor가 관련연구 [9][10]로부터 차용하는 추출 규칙은 다음과 같다:

- 태그 구문에 "X is an instance of T"라는 표현이 있고 X에 특정 객체 X'를, T에 특정 클래스 명 T'을 대응할 수 있을 경우"(X' instanceof T') == true" 형태의 요구사항을 추출한다.
- 태그 구문에 "X and Y is smaller than V"가 있고, X와 Y에 멤버 변수 X'과 Y'이 대응되고 상수 V가 대응될 경우, "X' < V && Y' < V"를 요구사항으로 추출한다. "X or Y is smaller than V"도 대응적으로 "X' < V || Y' < V"로 추출한다.
- 태그 구문에 "X is in range [V1, V2]" 혹은 X is between V1 and V2"가 있을 경우 X, V1, V2 각각에 대응되는 변수 X'에 대하여 "V1 <= X' && X' <= V2"를 생성한다.
- 태그 구문에 "X is one of S"라는 표현이 있고 X에 객체 X'가 대응되고 S에 Set 타입 객체 S'이 대응 될 경우, "S'.contains(X') == true"를 요구사항으로 추출한다.
- 태그 구문에 "X is out of bounds of A"라는 표현이 있고 X에 객체 X'가 대응되고 A에 Array 객체 A'이 대응 될 경우, "(0 <= X' && X' <= A'.length) == false"를 추출한다.

이와 같이 현재까지 제시된 문서로부터 요구사항을 추출하는 기법은 메소드 단위로 태그가 명시된 Javadoc 구문에 대해 이미 정의된 특정한 구문 패턴을 탐지한 후 이미 정의된 요구사항 템플릿에 맞는 파라미터를 유추하여 요구사항 명세를 생성하도록 가능함을 알 수 있었다.

3. 실험 설계 및 결과

3.1. 실험 설계

1) 연구 질문. 본 연구에서는 tComment, Toradocu, Jdoctor가 실제 JavaDoc의 테스트 요구사항을 얼마만큼 정확히 추출하는지 확인하기 위하여 각 기법을 실제 오픈소스 프로젝트 JFreeChart에 적용하여 다음 세 가지 연구 질문을 탐구하는 실험을 진행하였다:

- RQ1. Javadoc 주석으로 실제로 테스트 요구사항 명세가 포함된 구문 중 얼마에 대하여 각 기법이 테스트

<표1. 실험에 사용한 기법 적용 대상 프로그램>

Class Name	LOC	# method	# JavaDoc Stmt	# Req Stmt
CategoryLabelPositions	408	11	48 (25)	14 (14)
ChartColor	189	2	9 (4)	3 (3)
DefaultKeyedValues	474	22	81 (40)	25 (24)
GrayPaintScale	237	10	42 (17)	8 (6)
HistogramDataset	514	21	111 (56)	33 (30)
StackedXYAreaRenderer	696	16	101 (52)	12 (10)
Marker	696	37	139 (48)	23 (23)
PiePlot	3722	145	689 (231)	68 (52)
Range	478	23	99 (49)	18 (11)
TimeSeries	1433	58	295 (110)	38 (32)
XYBlockRenderer	448	15	74 (28)	8 (8)
Average	845	32.72	153.45	22.72

요구사항을 추출(구문 패턴이 탐지)하는 가?

- RQ2. 각 기법이 테스트 요구사항을 추출한 Javadoc 구문(구문 패턴이 탐지한 구문)은 실제 테스트 요구사항이 기술된 구문인가?
- RQ3. 각 기법이 Javadoc 문서에서 추출한 테스트 요구사항은 실제 요구사항을 올바르게 기술하는 가?

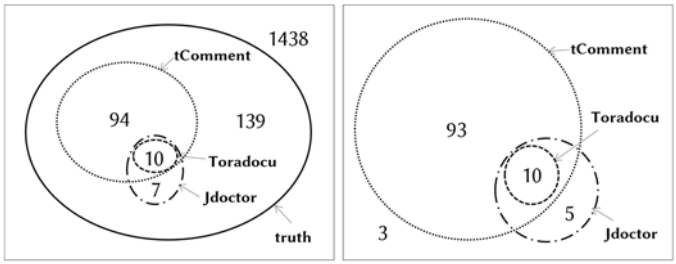
2) 테스트 요구사항 추출 도구. 본 연구에서는 Java 프로그램에 대해 동작하는 기법으로서 tComment, Toradocu, Jdoctor의 공개 버전 도구를 사용했다.

3) 기법 적용 대상 프로그램 및 정답 데이터 준비. 본 실험에서는 그래프, 표 등 데이터를 그래픽으로 표현하는데 사용하는 오픈소스 Java 프로젝트인 JFreeChart 버전 1.0.19 중 11개 클래스의 메소드 별 JavaDoc 문서(총 360개 메소드)를 선택하여 테스트 요구사항 추출 대상 프로그램으로 사용했다. 11개 클래스는 JFreeChart에 존재하는 총 629 클래스 중 JavaDoc 문서가 충실히 작성된 40개를 일차 선별한 후, 서로 특징이 유사한 클래스를 가능한 배제하고 다양한 특징을 갖도록 검토하여 정하였다.

적용 대상으로 정한 11개 클래스의 메소드 별 Javadoc 주석을 수작업으로 검토하여 (1) Javadoc 문서를 구문/문장별로 구분했고, 그 후 (2) 유닛 테스트의 테스트 오라클로 작성이 필요한 테스트 요구사항 구문을 판별하여 RQ1과 RQ2에 대한 실험에 필요한 정답 데이터(ground truth)를 구했다. RQ3의 경우, 정답 데이터를 미리 작성하지 않고, 대신 실험 중 어떤 기법이 특정 구문으로부터 추출한 테스트 요구사항 명세를 추출하였을 때, 수작업으로 검토를 통해 해당 명세가 해당 구문에 내포된 테스트 요구사항의 일부 혹은 전부를 표현하는지 여부를 판별하는 방식으로 진행하였다.

표1은 실험에 사용한 11개 클래스의 이름(Class Name), 코드 라인 수(LOC), 메소드 개수(# method), Javadoc 구문(문장) 수(# JavaDoc stmt)와 그리고 정답 작성 시 파악한 요구사항을 포함하는 Javadoc 구문 수(# Req Stmt)를 나타낸다(총 250개). 괄호 안의 숫자는 Javadoc 주석 중 태그(e.g., `@param`, `@return`, `@throws`)가 있는 주석에 해당하는 수를 나타낸다.

4) 실험 수행 및 측정. 각 기법에 대하여 총 11개 적용 대상 클래스를 각각 입력하여 결과를 얻은 후, 이를 정답과 비교하여 연구 질문에 대한 답을 얻었다[1]. RQ1에 대해서는, 정답에서 실제 테스트 요구사항을 포함하는 것으로 파악한 총



(a) 각 기법이 탐지한 요구사항 포함 구문 (b) 각 기법이 요구사항을 정확히 추출한 구문

250 구문 중 각 기법이 요구사항 명세를 1개 이상 추출한

<그림 1. 세 기법의 Javadoc 주석의 요구사항 탐지 및 추출 결과>

구문의 비율(포괄도; recall)을 측정했다. RQ2에 대해서는 각 기법이 추출한 요구사항 명세 중 정답 데이터로부터 실제 테스트 요구사항을 포함하는 것으로 파악된 구문의 비율(정밀도; precision)을 측정하였다. RQ3에 대해서는 각 기법이 특정 Javadoc 구문으로부터 추출한 테스트 요구사항 중 실제 해당 구문의 요구사항의 일부 혹은 전부를 표현하는 것으로 판별된 요구사항의 비율을 측정하였다.

3.2. 실험 결과

1) RQ1 결과. 그림 1(a)는 11개 적용 대상 클래스에서 파악된 1688개의 Javadoc 구문 중 정답으로 정의한 요구사항을 포함하는 총 250개 Javadoc 구문(truth 영역)과 각 기법이 요구사항을 추출한(탐지한) 구문의 관계를 나타낸다. 그림 1(a) 각 숫자는 해당 영역에 속한 구문의 개수이다.

그림 1(a)의 결과에서 확인할 수 있듯이, 총 250개 탐지 대상 중 총 139개 구문(55.6%)에 대해서는 세 기법 모두 요구사항을 탐지하지 못했다. 나머지 111개 구문에 대해서는 tComment, Toradocu, Jdoctor 중 하나 이상의 도구가 요구사항을 탐지하였으며, 이 때 tComment는 포괄도 37.6%(=94/250), Toradocu는 포괄도 4%(=10/250), Jdoctor는 포괄도 7%(=17/250)를 보였다.

2) RQ2 결과. 그림 1(a)에서 알 수 있듯, tComment, Toradocu, Jdoctor가 식별한 요구사항 구문은 모두 정답(truth) 영역 내에 있으므로, 세 기법 모두 정밀도 100%이며 잘못된 탐지는 한 건도 없었다. RQ1과 RQ2의 결과를 볼 때, 본 연구의 대상인 세 가지 요구사항 추출 기법은 한정적인 형태의 요구사항 패턴에 집중하여 높은 정밀도를 보임을 알 수 있다.

3) RQ3 결과. 그림 1(b)는 세 기법이 추출한 요구사항(총 111개) 중 실제 Javadoc 구문의 요구사항을 올바르게 표현한 수를 나타낸다. 그림 1(a)와 비교하여 볼 때, tComment가 추출한 104개 요구사항 중 103개가 정확함을 알 수 있다(추출된 요구사항의 정밀도 99%). Toradocu는 추출한 요구사항 10개가 모두 정확하였다(정밀도 100%). Jdoctor의 경우, 추출한 17개 중 15개가 정확하였으며 2개에는 오류가 있었다(정밀도 88%).

4. 논의

4.1. 연구대상 기법이 식별/추출하지 못한 요구사항

1) 모든 기법이 식별(탐지)에 실패한 요구사항. 그림 1(a)에서 알 수 있듯이 총 250개의 요구사항 포함 구문 중 139개는 tComment, Toradocu, Jdoctor 모두 요구사항 탐지는

[1] Jdoctor의 경우, [6]과 공개버전의 설정이 달라, [6]의 설정을 따랐다.

물론 추출에도 실패하였다. 이 139개 요구사항 포함 구문을 분석한 결과 다음과 같이 분류할 수 있었다:

A. 태그(예: `@param`, `@return`, `@throws`)가 아닌 비정형적 주석 구문에 작성된 요구사항 처리에 대한 한계. 표 1에서 나타난 바와 같이 태그로 시작하지 않는 37개 구문에도 테스트 요구사항이 표현되어 있었으나, 세 기법 모두 태그 구문에 한정되어, 이들을 탐지하지 못하였다. 태그 구문에 등장하는 요구사항 구문은 비교적 정형적인 특성을 갖고 있으나, 태그 외에 나타나는 요구사항은 비정형적이므로 이에 대한 포괄적 처리가 부족하다.

B. 특정한 값(상수)에 관한 요구사항에 대한 한계. 어떤 파라미터나 변수가 특정한 값으로 지정되어 있는 지에 대한 조건, 혹은 특정 변수에 저장되어 있던 값이 변경되었는지 확인하는 요구사항의 경우, 탐지되지 않았다.

C. 값 범위에 대한 다양한 표현으로 기술된 요구사항. 실제 개발자가 작성하는 주석에는 값 범위(range)를 기술하기 위해 특정 상수를 지칭하거나, 세 가지 도구들이 탐지하는 것 외에 다양한 표현(예: max, min)을 사용하여 표현하고 있다. 하지만 이들에 대해 세 기법 모두 요구사항을 탐지하지 못하였다.

D. 복합적 조건을 포함하는 요구사항. 값 범위와 부등식을 동시에 사용하는 복합적 조건의 경우, 세 기법을 통해 요구사항으로 탐지되지 않았다.

E. 절차적 요구사항. 실행 순서를 기술한 요구사항의 경우, 탐지되지 않았다. 이러한 절차적 요구사항은 세 가지 기법이 표현하는 요구사항의 종류(method annotation)으로 표현이 간단하지 않을 것으로 보인다.

F. 이미 등장한 요구사항을 인용하여 기술하는 요구사항. 하나의 Javadoc 구문에 한정되지 않고, 여러 구문에 걸쳐서 있는 요구사항을 인용하거나 대명사로 앞서 기술한 요구사항을 지칭하는 경우, 세 기법으로 탐지가 불가능하였다.

G. 그 외 구현 상 제약점. 총 15개 구문은 `private`메소드의 주석에 존재하였으나, tComment는 `public`, `protected` 메소드에 대해서, Toradocu, Jdoctor는 `public` 메소드에 대해서만 요구사항을 추출하여 이들을 탐지하지 못했다. 또한 타입 정보로 명시되어 있지 않은 런타임 Exception(예: `ClassCastException`)의 발생 조건, 처리 방식에 관련된 요구사항이 올바르게 탐지되지 않았다.

2) 식별하였으나 정확한 추출이 실패한 요구사항. RQ3결과에서 설명한 바와 같이, tComment는 1개 구문에 대해, Jdoctor는 2개 구문에 대해 잘못된 요구사항을 추출한 경우가 있었다. 이를 사례를 관찰한 결과, (1) tComment는 구문에 “not”이라는 단어가 다른 단어에 일부로(예: “annotation”) 있는 경우를 잘못 인식해서 1개의 오류가 발생하였으며, (2) Jdoctor는 요구사항 구문에 있는 술어(예: “at least”)를 정확히 번역하지 못해 2건의 오류가 발생했다. 즉, 3건 모두, tComment와 Jdoctor의 구문 패턴 종류가 제한적이어서 발생한 오류였다.

4.2. 연구대상 기법 간의 상호 비교

세 기법들이 공통적으로 탐지한 10개의 요구사항은 Javadoc의 `@throws` 주석에 `null`과 관련하여 주어와 서술어가 명시적으로 나타난 경우였다. 반면, `null`과 관련하여 tComment만 고유하게 추출한 경우가 93건 있었는데, (1) 주어 또는 서술어가 생략된 경우, (2) 괄호와 같이 복잡한 형태를 가진 구문의 경우가 대표적이다. Jdoctor만 고유하게 추출한 5개의 요구사항은 `null` 조건 검사 이외의 조건식 형태(예: 대소 비교)를 갖는 요구사항이었다.

4.3. 요구사항 추출 기법 발전을 위한 연구방향 제안

본 연구의 실험 결과로 비추어 볼 때, 기존의 요구사항 추출 기법들은 제한된 특정한 패턴에 대해서 요구사항 명세를 명확히 추출하는 방향으로 개발되었으며(예: `null` 관련), 반면 실제 Javadoc 주석에 내포된 요구사항의 50% 정도는 기존 기법이 집중하는 패턴과 맞지 않는 문제가 있음을 알 수 있었다. 이를 개선하기 위해서는 (1) 새로운 요구사항 추출 규칙 정의는 물론 (2) 자연어 처리 기술을 통해 포괄적인 구문 패턴 매칭이 가능하도록 개선하는 연구가 필요함을 확인할 수 있었다. 특별히, 실제 개발자가 사용하는 표현의 다양성을 고려할 때, Javadoc 주석에서 빈번히 발생하는 요구사항에 대한 구문의 패턴을 통계적으로 식별할 수 있도록 데이터 기반의 연구가 요청됨을 확인할 수 있었다.

4.4. 논문에서 다루지 않은 관련 기법

ALICS[9]은 품사(Part-Of-Speech) 태그[11]와 프로그래밍 언어의 특징이 반영된 사전(dictionary)을 정의하여(예: `null`은 명사) Javadoc 구문을 정의된 탬플릿에 따라 1차 논리식(First Order Logic)으로 변환한다. ALICS[9]은 `String` 클래스, `Integer` 클래스, `null`값 검사, `@return`, `@throws` 태그에 대하여 미리 정의된 패턴에 따라 1차 논리식을 요구사항으로 추출한다. ALICS[9]은 도구가 공개되었으나, 프로그램 실행 문서 부재로 본 연구에서는 실험 대상으로 포함시키지 못하였다.

도구 미공개로 본 실험에서는 포함하지 못한 관련 연구에는 Zhou 등이 제안한 기법[10]이 있다. 이 기법은 Javadoc 주석 중 `@param`, `@throws` 또는 `@exception` 주석 구문에 작성된 `null`값 허용, `null`값 불허, 타입 검사, 범위 제한에 대한 4가지 종류의 요구사항을 패턴에 따라 추출하여 대상 프로그램의 동작을 테스트한다. 수작업으로 `java.awt`, `javax.swing` 두 패키지의 Javadoc 구문을 분석하여 작성한 패턴을 정의된 규칙에 따라 1차 논리식으로 변환하여 코드와 Javadoc 사이의 일치성을 SMT solver를 사용하여 검사한다.

5. 결론

본 논문은 Javadoc 주석으로부터 테스트 요구사항 명세를 추출하는 tComment, Toradocu, Jdoctor 기법을 오픈소스 프로젝트인 JFreeChart에 적용하여, 이들이 테스트 오라클 생성에 필요한 요구사항 탐지 및 추출 성능을 평가하였다. 평가 결과, 세 기법은 높은 정밀도를 보이나, 포괄도는 최대 41.6% 수준이며 여러 발전 과제가 있음을 관찰할 수 있었다.

참조문헌

[1] V. J. M. Manes et al., The Art, Science and Engineering of Fuzzing; A Survey, IEEE Transactions on Software Engineering, early access, Oct 2019

[2] C. Cadar et al., Symbolic Execution for Software Testing in Practice: Preliminary Assessment, ICSE 2011

[3] E. T. Barr, The Oracle Problem in Software Testing: A Survey, IEEE Transactions on Software Engineering, 41(5), May 2015

[4] S. H. Tan et al., @tComment: Testing Javadoc Comments to Detect Comment-Code Inconsistencies, ICST 2012

[5] A. Goffi et al., Automatic Generation of Oracles for Exceptional Behaviors, ISSTA 2016

[6] A. Blasi et al., Translating Code Comments to Procedure Specification, ISSTA 2018

[7] C. Pacheco et al., Randoop:Feedback-directed Random Testing for Java, OOPSLA 2007

[8] G. Fraser et al., EvoSuite: automatic test suite generation for object-oriented software, FSE 2011

[9] R. Pandita et al., Inferring Method Specifications from Natural Language API Descriptions, ICSE 2011

[10] Y. Zhou et al., Analyzing APIs Documentation and Code to Detect Directive Defects, ICSE 2017

[11] D. Klein et al., Accurate unlexicalized parsing, ACL 2003

임베디드 소프트웨어에 대한 테스트케이스 생성을 지원하는 분산 Concolic 테스팅 도구의 설계와 구현

최한솔, 임혜린, 김효림, 홍신
한동대학교 전산전자공학부
{hansolchoe, hyerinleem, 21600193, hongshin}@handong.edu

Design and Implementation of Distrusted Concolic Testing Tool for Embedded Software

Hansol Choe, Hyerin Leem, Hyorim Kim, Shin Hong
School of Computer Science and Electrical Engineering, Handong Global University

요약

본 논문은 임베디드 소프트웨어에 대한 Concolic 테스팅을 효과적이고 효율적으로 지원하기 위해, 임베디드 타겟(target) 시스템과 호스트(host) 시스템의 분산적이고 동시적으로 테스트 생성을 위한 작업을 수행하는 Concolic 테스팅 도구의 설계와 구현 사례를 소개한다. 소개하는 테스트 케이스 생성 도구는 Concolic 테스팅 과정 중 임베디드에 종속적인 특성을 갖는 테스트검증 대상 프로그램의 실행 부분은 임베디드 타겟 시스템에서 수행하고, 시스템에 비종속적인 실행 부분인 탐색 전략, 제약식 해법기 실행 과정은 계산성능이 좋은 호스트 시스템에 분산하고, 독립적인 단계를 동시적으로 실행하도록 기존 Concolic 도구를 개선하였다. Arm Cortex A54 아키텍쳐의 임베디드 타겟 시스템과 x86-64 아키텍쳐의 호스트 시스템을 대상으로 본 기법을 구현하여 오픈소스 C 프로그램인 Grep, Busybox Awk, Busybox Ed를 대상으로 실험한 결과, 기존 도구 보다 1.59~2.64배 테스트케이스 생성속도가 향상됨을 확인할 수 있었다.

1. 서론

산업과 생활의 다양한 영역에 임베디드 시스템이 이용됨에 따라 이를 구동하는 임베디드 소프트웨어의 오류가 사회의 안전을 위협하는 새로운 요인으로 대두되고 있으며[1], 이를 탐지하고 제거하기 위한 임베디드 소프트웨어 검증 기술의 수요가 증가하고 있다. 일반적인 소프트웨어 검증 상황과 달리, 임베디드 소프트웨어에 대한 검증은 특정 하드웨어 장치에 종속적인(비표준적) 기능과, 제약적인 하드웨어 자원 등을 필수적으로 반영하여 수행돼야 한다. 이러한 이유로, 표준적 프로그램 의미론에 기반한 기법이나 높은 계산비용이 소요되는 기존의 소프트웨어 검증 기법을 임베디드 시스템에 직접적으로 적용하기 어려우며, 이로 인해 실제 임베디드 소프트웨어 검증 과정은 비체계적 관행에 그치는 경우가 많다.

본 논문은 임베디드 소프트웨어에 대한 Concolic 테스팅을 효과적이고 효율적으로 지원하기 위해, 임베디드 타겟(target) 시스템과 호스트(host) 시스템을 분산하여 동시적으로 테스트 생성 작업을 수행하는 Concolic 테스팅 도구의 설계와 구현 사례를 소개한다.

Concolic 테스팅[2][3]은 테스트 대상 프로그램으로부터 추출한 동적 실행 경로식을 SMT 해법기로 계산하여 새로운 테스트 케이스를 생성하는 기법이다. Concolic 테스팅 기법은, Z3[4] 등 고도로 발달된 SMT 해법기를 실행 경로식 계산에 효율적으로 활용함으로써, 대규모 코드로 이루어진 소프트웨어의 테스팅 자동화에도 높은 확정성을 보이며 이용되고 있다[5][6][7]. 이와 같은 현재 Concolic 테스팅 도구는 테스트 대상 프로그램, SMT 해법기, 테스트 생성 알고리즘을 같은 시스템 상에서 수행하도록 구성되어 있는데, 이를 임베디드 소프트웨어 테스팅에 적용하기 위해서는 다음 두 가지 방식이 가능하다:

- 임베디드 시스템에 Concolic 테스팅 도구 전체를 이식한 후 실행시키는 방법: SMT 해법기 등 많은 자원이 소요되어, 제한적인 하드웨어 상황에서 테스트 생성이 불가하거나 테스트 생산성이 현저히 낮은 문제가 발생함.
- 호스트 시스템에서 테스트대상의 일부 코드 만을 가상적으로 실행: 호스트 시스템에서 임베디드 타겟 시스템을 대신하는 테스트 스텁(test stub)을 작성해 일부 코드만을 실행시킬 수 있다. 이 경우, 임베디드 시스템을 모델링하는 비용이 소요되며, 실제 하드웨어 동작과 간극이 있을 경우, 결함 검출에 실패할 수 있다.

이와 같은 기존 Concolic 테스팅 도구의 한계를 효과적으로 해결하기 위하여, 본 논문에서는 Concolic 테스팅 과정 중 (1) 하드웨어 및 임베디드 시스템에 종속적인 특성을 갖는 테스트검증 대상 프로그램의 실행 부분을 임베디드 타겟 시스템에서 수행하고, (2) 하드웨어나 시스템에 종속적이지 않게 실행이 가능한 테스트 케이스 생성 탐색전략 실행과 제약식 해법기 실행 과정은 계산성능이 좋은 호스트 시스템에 분산하고, 독립적인 단계를 동시적으로 실행하는 개선된 Concolic 테스팅 도구를 소개한다.

본 연구는 기존에 개발된 C 프로그램 대상 Concolic 테스팅 도구인 CREST를 확장하여 제안한 기법을 Arm Cortex A54 아키텍처를 갖는 임베디드 타겟 시스템과 x86-64 아키텍쳐를 갖는 호스트 시스템을 대상으로 구현하였다. 실제 오픈소스 C 프로그램을 대상으로 한 실험 결과, 본 논문이 제안한 기법은 임베디드 시스템 상에서 기존 도구 전체를 실행하는 기존

본 연구는 정부의 재원으로 소프트웨어중심대학 지원사업(2017-0-00130)과 한국연구재단의 지원(NRF-2017R1C1B1008159, NRF-2017M3C4A7068179)을 받아 수행됨.

방식 대비 초당 평균 2.14배의 테스트 케이스 생성을 수행함을 확인할 수 있었다.

2. 분산형 Concolic 테스팅 기법

본 논문은 실제 임베디드 시스템 상에서 테스트 대상 프로그램을 실행하는 동시에 탐색전략 휴리스틱, SMT 해법기 등 계산비용이 높은 Concolic 테스팅 알고리즘을 효율적으로 구동하기 위해 기존 연구에서 소개한 기본 Concolic 테스팅 알고리즘을 호스트 시스템 실행 부분과 타겟 시스템 실행 부분으로 분산하고, 독립적인 부분은 병렬화 한 알고리즘을 제안한다.

타겟(target) 시스템은 테스트 대상 프로그램이 내장되고 운영될 실제 임베디드 시스템이다. 미탐이나 오탐 없이 정확한 결함 검출이 가능한 테스팅을 위해서는 테스트 대상 프로그램를 실제 하드웨어/시스템 종속적 명령의 직접적인 실행이 가능한 타겟 시스템에서 실행하여 실제적인 실행 경로식을 추출해야 한다. 반면, 임베디드 시스템 특성 상 타겟 시스템은 속도가 느리거나 메모리 자원에 제약이 크기 때문에, 테스트 대상 프로그램 실행 이외의 추가적인 컴퓨팅/메모리 자원 사용이 최소화되어야 한다. 호스트(host) 시스템은 계산 속도가 빠르고 메모리 자원이 많은 시스템으로, 탐색 전략 알고리즘 구동이나 SMT 해법기 구동 등 하드웨어/시스템과 독립적인 연산을 맡아서 수행하게 된다.

본 논문이 제안하는 분산형 Concolic 테스팅 알고리즘은 다음의 과정을 연속적으로 수행하여 구동된다:

1. 호스트 시스템 모듈과 타겟 시스템 모듈이 동시에 실행되고 상호 간 연결을 구성한다.
2. 호스트 시스템에서 초기 테스트 케이스를 타겟 시스템 모듈에 전송한다.
3. 타겟 시스템이 전송 받은 테스트 케이스를 입력하여 테스트 대상 프로그램을 실행하여 경로조건식을 추출한다. 추출한 경로식을 호스트 시스템에 전송한다.
4. 호스트 시스템과 타겟 시스템은 동시에 각각 다음의 과정을 수행한다:
   - 호스트 시스템: 전송 받은 경로조건식을 바탕으로 탐색 전략과 SMT 해법기를 이용한 제약식 해결을 통해 새로운 테스트 케이스를 생성한다. 생성된 테스트 케이스를 타겟 시스템에 전송한다.
   - 타겟 시스템: 신규 프로세스를 생성하여 테스트 대상 프로그램을 로딩한 후 호스트 시스템으로부터 신규 테스트 케이스가 전송 되기를 기다린다.
5. 테스트 종료 조건이 만족될 때까지, 과정 3으로 돌아가서 반복적으로 테스트 케이스르 생성한다.

3. 도구 구현 및 실험

본 연구는 제시한 분산 Concolic 테스팅 기법을 C 프로그램 대상 오픈소스 Concolic 테스팅 도구인 CREST[8]를 확장하여 구현하였으며, Arm Cortex A54 아키텍처를 갖는 Raspberry Pi 3 B+ 보드를 임베디드 타겟 시스템(1.4GHz CPU, 0.9GB RAM)으로, x86-64 아키텍처를 갖는 Ubuntu 서버를 호스팅 시스템(3.3 GHz CPU, 8GB RAM)으로 사용하는 상황에서 실험을 수행했다.

제안한 분산형 Concolic 테스팅 기법을 구현하기 위해 CREST를 수정/확장한 기술적 요소는 다음과 같다:

- **경로조건식 인코딩 통일**: 임베디드 타겟 시스템 상의 경로조건식 도출 단계에서 숫자 값이 아키텍처에 독립적으로 표현되도록 인코딩을 일관되게 통일함

<표1. 테스트 케이스 생성에 소요되는 시간(초)>

	DFS		Uniform Random		Random	
	CRS	CDX	CRS	CDX	CRS	CDX
Grep	190.2 (15.38)	77.0 (39.0)	126.2 (23.8)	50.3 (59.6)	177.9 (16.9)	67.1 (44.7)
Awk	169.77 (29.5)	106.5 (46.9)	161.6 (30.9)	101.5 (49.3)	194.8 (25.7)	117.3 (42.6)
Ed	121.0 (41.3)	59.6 (83.9)	125.8 (39.7)	59.9 (83.5)	151.5 (33.0)	59.3 (84.3)

- **테스트 대상 프로그램 실행 모듈과 테스트 케이스 생성 모듈 분할**: 기존 CREST의 테스트 케이스 생성 모듈 중 테스트 대상 프로그램을 실행시키는 부분은 임베디드 타겟 시스템에서 실행되는 프로그램으로, 탐색 전략 및 SMT 해법기를 통한 테스트 케이스 생성 부분은 호스트 시스템에서 실행되는 프로그램으로 분할함.
- **테스트 프로그램 실행을 위한 프로세스 생성과 테스트 케이스 생성의 병렬화**: 호스트 시스템에서 테스트 케이스를 생성하는 작업(예: SMT 해법기 실행)을 수행함과 동시에 임베디드 타겟 시스템에서는 테스트 대상 프로그램 실행을 위해 프로세스 생성(i.e. `fork()`)을 수행하도록 하여 Concolic 테스팅 수행의 동시성을 향상함.

표1은 CREST(CRS열)와 CREST-DX(CDX열)가 3개의 테스트 대상 프로그램에 대하여 3종의 탐색 전략 휴리스틱(DFS, Uniform Random, Random)[9]을 이용하여 오픈소스 유닉스 유틸리티인 Grep에 대해 3000개, Awk과 Ed에 대해 각각 5000개 테스트 케이스의 생성하는 데 소요된 시간이다. 각 칸의 괄호 안의 수는 1초당 테스트 케이스 생성 개수로 환산한 값이다. 표1의 결과, 본 연구가 제안한 기법(CDX)는 기존 CREST를 그대로 적용한 방법 대비 최소 1.59배(Awk, DFS)에서 최대 2.64배(Grep, Random) 테스트 케이스 생성 속도의 향상을 달성하였음을 확인할 수 있다.

4. 결론

본 논문은 임베디드 소프트웨어의 효과적이고 효율적인 테스팅을 임베디드 타겟 시스템과 호스트 시스템의 분산적 Concolic 테스팅 기법을 제안한다. 제안한 기법을 CREST 도구를 확장하여 구현한 결과, 테스트 생성 속도가 1.59~2.64배 향상됨을 확인할 수 있었다. 향후에는 제안한 기법을 MCU 등 보다 제약적인 임베디드 시스템 환경에 적용하고, Concolic 테스팅의 탐색 전략 과정에서도 병렬화를 통해 보다 효율성을 향상하는 방안에 대해 연구할 계획이다.

참조문헌

[1] N. Zhang et al., Understanding IoT Security Through the Data Crystal Ball: Where We Are Now and Where We Are Going to Be, CoRR, vol. abs. 1703.09809, 2017

[2] K. Sen, D. Marinov, G. Agha, CUTE: a concolic unit testing engine for C, ESEC/FSE, 2005

[3] C. Cadar, et al., KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs, USENIX OSDI, 2008

[4] The Z3 Theorem Prover, https://github.com/Z3Prover/z3

[5] Y. Kim, Y. Kim, T. Kim, G. Lee, Y. Jang, M. Kim, Automated Unit Testing of Large Industrial Embedded Software Using Concolic Testing, ASE, 2013

[6] Y. Park, S. Hong, M. Kim, D. Lee, J. Cho, Systematic Testing of Reactive Software with Non-deterministic Events: A Case Study on LG Electric Oven, ICSE, 2015

[7] Y. Kim et al., Concolic Testing for High Test Coverage and Reduced Human Effort in Automotive Industry, ICSE, 2019

[8] J. Burnim, CREST, https://github.com/jburnim/crest

[9] J. Burnim and K. Sen, Heuristics for Scalable Dynamic Test Generation. ASE 2008

STPA를 이용한 군집 운행 시스템의 안전성 분석 사례 연구

김의섭○, 유준범
건국대학교 컴퓨터공학과
atang34@konkuk.ac.kr, jbyoo@konkuk.ac.kr

A Study on Application of STPA in Safety Analysis of Platoon System

Eui-Sub Kim, Junbeom Yoo
Division of Computer Science and Engineering

요 약

본 연구는 군집 운행 시스템에 STPA를 이용하여 안전성 분석을 수행한 결과를 기술하고자 한다. 군집 운행 시스템에서의 사고는 사람의 생명을 위협하고 큰 재산의 손실을 유발하기 때문에, 반드시 안전성 분석을 통해 발생할 수 있는 다양한 사고 원인과 위협원을 사전에 식별하고, 이를 제거 및 감소시키는 노력이 필요하다. 하지만 군집 운행 시스템의 복잡성과 복합적인 상호 운용성은 기존의 단순 컴포넌트 수준의 안전성 분석 기법으로는 위협원을 식별하기 쉽지 않다. 본 논문에서는 시스템 수준의 통합적인 관점을 가지고 있는 STPA 방법을 이용해 해당 시스템을 적용 과정과 이를 통해 식별한 위협 시나리오의 예를 기술하고자 한다.

1. 서 론

차량 간 통신 커뮤니케이션의 발전과 자율주행 기술의 발전으로 인해 군집주행 기술이 다양한 곳에서 연구 개발 중이다. 군집 주행이란 두 대 이상의 차량이 일정한 차량 간격을 유지하며 하나의 그룹을 이루어 주행하는 것을 의미한다. 군집 주행을 통해 앞차와의 거리를 줄여 공기저항을 줄이고, 이를 통해 연비 개선을 도모할 수 있다. 또한, 자율운전으로 인해 운전자의 피로도를 줄이고, 졸음 및 부주의한 운전으로부터 생길 수 있는 사고를 미연에 방지할 수 있다[1, 2, 3].

군집 운행 시스템은 다양한 이점이 있는 반면, 해당 시스템에서의 사고는 사람의 생명을 직접적으로 위협하며 재산상의 큰 손실을 유발하는 안전 필수 시스템이기 때문에, 반드시 안전성 분석을 통해 발생할 수 있는 다양한 사고 원인과 위협원을 식별하고, 사전에 제거 및 감소시키는 노력이 필요하다. 하지만 군집 운행 시스템의 복잡성과 상호 운용성은 개발 시 의도하지 않은 오동작을 유발시킬 수 있다. 따라서 개발 설계 단계부터 안전성 분석을 통해 안전 요구사항을 식별하고 이를 개발에 반영하도록 하는 안전성 분석 기반 개발이 수행되어야 한다. 이러한 시스템을 분석하기 위해서는 단일 컴포넌트 수준이 아닌 전체 시스템 수준에서 분석을 수행하는 것이 중요하다.

본 논문에서는 시스템 이론 기반의 안전성 분석 기법인 STPA (System-Theoretic Process Analysis) [4, 5, 6]를 이용해 군집 운행 시스템의 안전성 분석을 수행하고, 이를 통해 밝혀낸 시스템의 위협 시나리오와 STPA를 군집 운행 시스템과 같은 동적 변화가 심한 시스템의 적용에 어려움에 대해 기술한다.

2. STPA (System-Theoretic Process Analysis)

STPA는 기존 안전성 분석 기법들의 개념인 '컴포넌트의 실패' 보다는 '시스템 또는 컴포넌트 간 제어 문제'에 의해 사고가 발생한다고 보고, 시스템을 컨트롤 관계를 중심으로 구조화하여 위험 분석을 수행한다. 따라서 단순 컴포넌트의 나열 후 분석을 수행하기 보다 안전에 영향을 미치는 컨트롤을 식별하고 이를 중심으로 시스템을 추상화하여 분석함으로써 보다 규모가 크고 복잡한 시스템을 분석하는데 용의한 분석 방법이다.

STPA의 단계는 그림 1과 같이 크게 4단계로 구성된다.
1) Define Purpose of the Analysis: Losses와 System-level hazards, system-level constraints를 식별한다
2) Model the Control Structure: 시스템을 안전에 영향을 주는 컨트롤러와 컨트롤, 피드백을 중심으로 Control Structure를 작성한다.
3) Identify Unsafe Control Actions: 4가지 유형에 따른 Unsafe Control Action을 도출한다. 4가지 유형은 다음과 같다. ⓐ Control action is not provided, ⓑ Unsafe Control Action is provided, ⓒ Control Action is too early, too late, out of sequence, ⓓ Control

Action is stopped too soon, applied too long.
4) Identify Loss Scenarios: 3에서 식별된 Unsafe Control Action의 발생 원인을 도출한다. 해당 단계에서는 위험을 유할 할 수 있는 Unsafe Control Action이 왜 발생하게 되었는지를 분석함으로써 시스템의 위험요소를 식별하고 시스템의 안전성을 확보하는데 기여할 수 있다.

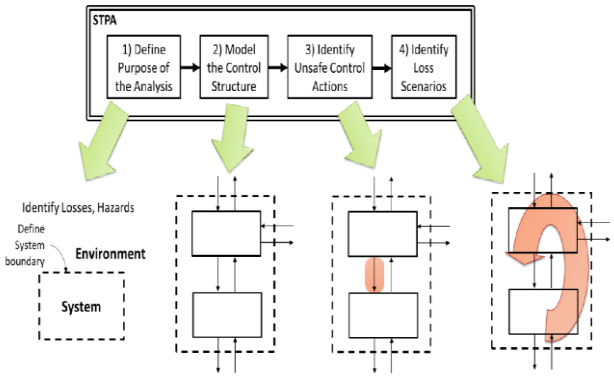


그림 1. Overview of the basic STPA Method [4]

3. 군집 운행 시스템의 안전성 분석

3.1 타겟 시스템

타겟으로 하고 있는 군집 운행 시스템의 주요 기능은 다음과 같다.

1) 군집 합류: 군집 외 차량이 군집의 끝 또는 중간에 합류하는 기능. 합류하고자 하는 차량의 운전자로부터 합류 요청을 받아 리더의 허락 하에 합류가 이루어진다. 합류 후에는 리더는 해당 차량의 정보를 저장하고, 합류 차량은 자율 주행 모드로 전환된다.
2) 군집 운행 유지: 리더가 군집의 속도를 제어하는 기능. 리더의 운전자는 운전을 직접 수행하며, 리더의 운전자에 의해 변경된 속도를 군집 내 모든 차량에게 전파하여 군집의 속도를 전체적으로 제어한다. 이를 통해 보다 전체적인 속도 제어를 할 수 있어 효율적인 군집 운행이 가능하다.
3) 군집 이탈: 군집 내 차량(리더 포함)이 군집을 이탈하는 기능. 이탈하고자 하는 차량의 운전자로부터 해당 요청이 이루어지며, 리더의 허락 하에 이탈이 수행된다. 이탈 후에는 자율주행 모드에서 수동 주행 모드로 변경되며, 리더의 경우 리더의 뒤 차량에게 리더를 인계하는 과정이 수반된다.
4) 자율 주행: 군집 내 차량들이 스스로 차간거리를 유지하며 군집 운행이 이루어지도록 제어하는 기능이다. 해당 시스템의 아키텍처는 그림 2와 같다. 해당 시스템은 차량내 위치하게 되며 외부와의 V2X 커뮤니케이션과 차량 내 Sensor, 운전자로부터 입력을 받아 차량 내 ECU에게 명령을 전송한다. 크게 군집 운행을 관리하는 Platooning Controller와 자율 주행을 담당하는 Adaptive Cruise Controller로 구성되어 있다.

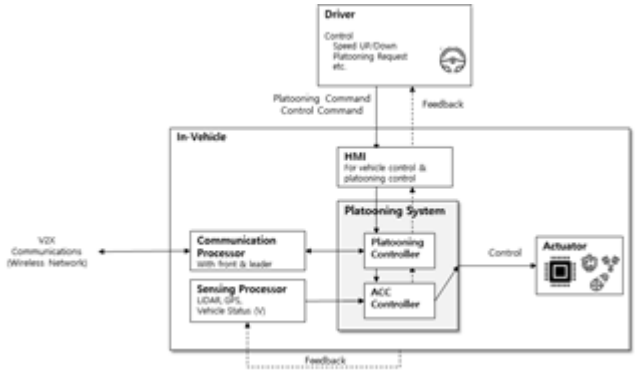


그림 2 군집 운행 시스템의 아키텍처

3.2 STPA 적용

3.2.1 Define Purpose of the Analysis

타겟 시스템의 accident와 hazard는 아래와 같고, 개략적인 내용은 기존 논문들[7, 8]을 참고 하여 작성하였다.

A-1) people die or become injured
A-2) damage to vehicle
A-3) fail of the platooning
H-1) Vehicles collide with each other
H-1.1) inappropriate distance interval with front vehicle
H-1.2) inappropriate distance interval with back vehicle
H-1.3) fail to measure distance surrounding vehicles
H-2) Vehicles dash to pedestrian
H-3) fail to communication between vehicles

3.2.2 Model the Control Structure

Platoon 시스템에서 차간거리 유지를 컨트롤하는 컨트롤러인 Adaptive Cruise Controller (ACC)를 중심으로 작성된 Control Structure는 그림 3과 같다.

3.2.3 Identify Unsafe Control Actions

작성한 Control Structure를 바탕으로 Unsafe Control Action에 대해 4가지 Type으로 작성을 진행하였다 <표 1>. 표를 살펴보면, ACC는 앞차와의 거리가 Safe 차간거리보다 가까울 경우 감속 Control Action을 제공해야 한다. 이를 제공하지 않을 경우 앞차량과의 추돌로 이어지기 때문에 사고가 발생할 수 있다. 해당 경우는 제공되어야 할 Control Action이 제공되지 않아 사고가 발생할 수 있는 Unsafe Control Actions으로 식별을 진행할 수 있다. 마찬가지로 앞차와의 차간거리가 Safe 차간거리보다 가까울 경우, 가속 Control Action이 발생한다면 H.1.1과 연관된 Unsafe

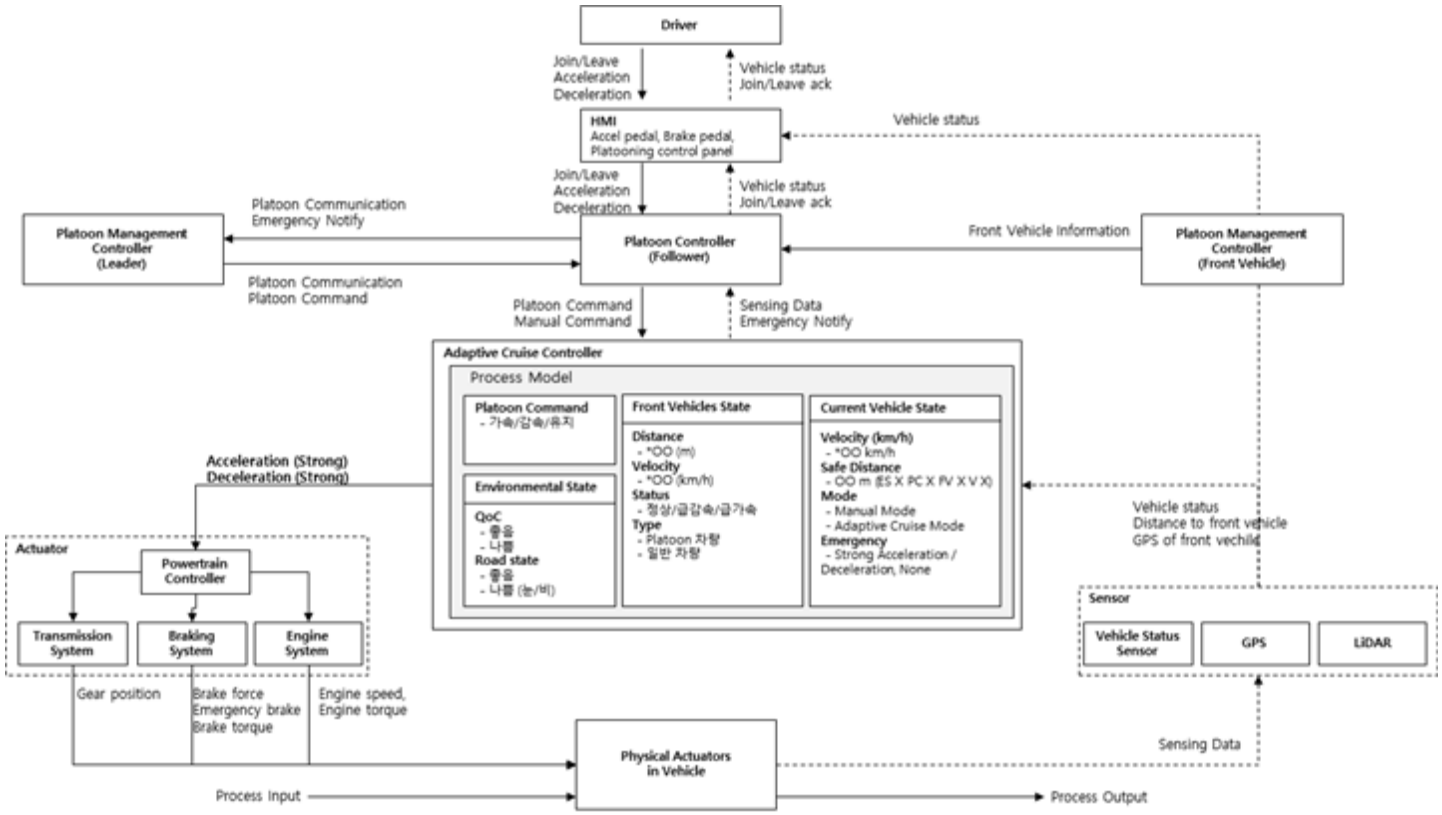


그림 3 Platoon 시스템의 Control Structure

Control Action으로 분석할 수 있다.

3.2.4 Identify Loss Scenarios

마지막으로 3.2.3에서 분석한 Unsafe Control Action이 왜 발생하게 되었는지 분석한다. 크게 두가지 유형으로 분류를 할 수 있다. 1) 왜 Control Action이 Unsafe하게 제공되었는가? 2) Control Action이 왜 부적절하게 수행되거나 제대로 수행되지 못했는가? 해당 분석을 바탕으로 Causal Scenario를 작성해 낼 수 있다. 표1의 Unsafe control action 중 하나인 “Adaptive Cruise Controller가 앞차와의 거리가 Safe 차간거리보다 가까울 때, Deceleration을 제공하지 않는다.”에 대해서 식별한 Causal Scenario는 다음과 같다.

o (Controller receives incorrect feedback/information) 앞차와의 거리가 Safe 차간거리보다 가까울 때, 산악지대라 앞차와의 통신 상태가 좋지 못해, 앞차의 속도 및 GPS 값을 수신하지 못해서, Safe distance 계산을 실패하였다. 따라서 Deceleration을 제공하지 않는다.

o (Controller receives correct feedback/information but interprets it incorrectly or ignores it) 앞차와의 거리가 Safe 차간거리보다 가까울 때, Cut in 차량과 Leave 차량이 동시에 있을 경우, Cut in 차량을 기준으로 safe distance를 측정해야 하는데, Leave 차량을 고려하여 safe distance를 계산 Distance가 충분히 멀다고 판단하여, Deceleration을 제공하지 않는다.

o (Controller does not receive feedback/information when needed (delayed or never received)) 앞차와의 거리가 Safe 차간거리보다 가까울 때, CAN의 연결되어 있는 다른 장비의 통신 과사용으로 인해, Sensing 정보가 제대 들어오지 못해, Safety distance를 계산하여, Distance가 충분히 멀다고 판단, Deceleration을 제공하지 않는다.

o (Feedback/info sent by sensor(s) but not received by controller) 앞차와의 차간거리가 Safe 차간거리보다 짧을 때, LiDAR 센서의 정보가 CAN의 보안 공격으로 인해 정확한 센싱 값을 수신하지 못함 Safe 차간거리가 충분한 것으로 잘못 판단 내림, 따라서 Deceleration을 제공하지 않았다.

o (Control action is not applied or received by the controlled process but the process responds as if the control action had been applied or received) 앞차와의 차간거리가 Safe 차간거리보다 짧을 때, Automatic Cruise controller는 아무것도 송신하지 않았지만, 타 시스템 (후방 차량 감지 시스템)의 영향으로 Acceleration을 수행함, Deceleration을 수행하지 않음 & Acceleration을 수행함 (Manual vs. Automatic)

또한, “Automatic Cruise Controller가 앞차와의 거리가 Safe 차간거리보다 가까울 때, Deceleration을 너무 늦게 제공하였다.”에 대해서도 원인 분석을 해보면, 앞차의 급정거가 Sensor가 반응하기에 또는 Physical

표 1 Unsafe Control Action 도출

Unsafe Control Actions	Not providing causes hazard	Providing causes hazard	Too soon, too late, out of sequence	Stopped too soon, applied too long
Deceleration	Adaptive Cruise Controller가 **앞차와의 거리**가 **Safe 차간거리**보다 가까울 때, **Deceleration**을 제공하지 않는다. [H1] Adaptive Cruise Controller가 **앞차가 급 감속 중** 일 때, **Deceleration**을 제공하지 않는다. [H1] Adaptive Cruise Controller가 **앞 공간에 Join 요청이 있을 경우에**, **Deceleration**을 제공하지 않는다. [H1]	Adaptive Cruise Controller가 **Driver가 감속 명령**을 했을 때, **Deceleration**을 제공하지 않는다. [H1]	Automatic Cruise Controller가 **앞차와의 거리**가 **Safe 차간거리**보다 가까울 때, **Deceleration**을 너무 늦게 제공하였다. [H1] Automatic Cruise Controller가 **앞차가 급 감속 중** 일 때, **Deceleration**을 너무 늦게 제공하였다. [H1]	
Acceleration	Adaptive Cruise Controller가 **Driver가 가속 명령**을 했을 때, **Acceleration**을 제공하지 않는다. [H1]	Automatic Cruise Controller가 **앞차와의 거리**가 **안전 차간거리**보다 가까울 때, **Acceleration**을 제공한다. [H1] Automatic Cruise Controller가 **앞차가 급 감속 중** 일 때, **Acceleration**을 제공한다. [H1]		

Process가 반응하기에 너무 빠른 경우도 원인으로 분석할 수 있다. 또한, 앞차의 앞차의 급정거가 전파되어 차간거리를 유지하는 Control Action이 Unsafe Control Action이 되는 원인으로 작용할 수 있다. 이럴 경우 앞차와의 간격을 센서만으로 식별하기 보다 급정거에 대한 정보를 리더 차량에 전달하여 급정거 후방의 차들이 적절한 대응을 할 수 있도록 안전 요구상항을 만들어 시스템에 반영하는 것이 안전성을 보장하는데 도움이 될 것으로 판단 내릴 수 있다

본 논문에서는 차간거리와 관련된 Hazard를 중심으로 STPA를 분석을 진행하였다. 하지만 A-3) fail of the platooning 과 같이 차간거리가 아닌 다른 컨트롤에 대해 분석하려면 해당 컨트롤과 관련 있는 Control Structure를 다시 작성하여 분석을 수행해야 하는 번거로움이 있다. 또한, 군집 운행과 같이 다양한 조합과 다양한 상황이 발생하는 경우 각각의 상황을 모두 계획하여 다시 분석해야 하는 어려움이 있다. 이런 가변적인 모든 상황을 고려하여 분석하는 것은 시간과 노력 측면에서 매우 비효율적인 모습을 가질 수 있다. 따라서 가변적으로 조합되어 상호 운용되는 시스템에 대해서 효과적으로 표현하고 분석하는 추가적인 방안이 필요할 것이다.

4. 결 론

본 논문에서는 군집 운행 시스템에 대해 STPA를 이용하여 안전성 분석을 수행한 과정과 일부의 결과를 기술하였다. 안전성 분석의 결과로 Loss Scenarios까지 도출한 결과의 일부를 도출하였으나, STPA의 분석 시 군집 운행과 같은 동적으로 다변하고 조합되어 상호 운용되는 시스템을 분석하기에 제한적인 모습이 있다는 것도 알 수 있었으며, 추후 이런 제한적인 부분에 대해 연구하여 효과적으로 군집 운행과 같은 사이버-피지컬 시스템을 분석하는데 도움이 되는 연구를 진행 하고자 한다.

사 사

이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업 (NRF-2017M3C4A7066479)의 지원을 받아 수행한 연구임.

참고 문헌

[1] Kit, Michal, et al. “An architecture framework for experimentations with self-adaptive cyber-physical systems.” Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. IEEE Press, 2015.

[2] Muccini, Henry, Mohammad Sharaf, and Danny Weyns. “Self-adaptation for cyber-physical systems: a systematic literature review.” Proceedings of the 11th international symposium on software engineering for adaptive and self-managing systems. ACM, 2016.

[3] Bergenhem, Carl, et al. “Overview of platooning systems.” Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria. 2012.

[4] Nancy G. Leveson, “Engineering a Safer World.” MIT Press (MA), 2011.

[5] Nancy G. Leveson, “STPA: A new hazard analysis technique.” MIT Press, 2012.

[6] Leveson, Nancy G., and John P. Thomas. “STPA handbook.” Cambridge, MA, USA, 2018.

[7] Abdulkhaleq, Asim, and Stefan Wagner. “Experiences with applying STPA to software-intensive systems in the automotive domain.” 2018 3rd International Conference on System Reliability and Safety (ICSRS), 2013.

[8] Stoltz-Sundnes, Max, “STPA-Inspired Safety Analysis of Driver-Vehicle Interaction in Cooperative Driving Automation”, KTH, School of Industrial Engineering and Management (ITM), TRITA-ITM-EX, 2019:687, 2019.

시장의 리드 타임에 충족하는 속도와 품질 확보를 위한 코드분석 기반의 배포프로세스 수립과 적용사례

안선희, 김진태 [0]
소프트웨어공학엑스퍼트그룹㈜
{sunnyan, jtkim}@swexpertgroup.com

A Case Study on Establishment of Release Process based on Code Analysis for Time-to-Market and High Quality

An Sun Hee, Kim Jin Tae [0]
Software Engineering Expert Group

요 약

최근 다양한 환경과 빠른 시장 점유를 위해 소프트웨어의 배포 속도가 점차 빨라지고 배포 관리 또한 복잡해지고 있다. 이로 인해 배포 속도에 비해 상대적으로 배포되는 산출물의 품질은 충분히 확보되지 못한 채 고객에게 전달되는 문제가 발생하고 있다. 본 논문은 배포의 속도와 배포 산출물의 품질을 확보할 수 있는 코드 분석 기반 배포 프로세스를 수립하고 실제 기업의 적용 사례를 통한 연구를 다룬다. 우리의 사례는 버전관리 도구의 Commit Log 분석을 통해 자주 변경되는 소스파일을 식별하고 배포 시 해당 소스파일의 품질을 확보할 수 있는 배포프로세스 수립에 대해 설명한다. 또한, 실제 기업의 적용 사례를 통해 소스코드 분석 중심의 배포 프로세스의 적합성을 설명한다.

1. 서 론

비즈니스에 있어서 지속적으로 고객을 만족시키는 것은 기업의 존폐와도 연관될 정도로 매우 중요한 부분이다. 고객을 만족시키기 위한 중요한 요소는 고객에게 전달되는 제품 혹은 서비스의 최종 품질과 전달되는 속도이다. 배포 속도와 배포 산출물의 품질 향상에 관한 문제를 해결하기 위해 다양한 연구와 사례들이 존재한다[1~4].

IT 분야는 최근 10년간 몇 가지 큰 변화가 있었다. 모바일 환경으로의 패러다임 변화[5], 애자일[6]과 DevOps의 보편화[7], 클라우드 인프라를 이용한 개발[8] 등이 바로 그것이다. 이런 변화는 SW의 배포 속도를 더욱 빠르게 이끌었으며, 또한 다양한 환경에 적합한 산출물을 배포하는 결과로 이어졌다. 다시 말해 IT 분야의 큰 변화가 배포의 복잡성을 증가시키는 결과를 초래했다.

배포가 복잡해지다보니 일단 빠르게 배포하는 것 즉, 배포의 속도에만 목표를 두는 경향이 높아졌다. 이런 경향은 곧 최종 산출물의 품질이 적절히 확보되지 않은 채 고객에게 전달이 되고, 고객으로부터 결함의 보고, 재배포의 사이클 반복으로 이어졌다. 이런 악순환의 사이클에 고객의 만족도는 점차 낮아져 비즈니스에 영향을 미치게 된다.

이런 악순환의 고리를 끊기 위한 다양한 연구가 진행되고 있다. 배포 산출물의 품질 확보 관점으로는 SDLC(SW Development Cycle)의 각 단계인 요구사항 분석[9], 설계[10], 개발[11], 테스트[12] 각 단계별로 최종산출물의 품질을 확보하기 위한 연구가 진행되고 있다. 배포의 속도 측면으로는 전체 프로젝트 수행 관점에서 산출물을 확보하기 위한 프로세스 수립에 관련된 연구가 진행되고 있다[1~4].

배포 대상의 품질과 속도에 대한 연구는 Agile 개발방법론에서 이야기하는 CI(Continuous Integration), CD(Continuous Deploy, Continuous Delivery)에 대한 연구로 특정 방법론에 국한되어 그 연구가 진행되고 있다.

본 논문에서는 배포 대상의 품질을 보장하면서 빠른 배포 속도를 유지할 수 있도록 하는 두 가지 관점을 모두 포괄하는 연구를 수행하였다. 수행한 연구는 기존 대비 세 가지 차별점이 있다.

첫째, 배포프로세스 수립을 위해 배포되는 산출물의 근원인 소스코드의 품질을 분석한다.

둘째, 소스코드 품질 분석 시, 배포에 영향을 미치는 즉, 자주 변경되는 파일을 중심으로 품질을 분석한다.

셋째, 자주 변경되는 파일의 품질을 확보하기 위한 구체적인 활동과 가이드라인을 배포프로세스 수립 시

적용한다.

다시 말해, 기존에 배포프로세스 수립 시 중요하게 다루지 않았던 배포 대상 그 자체를 배포프로세스 수립의 중심에 두고, 배포의 속도는 물론 품질까지 확보하는 것이 본 연구의 차별점이다.

이에 본 논문 2장에서는 배포프로세스 수립과 관련된 선행 연구를 소개하며, 3장에서는 본 논문에서는 수행된 코드분석 기반의 배포프로세스 수립의 연구에 대한 설명과 실제 기업의 실증 사례를 소개한다. 특히 3장에서는 배포프로세스를 수립하기 위해 소스코드 분석 과정과 소스코드로부터 식별된 문제점을 어떻게 배포프로세스에서 해결할 수 있는지 그 과정을 설명하며, 4장에서는 연구 결과를 실제 기업에서의 적용 사례와 수행 결과를 소개한다. 마지막으로 5장에서는 본 연구의 결과와 향후 연구 방향에 대해 설명한다.

2. 관련연구

배포와 관련된 연구로는 크게 두 가지로 나눌 수 있는데 첫째는 배포의 속도이고, 둘째는 배포되는 산출물의 품질 확보에 대한 것이다. 먼저 배포 속도에 관한 연구는 다음과 같은 연구가 있다. 형상관리 도구의 Branch 전략과 개발 방법론을 분석하여 배포프로세스를 수립하는 방법을 제시한 연구가 있다. Branch 전략을 분석하고 배포의 복잡성을 고려하여 빠른 배포를 수행하는 것은 좋은 접근 방법이기는 하나, 이런 접근법은 최종 배포되는 산출물의 품질 향상은 결여된 채 배포를 수행하게 된다.

또 다른 연구에서는 Github와 같이 특정 형상관리 도구를 이용하여 빠른 배포 방법을 제시하기도 한다[9]. 이런 접근 방법은 Agile Practice인 CI(Continuous Integration)을 위한 방법을 How의 관점에서 다룬 것으로 이런 연구 역시 배포 대상의 품질에 대한 부분은 고려되지 않는다.

배포의 속도보다는 배포되는 산출물의 품질과 관련된 연구도 진행된다. 특히, 배포와 관련되어 많이 활용되는 것은 ITIL(IT Infrastructure Library)이다[13]. ITIL은 IT 서비스 관리에 가장 널리 사용되는 프레임워크이다. ITIL은 서비스 데스크로 접수된 인시던트(Incedent)를 처리하여 서비스를 다시 배포함으로써 IT 서비스의 품질을 확보한다. ITIL의 경우, 배포(Release) 관리를 포함하고 있는데 Release Unit이라는 단위로 배포가 관리된다. Release Unit의 단위가 ITIL을 도입한 회사별로 상이하기에 Release Unit의 품질을 확보할 수 있는 구체적인 가이드라인은 부족하다.

SDLC 각 단계별로 산출물의 품질 확보를 위한 활동의 경우, Bad Smell[9, 11] 개념을 이용한 품질 확보 방안에 관한 연구가 있다. 이는 사전에 문제로 예상되는 항목을 정의하고, 이를 활용하여 각 단계에서 지속적인 검토를 통해 문제를 해결해 나가는 방법이다. 각 단계별 활동을 통해 산출물의 품질을 높이는 활동을 수행한다. 하지만 이런 접근 방법은 특정 단계에서의 품질만을 확인할 뿐 최종적으로 동작하는 SW의 품질을 확인하는 연속적인 접근은 부족하며 특히 배포라는 주요한 활동이 누락이 된 접근법이다.

배포프로세스를 수립하기 위해 특정 기업의 상황을 파악하는 형태로 접근하는 방법도 존재한다[1, 4]. 이런 접근 방식의 경우, 배포에 활용되는 각종 문서 산출물과 배포에 관여하는 담당자들과의 인터뷰를 통해 현재의 문제점을 분석하여 개선하는 형태로 진행이 된다. 이런 접근 방법의 문제는 배포에서 가장 중요한 배포 대상인 코드 품질에 대한 다각적인 검토가 진행되지 않는다는 것이다.

3. 본 론

본 논문에서는 앞선 연구의 접근방법과는 달리 배포의 속도를 유지하면서도 산출물의 품질 확보를 고려한 프로세스 수립 방법에 대해 연구하였다. 전체적인 진행 과정은 [그림1]과 같다.

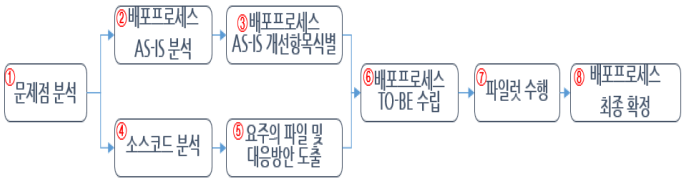


그림 1 코드분석 기반의 프로세스 수립 단계

우리가 연구한 코드 분석 기반의 배포프로세스 수립은 총 8단계로 진행되며 각 단계별로 수행 활동은 다음과 같다. ① 문제점 분석에서는 기업의 관리자 인터뷰, 관련 자산 분석, 구성원의 역량 평가를 통해 종합적인 기업의 문제점을 파악한다. ②배포프로세스 AS-IS 분석에서는 배포프로세스에 국한하여 관련된 산출물 검토, 구성원의 인터뷰를 통해 현재 기업의 배포프로세스를 가시화하는 활동을 수행하며, ③ 배포프로세스 AS-IS 개선 항목 식별에서는 문제점 분석에서 발견된 문제점과 가시화된 배포프로세스 AS-IS 분석을 통해 개선사항을 식별한다. ④소스코드 분석에서는 자주 변경되는 파일을 식별하기 위해 소스코드 Commit log를 분석하여 자주 변경되어 배포되는 파일을 식별 및 코드 품질을 분석한다. ⑤요주의 파일 및 대응방안 도출에서는 자주 변경되는 파일

중 변경 시 다른 파일에 영향을 미치는 요주의 파일을 식별하고 개발자 레벨에서 대응할 수 있는 대응방안을 도출한다. 앞서 설명한 2, 3 단계와 4, 5 단계는 분석의 성격이 상이하므로 병행하여 진행이 가능하다. 단, 배포프로세스 TO-BE 모델 수립 전까지는 앞선 단계가 종료되어야 하는 제약이 있다. 이어 진행되는 ⑥ 배포프로세스 TO-BE 수립에서는 ①에서 식별된 기업의 문제점, ③에서 식별된 배포프로세스 상의 문제점과 ⑤에서 식별된 요주의 파일을 대응할 수 있는 방법을 모두 고려하여 배포프로세스 TO-BE 모델을 수립하게 된다. ⑦파일럿 수행에서는 수립된 배포프로세스 TO-BE 모델의 적합성을 확인하기 위한 파일럿이 진행되며, 마지막 ⑧배포르로세스 최종 확정에서는 최종적으로 기업에 적합한 배포프로세스를 확정하게 된다.

이런 8 단계의 과정을 통해 배포 대상의 품질 확인은 물론 배포 대상의 품질을 확보할 수 있는 배포프로세스를 수립할 수 있게 된다.

3.1 문제점 분석 단계

문제점 분석 단계에서는 기업의 현재 문제점이 무엇인지를 분석하는 단계이다. 관리자 인터뷰를 통해 기업에 어떤 문제가 발생하고 있는지를 도출하는 활동을 수행하는데 조직 차원의 제도나 가이드의 미흡이 원인인지 아니면 구성원의 역량 차이 때문인지 확인하는 활동도 수행한다. 구성원의 역량은 소프트웨어공학 6개 영역(역량, 프로젝트관리, 개발, 지원, 인프라, 글로벌화), 15개 항목(참여자 인식 수준, 프로세스 이행 역량, 적용 기술 역량, 프로젝트 계획, 프로젝트 통제, 위험 관리, 공급업체 관리, 분석, 설계, 코드리뷰, 테스트, 요구사항 관리, 형사오간리, 품질 점검, 프로세스 인프라, 도구 인프라, 글로벌화 역량, 글로벌화 적용 준비도)에 대해 역량을 측정한다.

3.2 배포프로세스 AS-IS 분석 및 개선 항목 식별 단계

후속 단계로는 배포의 과정이 어떻게 진행되는지 관련자 인터뷰를 통해 현재의 배포프로세스 가시화 및 분석을 수행하고 개선 항목을 식별하게 된다. 먼저 배포프로세스 분석 시, 배포에 참여하는 모든 인력을 대상으로 인터뷰를 수행한다. 기업마다 서로 상이하나 RACI차트[14]를 활용하여 각 책임별 담당자를 인터뷰한다. 이 단계에서는 인터뷰로 끝나는 것이 아니라 배포의 과정을 가시화하며 진행함으로써 오해의 소지를 불식시키는 것이 바람직하다. 각 담당자와의 인터뷰가 종료되면 전체를 포괄하는 배포프로세스 맵(Map)을 작성하여 인터뷰 대상자에게 확인하고 조정하는 작업을 거쳐 최종적으로 배포프로세스 AS-IS를 도출한다.

배포프로세스 AS-IS로부터 개선 항목을 식별하는 활동은 프로세스, 도구(산출물), 담당자의 세 가지 관점으로 개선 항목을 식별할 수 있다.

3.3 소스코드 분석 및 요주의 파일대응 방안 도출

이 단계는 배포 형상의 문제점을 식별하고 품질을 확보하기 위한 단서를 수집하는 단계로 본 연구의 핵심 단계라 설명할 수 있다. 소스코드는 결함 집중의 원리, 지역성의 원리에 따라 특정 파일이 집중적으로 변경된다. 이에 착안하여 자주 변경되는 파일을 식별하고, 해당 파일의 품질을 확보하면 배포되는 형상의 품질까지 확보할 수 있다는 아이디어를 도출하였다. 아이디어를 수행하기 위해 [그림2]와 같은 워크플로우(Workflow)를 정의하였다.

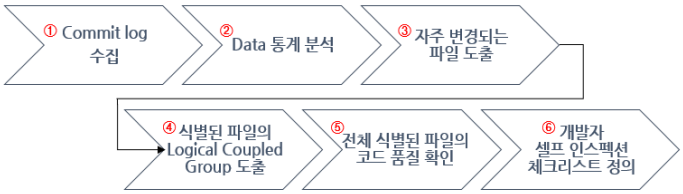


그림 2 소스코드 분석 단계의 워크플로우

배포 형상의 문제점을 식별하기 위한 워크플로우는 총 6단계로 구성된다. ①Commit log 수집은 버전관리 도구로부터 Commit log를 수집하는 단계이다. Repository가 생성된 이후 또는 특정 기간 이후로부터 소스 파일의 변화 양상을 확인할 수 있는 기초자료를 수집하는 단계이다. ②Data 통계분석은 앞 단계에서 수집한 Log data를 기반으로 다각도의 분석이 진행된다. LOC의 추이변화, 개발자의 유입과 이탈, 특정 개발자에 의해 변경되는 파일, 다수의 개발자에 의해 공통으로 변경되는 파일, 결함에 의해 변경되는 파일 등을 분석한다. 특히 이 단계에서는 1회 이상 변경된 파일들의 변경 횟수 평균과 표준편차를 도출하여 자주 변경된다는 기준을 정의하게 된다. ③자주 변경되는 파일 도출에서는 파일의 변경 빈도 수를 기반으로 전체 소스 코드에서 자주 변경되는 파일을 도출하게 된다. 이때에는 분석의 정확성을 높이기 위해 개발자의 도움을 받아 도출된 파일 중 개발자들이 사용하지 않는 파일이 포함되어 있는지 확인하는 작업을 거치는 것이 분석의 정확도를 높일 때 도움이 된다. 다음으로 진행되는 과정은 ④식별된 파일의 Logical Coupled Group[15]을 식별하는 작업이다. Logical Coupled Group은 하나의 파일이 변경될 때 같이 변경될 가능성이 높은 파일 그룹을 말한다. 자주 변경되는 파일은 배포에도 관여할

가능성이 높으므로 혹시나 같이 변경되는 파일에 영향을 주어 생각치 못한 오류가 발생하지 않도록 예방하는 활동이 필요하기에 Logical Coupled Group을 도출하는 것이 중요하다. 다수의 파일의 Logical Coupled Group을 도출하게 되면 유독 특정파일이 다수의 Logical Coupled Group에 속해 있는 것을 발견하게 되는데, 해당 파일은 집중 관리가 필요한 파일로 정의 할 수 있다.

⑤전체 식별된 파일의 코드 품질 확인은 자주 변경되는 파일, Logical Coupled Group에 대한 분석이 완료된 결과로 식별된 중점 분석 대상 파일을 정적분석 도구를 이용하여 코드 레벨의 품질을 분석하게 된다. 분석 시, 중복코드, 함수복잡도[16], CK Metrics[17], 양방향 의존관계 등의 항목을 분석하게 되는데 이 때에는 도메인 및 비즈니스, 조직 등 다양한 요소를 고려하여 분석 전략을 수립하고 이에 적합한 분석을 수행하는 것이 중요하다. 코드 품질을 측정한 결과 많은 문제점을 갖고 있는 파일의 목록을 도출할 수 있게 된다. ⑥개발자 셀프 인스펙션 체크리스트 정의는 앞서 도출된 요주의 파일의 품질을 확보하기 위해서 개발 단계에서 개발자들이 활용할 수 있는 체크리스트를 정의하게 된다. 체크리스트 항목은 구체적인 가이드라인을 두어 개발단계에서 개발자가 셀프 인스펙션을 수행할 때 활용할 수 있도록 한다.

3.4 배포프로세스 TO-BE 모델 수립

배포프로세스의 개선사항 식별과 코드 분석이 완료되면 배포프로세스 TO-BE 모델을 수립하게 된다. 배포프로세스 TO-BE 모델 수립은 여느 프로세스 수립 단계와 유사하게 진행된다. 하지만 앞서 코드 분석에서 도출된 개발자 셀프 인스펙션 체크리스트를 활용하는 것이 본 연구의 차별점이다. 개발자 셀프 인스펙션 체크리스트는 배포 형상의 품질을 확보했는지의 여부를 확인할 수 있는 주요한 산출물이다. 이에 따라 개발자 셀프 인스펙션 체크리스트는 개발단계에서 테스트 단계로의 전환 여부를 결정할 수 있는 주요한 산출물로 활용된다. 그러므로 배포프로세스 TO-BE 모델을 수립할 때에는 개발자와 관리자가 모여 셀프 인스펙션 체크리스트의 항목을 하나 하나 확인해가면서 필수, 선택 항목을 정의해야 하는 활동이 필요하다.

3.5 파일럿 수행 및 배포프로세스 최종 정의

배포프로세스 TO-BE 모델이 수립되면 기업에 적합하게 수립되었는지 확인하는 파일럿을 수행하게 된다. 파일럿 수행 전, 파일럿에 참여하는 구성원을 대상으로 공청회를 진행하여 배포프로세스 TO-BE 모델에 대한 이해도 향상과 파일럿 수행에 대한 공감대를 형성이 필요하다.

파일럿 중에는 테일러링 해야 할 Task를 도출하는 작업을 수행하며, 식별된 Task는 파일럿 종료 시 프로세스에서 조정작업을 거쳐 최종 배포프로세스로 정의한다.

3장에서는 우리가 연구한 코드 분석 기반의 배포프로세스 수립 단계별 수행활동을 설명하였으며 이어지는 4장에서는 실제 기업에 적용한 사례를 소개하고 결과를 설명한다.

4. 기업 적용 사례

우리는 모바일 오피스 솔루션을 개발하는 기업을 대상으로 코드분석 기반의 배포프로세스 수립을 적용하였다. A기업은 모바일 오피스 솔루션을 개발하는 기업으로 B2B 사업을 수년째 진행하고 있다. B2B 사업의 특성상 고객의 요청에 따라 일부 기능의 추가 혹은 삭제 등 하나의 코드에서 다양한 변경이 적용되고 배포가 이뤄졌다.

4.1 문제점 분석

문제점 분석을 위해 A기업의 관리자와 인터뷰를 통해 기업의 현황을 확인한 바 첫째, 배포를 수행하고 난 이후 일주일 내에 평균 5건의 결함이 고객으로부터 보고되고 있고, 발견된 결함을 수정하여 재배포한 이후에는 새로운 결함이 고객으로부터 보고되었다. 둘째는 특정 고객의 요청에 따라 코드 수정이 이뤄진 이후 이와 상관없는 다른 고객의 형상에서 Side effect가 발생한다는 문제가 지속적으로 발생하고 있었다. 셋째는 고객에게 배포를 수행하기 위한 조직 차원의 표준 배포프로세스가 존재하지 않다는 점을 확인하였다. 또한, 구성원에 대한 역량 평가를 수행한 결과 15개 영역 중 상대적으로 형상관리, 코드 검토, 프로젝트 계획 영역이 취약한 것을 확인하였다. 인터뷰와 역량 평가를 통합적으로 분석한 결과 [표1]과 같은 문제점을 식별하였다.

표 1 A기업의 문제점과 문제의 근원

구분	설명
문제점	1. 복잡한 배포형상관리와 불명확한 배포프로세스 2. 품질이 확보되지 않은 최종 제품의 배포
문제의 근원	1. 결함의 근원이 해결되지 않은 현상 해결 중심의 결함 제거 2. 고객별 커스터마이징되는 요구사항 검증이 미흡한 제품의 배포 3. 개발자 역량에 의존한 배포 관리 4. 다수의 고객별 배포요구사항에 따른 복잡한 배포관리

4.2 배포프로세스 AS-IS 분석 및 개선 항목 식별 단계

이후 배포의 과정이 어떻게 진행되는지 관련자 인터뷰를 통해 현재의 배포프로세스 가시화와 분석을 수행하였으며 식별된 문제점은 다음과 같다.

1) 배포를 위한 단계가 나눠지지 않아 배포를 위한 통제가 명시적으로 진행되지 않음
2) 개발자 테스트가 진행되지만 테스트 목적이 불명확함
3) 테스터가 진행하는 테스트의 목표가 불명확하여 Time Based Test로 진행됨
4) 고객에게 전달되는 산출물의 배포 기준이 모호함
5) 개발자가 테스터에게 개발을 요청할 때 비로소 테스트 범위와 대상을 알게 됨

4.3 소스코드 분석 및 요주의 파일대응 방안 도출

A 기업의 SVN에 접근하여 Repository가 생성된 이후로 2019.5월까지 전체 Commit log를 추출하였다. A기업은 버전관리 도구로 SVN을 사용하고 있어 Commit log 분석을 위해 StatSVN[18]을 활용하였다.

A기업의 경우, 하나의 앱을 생성하기 위해서는 5개의 Repository가 필요했다. 이에 [표2]와 같이 Repository별로 자주 변경되는 파일의 기준을 달리 정의하였다.

표 2 Repository 별 자주 변경 빈도 기준

Repo. 별칭	파일당 평균 변경 횟수	표준편차	분석 기준
AView	1.3회	3.5	5회 이상 변경된 파일
Base	1.0회	2.4	4회 이상 변경된 파일
API	2.3회	2.6	5회 이상 변경된 파일
DView	2.9회	7.7	10회 이상 변경된 파일
Engine	36.1회	45.4	20*회 이상 변경된 파일

* Engine 폴더에 빌드 및 Configuration 설정 파일이 있어 해당 이상치로 파악되는 3개의 파일을 제외한 평균으로 정의

1,093개 파일에서 1차 분석 범위로 465개 파일로 확정되었다. 465개의 파일의 Logical Coupled Group을 도출하기 위해 동일한 날짜에 같이 변경된 파일을 확인하기 위한 작업을 진행하였다. 이후 우리가 개발한 Logical Coupled Group을 분석하는 프로그램을 활용하여 같이 변경되는 파일 중 그 빈도가 높은 파일을 최종 Logical Coupled Group으로 선별하는 작업을 진행하였다. 분석 작업을 통해 최종적으로 도출된 Logical Coupled Group은 61개 Group으로 이와 관련된 상세 내용은 [표3]과 같다.

표 3 Logical Coupled Group 분석 결과

항목	현황	설명
그룹 개수	61개	- 465개 파일내 같이 변경되는 그룹이 61개 확인됨
그룹 당 평균 파일 수	3.1개	- 한 개의 파일 변경 시, 평균 2개의 파일이 같이 변경됨
3.1개 이상의 파일을 포함하는 그룹	12개	- 그룹당 5.8개의 파일 존재 - 12개 그룹은 한 파일 변경 시, 평균 5개 파일이 같이 변경됨
12개 그룹중 가장 많은 그룹에 속한 파일	DocumentFragment.java	- 6개 그룹에 속해 있어 해당 파일 변경 시, 변경의 영향이 넓음

마지막으로 Commit Log를 분석하여 결함에 의해 변경되는 파일을 도출하였다. Commit Log 분석 수행 시, ‘Crash, 오작동, 동작하지 않음, 오류, 결함’과 같이 변경이 결함으로 추정되는 파일로 식별하였으며 결함으로 변경된 파일 개수는 202개, 결함으로 인해 가장 많이 변경된 파일은 EvCompInterfaceMsg.java (19회)로 분석되었다

앞선 소스코드의 변경을 분석하여 최종적으로 자주 변경되면서, 결함을 포함한 파일 76개를 식별하였다. 76개 파일에 대해서는 A기업의 도메인과 비즈니스, 개발자의 개발 습관 고려하셔 중복코드, 함수복잡도, 상속관계의 적절성, 의존관계 분석을 분석했다.

또한, 분석의 결과로써 1,093개의 파일 중 변경 시 주의를 필요로 하는 파일 8개를 도출하였다. 식별된 요주의 파일은 변경의 횟수가 높을 뿐더러, 결함 관련성, 많은 Logical Coupled Group에 포함, 높은 함수복잡도, 중복코드에 관여하고 있어 코드 품질이 낮은 파일로 확인되었다.

이런 분석 결과를 기반으로 개발자 셀프 인스펙션 체크리스트를 도출하였는데 개발자들의 개발 습관은 물론 코드 분석 결과를 반영했다. 개발자 셀프 인스펙션 체크리스트 항목은 15개로 그 중 일부 항목은 [표4]와 같다.

표 4 개발자 셀프 인스펙션 체크리스트 항목

No	평가항목
1	AMainFragment.java 파일 내 중복코드를 확인하였는가?
2	AMainFragment.java와 FEditorFragment.java 두 파일 간 중복코드를 확인하였는가?
3	AMainFragment, BEditorFragment, CEditorFragment, DEditorFragment 4개의 형제 클래스 간 중복코드를 확인하였는가?

No	평가항목
4	함수의 복잡도가 40 이상인 경우, 40이하로 낮추었는가?
5	EFragment.java가 변경 시, Understand를 이용하여 하위 클래스를 파악하여 해당 하위 클래스들을 영향도 입장에서 리뷰했는가?
6	8번 Logical Coupled group에 변경이 이뤄진 경우, 같이 변경될 가능성이 높은 파일의 변경 여부를 확인하였는가?
7	양방향 의존 관계가 있는 EvBaseEditorFragment.java와 BEditorFragment.java 두 개의 파일에 대한 코드리뷰가 진행되었는가?
8	요주의 8개 파일에 대해 각 파일에 대한 코드리뷰가 진행되었는가? (단일 파일에 대한 코드품질)
9	요주의 8개 파일이 속한 Logical Coupled group에 대해 변경 가능성을 확인하였는가?
10	Commit 시 Jira ID와 변경이유가 작성되었는가?
11	SVN Tagging 정책에 맞게 코드가 Tagging 되었는가?
12	소스코드가 빌드가 되는 것을 확인하였는가?

4.4 배포프로세스 TO-BE 모델 수립

우리는 앞선 4.2에서 배포프로세스의 개선항목을 몇 가지 식별하였으며, 해당 개선항목에 대한 해결방안을 다음과 같이 도출하였다.

1) 배포를 위한 단계가 나눠지지 않아 배포를 위한 통제가 명시적으로 진행되지 않음
   : 배포프로세스의 단계를 정의하고, 단계 전환 시 인수되어야할 산출물과 인수담당자를 정의한다.
2) 개발자 테스트가 진행되지만 테스트 목적이 불명확함
   : 개발자에게 할당된 이슈에 대한 완료 기준을 작성하게 하고, 완료 기준을 테스트(기능테스트)한다.
3) 테스터가 진행하는 테스트의 목표가 불명확하여 Time Based Test로 진행됨
   : 테스트 전략을 수립하고, 이에 준한 인수테스트를 수행한다.
4) 고객에게 전달되는 산출물의 배포 기준이 모호함
   : 각 단계별 수행활동이 완료되었고, 테스트 결과 Major 결함이 1건 미만인 경우 배포한다.
5) 개발자가 테스터에게 개발을 요청할 때 비로소 테스트 범위와 대상을 알게 됨
   : 배포일정계획 수립 단계를 추가한다. 배포에 참여하는 모든 구성원이 일정에 협의하는 것은 물론, 배포 대상 이슈에 대해 설명하는 자리를 가져 테스터가 테스트 준비 시간을 늘리도록 한다.

또한, [표4] 개발자 셀프 인스펙션 체크리스트를 개발단계에서 테스트 단계로 전환하기 위한 주요 산출물로 정의하여 개발자들이 코드 품질을 확보할 수 있도록 했다.

4.5 파일럿 수행 및 배포프로세스 최종 정의

파일럿을 수행하기에 앞서 이런 접근 방법이 실제 비즈니스 수행에 어떤 유의미한 결과로 나타나는지 판단하기 위해 측정 지표를 도출하였다. 측정 지표로 도출한 항목과 배포프로세스 수립 전 수준 수치는 [표5]와 같다.

표 5 A기업의 배포프로세스 수립 전 수준

항목	설명	배포프로세스 수립 전
배포 후 결함 발견 수 (평균)	- 배포 후 일주일 내에 고객으로부터 보고된 Major 이상의 결함 건 수를 1 건으로 정의	5 건
SDK 배포 기간 (개월)	- 배포 대상으로 확정된 이슈가 QA팀에 검증이 완료되어 고객에게 SW로 배포되는 기간	2 개월

파일럿은 2회 걸쳐 진행되었다. 1회차의 경우 2개 기업, 2회차의 경우 4개 기업을 대상으로 진행되었으며 총 1.5 개월에 걸쳐 수행되었다. 파일럿 수행 과정 중 각 단계별로 배포프로세스 적합성을 판단하기 위해 각 단계별 산출물 검토와 담당자의 인터뷰를 진행했다. 배포프로세스의 각 task별로 양호, 보통, 미흡의 3단계로 분석하였고, 또한, 프로세스의 적합성은 전체 task 20개 중 80% 이상이 양호 또는 보통인 경우 프로세스가 적합하다 정의했다.

4.6 배포프로세스 적용 결과

수립된 배포프로세스를 적용한 결과 최종적으로 기존 대비 고객에게 인도된 다음 보고되는 결함 수는 0건, 배포에 걸리는 시간이 기존 2개월에서 0.7개월로 단축된 것을 확인하였다. 배포프로세스 적합성을 위한 지표 이외에도 기업내 배포프로세스 도입을 통한 변화를 확인하기 위한 지표를 도출하여 확인한 결과 [표6]과 같이 확인되었다.

[표6]의 결과에서 유추할 수 있듯 기존에는 고객으로부터 발견되어 보고되던 결함이 코드기반의 배포프로세스를 적용한 이후에는 배포하기 전에 확인이 되어 고객으로부터 결함이 발생되지 않았다. 또한, 기존 대비 테스터를 배포 초기에 투입시킴으로써 테스트베이시스 확보의 노력이 2.5MD가 증가하였다. 개발단계에서는 개발자 셀프 인스펙션 체크리스트의 적용, 개발 기준 작성과 같이 코드 품질 확보 활동이 증가되면서 이슈를 분석하는 시간이 증가되었지만

검증 단계에서 이슈가 Reopen되는 개수가 낮아져 Rework이 감소하는 효과를 갖게되었다. 본 연구의 목표였던 적절한 수준의 배포 기간 확보도 가능한 것으로 확인이 되었다.

표 6 A기업의 배포프로세스 수립 전과 후 수준 비교

항목	배포프로세스 수립 전	배포프로세스 적용 후	개선 현황
배포 후 결함 발견 수 (평균)	5 건	0 건	**5건 감소**
배포 전 결함 발견 수(평균)	3.5 건	8 건	**4.5건 증가**
테스트베이시스 확보 노력(평균)	1.5 MD	4 MD	**2.5MD 증가**
이슈 당 분석 시간 (평균)	10 분	30 분	**20분 증가**
검증단계에서의 이슈 Reopen률 (평균)	3.5 건	1 건	**2.5건 감소**
SDK 배포 기간(개월)	2 개월	0.7 개월	**1.3개월 단축**

테스트베이시스의 확보나 이슈 분석과 같이 기존 대비 리소스가 추가 투입되어야 하는 부분이 있지만 Rework의 감소, 고객 중심이 아닌 개발팀 중심의 배포 일정을 수립하는 과정에서 고품질의 빠른 배포가 가능할 수 있었다.

5. 결론 및 향후 과제

본 논문에서는 코드 분석 기반의 배포프로세스 수립에 대해 설명하고, 실제 기업의 실증 사례를 소개하였다.

고객을 만족 시키는 배포를 수행하기 위해서는 두 가지 목표를 달성해야 한다. 첫째는 배포의 속도이며 둘째는 배포 대상 산출물의 품질이다. 배포의 속도를 일정 수준 이하로 유지하기 위해서 구성원들의 재작업을 줄이고, 구성원 간의 합의로 배포 범위와 일정을 수립함으로써 배포의 속도를 조절할 수 있다. 배포 대상 산출물 품질의 경우, 배포 대상 산출물의 근원인 코드의 품질을 확보하는 것과 동작하는 SW를 전략적으로 테스트함으로써 산출물의 품질을 확보할 수 있다. 실제 기업의 적용사례를 통해 배포의 속도는 물론 배포 산출물의 결과가 확보되는 것을 확인하였다.

본 연구와 관련되어 향후 연구가 필요한 부분은 크게 세 가지로 정의될 수 있다. 첫번째는 Git 기반의 버전관리 도구를 활용하는 경우에 대한 연구이다. 본 논문은 버전관리 도구로 SVN에 한정하여 분석했다. 최근에는 SVN보다 Git 기반의 도구를 사용하는 경향이 높으므로, Git 기반의 버전관리 도구를 사용하는 경우에 대해 연구가 필요하다. 특히, Git 기반의 버전관리 도구는 GitHub, GitLab 등 다양한 솔루션이 존재하므로 솔루션에 상관없이 콘솔 명령어를 이용하여 로그를 수집하고 분석하는 연구가 필요하다.

두번째로 연구가 필요한 부분은 SVN과 Git 등 두 개 이상의 버전관리 도구를 활용하고 있는 경우에 대한 연구이다. 조직에 따라 하나의 버전관리 도구를 사용할 수도 있고 그렇지 않은 경우도 있다. 이런 경우에는 각기 다른 버전관리 도구에서 수집한 Raw data를 어떻게 가공하고 분석할 지에 대한 연구가 진행되어야 한다.

마지막으로는 버전관리 도구를 활용하지 않는 조직을 대상으로 한 연구이다. 이번 연구에서는 버전관리 도구를 사용하고 있음을 전제로 연구가 진행되었기 때문에 상대적으로 쉽게 자주 변경되는 파일과 Logical Coupled Group을 식별할 수 있었다. 하지만 버전관리 도구를 사용하지 않는 조직의 경우에는 본 논문 3.2절에서 진행된 프로세스를 적용하기 어렵다. 향후 연구는 개발자 인터뷰를 통해 자주 변경되는 파일을 확인할 수 있는 분석 방법과 분석 활동을 정의하는 것과, 자주 변경되는 파일과 Logical Coupled Group을 도출하기 위한 인터뷰 질의 문항을 정의하는 것에 대한 연구가 진행될 예정이다.

[참고문헌]

[1] Hyrum K. Wright, Dewayne E. Perry, Release Engineering Practices and Pitfalls, 2012 34th International Conference on Software Engineering, 2012

[2] Antti Lahtela and Marko Jäntti, Challenges and Problems in Release Management Process: A Case Study, 2011 IEEE 2nd International Conference on Software Engineering and Service Science, 2011

[3] Noureddine Kerzazi,Pierre N. Robillard, Kanbanize the Release Engineering Process, 2013 1st International Workshop on Release Engineering, 2013

[4] M. Saboe, The Use of Software Quality Metrics in the Materiel Release Process-Experience Report, Proceedings Second Asia-Pacific Conference on Quality Software, 2001

[5] 강홍렬, 권지인, 모바일 산업의 패러다임 변화와 향후 산업전략의 변화, 정보통신정책연구원, 2009

[6] R Hoda, N Salleh, J Grundy, The Rise and Evolution of Agile Software Development, IEEE Software, 2018

[7] Samer I. Mohamed, DevOps shifting software engineering strategy Value based perspective, IOSR Journal of Computer Engineering , 2015

[8] 이영로, STANDARDS PRISM (2) 표준시론(時論) - 클라우드 컴퓨팅 전망과 표준화 방향 -향후 10년 이상 IT서비스 흐름 주도할 새로운 패러다임, 한국정보화진흥원, 2011
[9] Henning Femmer, Daniel Méndez Fernández, Stefan Wagner, SebastianEder, Rapid quality assurance with Requirements Smells, Journal of Systems and Software Volume 123, Pages 190-213, January 2017
[10] Nanette Brown, Robert L. Nord, Ipek Ozkaya, Manuel Pais, Analysis and Management of Architectural Dependencies in Iterative Release Planning, 2011 Ninth Working IEEE/IFIP Conference on Software Architecture. June 2011
[11] Jarallah S. Alghamdi, Raimi A. Rufai and Sohel M. Khan , OOMeter: A Software Quality Assurance Tool, e Ninth European Conference on Software Maintenance and Reengineering, 2005
[12] Chin-Yu Huang, M.R. Lyu, Optimal release time for software systems considering cost, testing-effort, and test efficiency, IEEE Transactions on Reliability , Volume 54 , Issue 4 , Dec. 2005
[13] https://www.certguidance.com/release-deployment-management-itil-itsm/
[14] https://www.projectsmart.co.uk/how-to-do-raci-charting-and-analysis.php
[15] Marco D'Ambros, Michele Lanza, Reverse engineering with logical coupling, 2006 13th Working Conference on Reverse Engineering, Dec. 2006
[16] B. Henderson-Sellers, Object-Oriented Metrics. Measures of Complexity , Prentice Hall, 1996.
[17] Chidamber Shyam, Kemerer Chris, “A metrics suite for object oriented design”, IEEE Transactions on Software Engineering, June1994.
[18] https://statsvn.org/

SW 개발자 커뮤니티를 통한 개발 문화 혁신 사례

2019. 12. 24

김상기, 유광엽, 주석원

0

목차

1

1. 추진 배경

Open Source Software 개요

Open Source Software(이하, OSS)는 누구나 자유롭게 사용하고 수정, 배포할 수 있도록 한 SW로, 개발 비용/기간 단축 및 상용 SW Vendor 종속 탈피 가능한 장점 보유

【 OSS 개념 】

❑ 소스 코드가 공개되어 있고 수정, 배포 자유로움

구 분	상용 SW	OSS		
		Enterprise	Community	
			유지보수	자체 운용
소스코드 접근성	불가	가능		
개발자	단일 개발기업	다수 개발자		
SW간 연동성	나쁨	좋음		
예시	UNIX, Oracle DB	Redhat Ent, Mysql Ent. 등	CentOS, MariaDB, Library/Framework(Jquery등)	
도입 비용	소유권 유지보수비용	사용권 유지보수 비용	유지보수 비용	무료 (사내 조직/인력)
기술 지원	Vendor, 전용BP	BP 원인 분석, Bug Fix 지원	Community 사이트 Q&A Googling을 통한 이슈검색	

소유권 : 상용 License 소유 계약, 사용권 : 1년 단위 사용 갱신 계약

【 OSS 장점 】

❑ **개발 비용 절감 및 기간 단축**

※ OpenStack 기반으로 Cloud 관리 솔루션 개발, 개발비용 절감 6,000 +α MM → 240MM
OpenStack Queens Release Note 근거

※ 자사에 필요한 핵심 기능 개발에 역량 집중 (10%), 나머지 (90%)는 커뮤니티를 통해 개발되는 결과물을 활용하여 개발 기간 단축 및 비용 절감 (당사 TACO[SKT All Container OpenStack] 개발 사례)

❑ **OS, DBMS등 글로벌 상용 SW 종속 탈피**

※ MySQL, MariaDB 개선으로 Oracle DB 종속성 탈피 신규 ICT 서비스 개발 시 대부분 Open Source DB 사용

※ x86 서버 OS는 Open Source Linux가 오래전 부터 MS Windows의 대체제로 활용되고 있음

2

2

1. 추진 배경

OSS 사용 이슈

OSS 사용 사례 분석 시, OSS 활용을 위해 고려해야 할 7가지 Risk가 예상됨

【 OSS 사용에 따른 Risk 사례 】

- **개발자 특성에 따라 유사한 기능을 하는 다른 오픈소스를 사용, 개발하여 전체 종류의 증가**
- **웹서비스를 제공하는 App. OSS를 대상으로 원격코드 실행가능한 보안 취약점(위협등급 :상) 이 다수 발생하여 조치에 많은 기간 소요**
- **오픈소스 License 위반으로 소스코드 전체 공개 또는 배상 합의** '17년 12월 23억 (사례 : 한글과 컴퓨터 PDF 변환 오픈소스 부정 사용)
- **성능/안정성이 미흡한 서비스 도입에 따른 장애 발생**
- **OpenStack의 경우, 6개월 주기 버전 업데이트를 진행하며 2년 이전 버전은 Community 사이트 기술지원 중단**
- **오픈소스 유지보수 역량 미확보에 따른 긴급상황시 신속한 지원 불가**
- **업데이트 불확실 및 OSS 비용 유료화**

【 이 슈 】

① **유사 OSS 난립**

② **보안취약점 대응력 미흡** (패치개발 및 조치)

③ **License 위반**

④ **서비스 안정성 검증 필요**

⑤ **과거 버전에 대한 기술지원 중단**

⑥ **유지보수 방안 확보 미흡**

⑦ **지속사용 불가**(정책변경, 업데이트 중단 등)

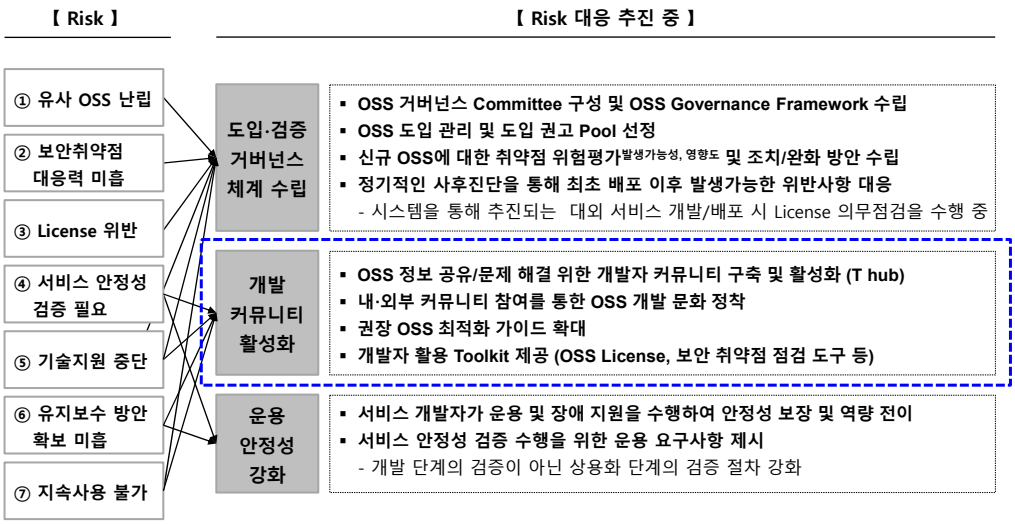
3

3

1. 추진 배경

OSS 대응 방안

OSS의 Risk 관리/해결을 위해 거버넌스 체계를 수립을 포함한 전사적 노력 추진 중



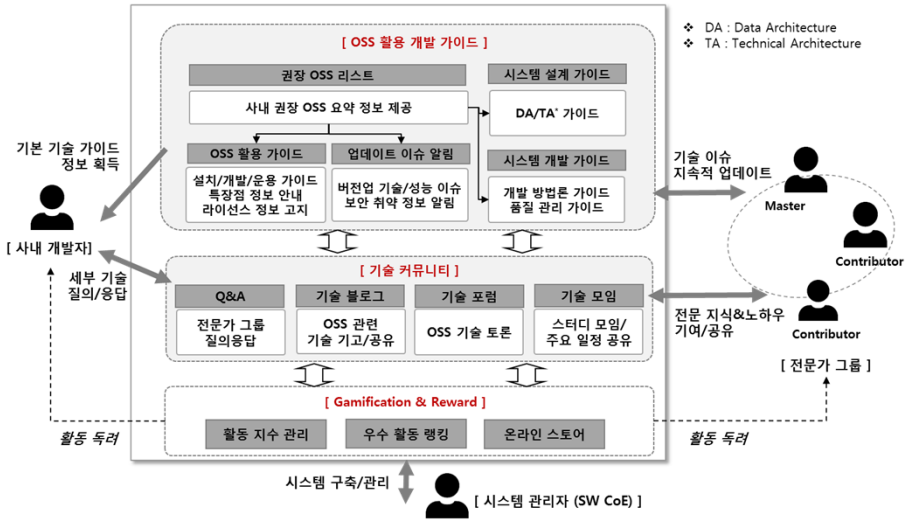
4

4

2. SW 개발자 커뮤니티 (T hub) 소개

Thub

OSS, SW 공학, Public Cloud 등 SK Telecom 개발자들의 SW 개발/운영 지식을 공유하고 협업을 통해 문제 해결하는 SK Telecom 개발자 Community(T hub) 구축 및 운영함

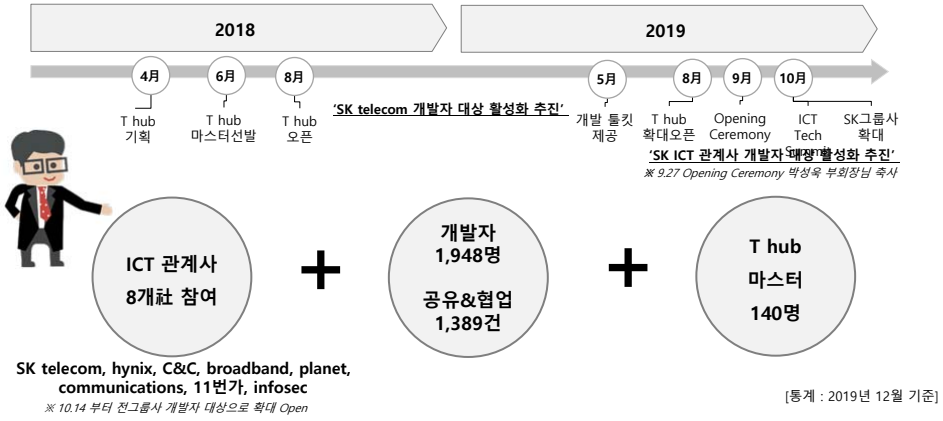


5

5

2. SW 개발자 커뮤니티 (T hub) 소개

Progress



['18년, '19년 T hub 오프닝 행사]

[마스터 Offline 모임]

[최우수 활동상 시상]

[기술 세미나 22회]

6

6

2. SW 개발자 커뮤니티 (T hub) 소개

커뮤니티 현황

AI, Big Data, 가상화 등 SKT에서 활발히 활용중인 46 종의 Open Source Software 와 Agile 방법론, MSA 구조, QA (Quality Assurance), UI/UX 등 21종의 Software 개발 기술 Sub Community로 구성되어 있으며, 분야별 사내 전문가인 T hub 마스터를 통해 활성화 추진 中

구분	권장 OSS (46종)
Application & Library (2)	Boost, DPDK
개발 Framework (11)	Angular.js, Echo, FFMpeg, Flutter, GStreamer, Hyperldeger fabric, jQuery, Node.js, React, Spring, Vue.js
AI Framework (7)	Caffe, Keras, MXNet, ONNX, PyTorch, Scikit-Learn, TensorFlow
WEB / WAS (2)	Apache WEB/WAS, Nginx
분산처리 / Big Data (6)	Druid, ElasticSearch, Hadoop, Kafka, Spark, Zookeeper
Cloud / 가상화 (8)	Ceph, Docker, GlusterFS, Kubernetes, KVM, Mesos, ONOS, OpenStack
DB (6)	CassandraDB, MariaDB, MongoDB, Mysql, PostgreSQL, Redis
HA (2)	HAProxy, Pacemaker
OS (2)	Android, Linux
구분	**SW 개발 기술 (21종)**
SW 공학 (12)	Agile, Data Architecture, Firmware, Git, License Compliance, MSA, Project Management, Programming Language, QA, SW Security, Tech Architecture, UI/UX
Public Cloud (4)	Amazon Web Service, Azure, Google Cloud Platform, Terraform
개발자 Toolkit(5)	Black Duck, mTworks, T hub 운영, T Performance, UXStudio

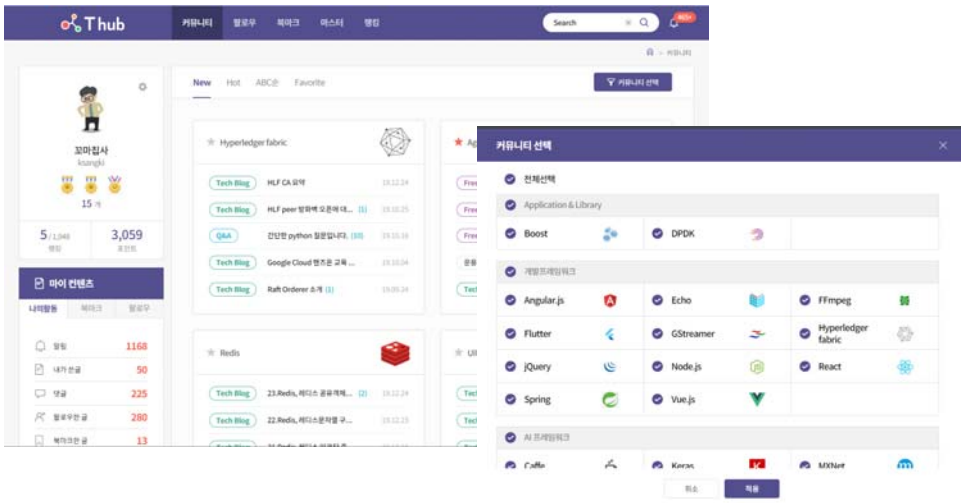
7

7

2. SW 개발자 커뮤니티 (T hub) 소개

데모

[커뮤니티 기능]



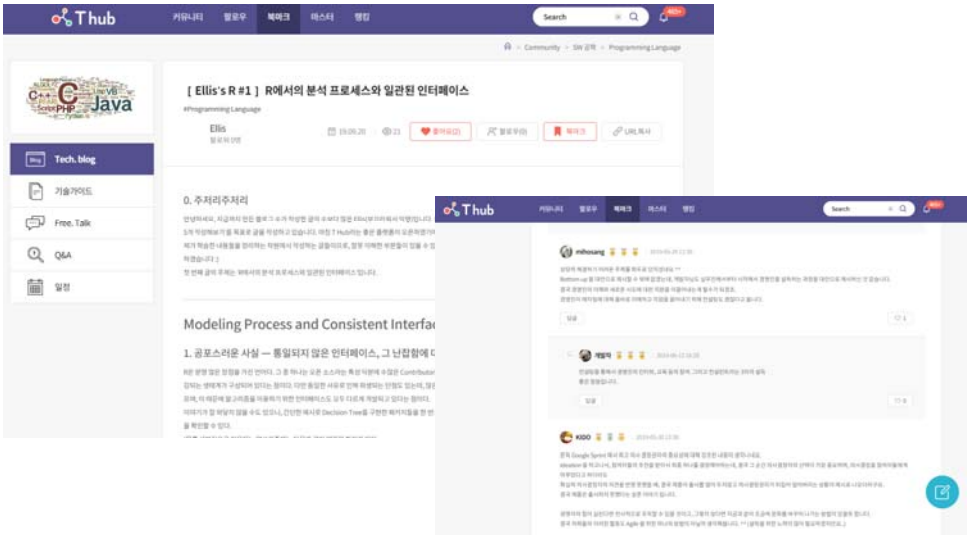
8

8

2. SW 개발자 커뮤니티 (T hub) 소개

데모

[Tech.Blog / Q&A]

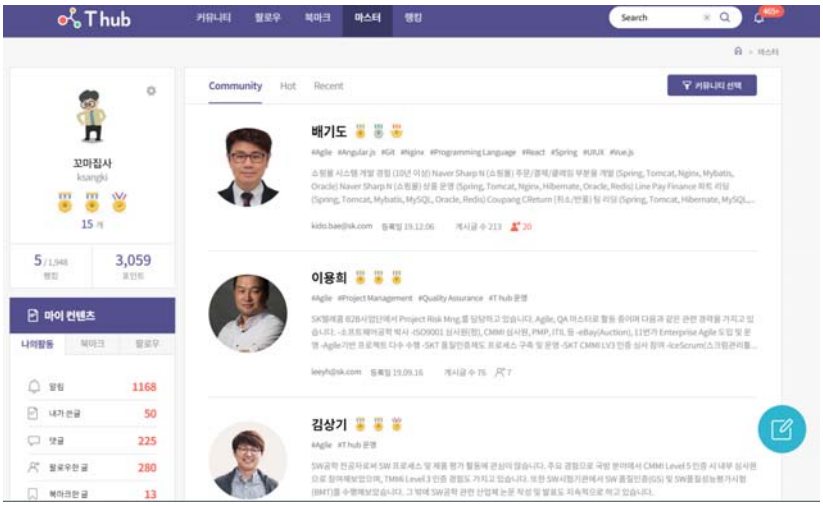


9

9

2. SW 개발자 커뮤니티 (T hub) 소개 데모

[마스터(전문가) 현황]

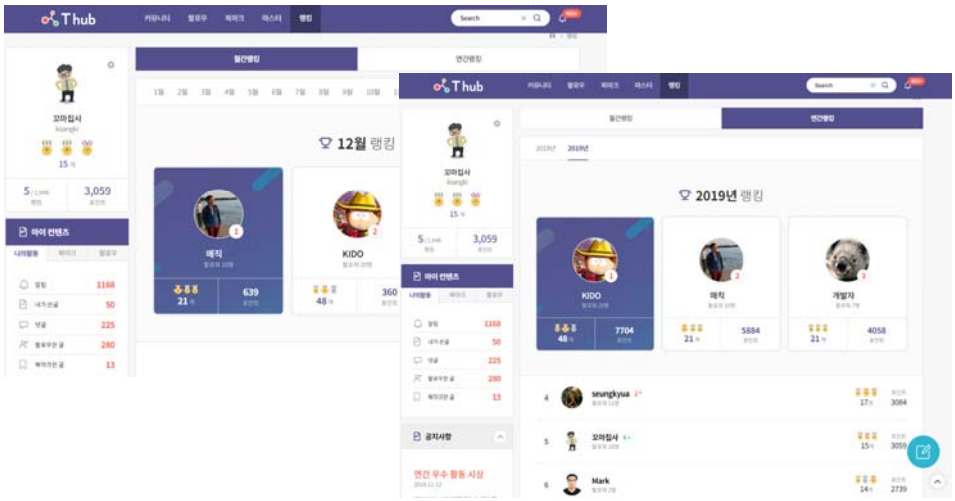


10

10

2. SW 개발자 커뮤니티 (T hub) 소개 데모

[활동 현황 – 랭킹]



11

11

3. 개발 문화 혁신 사례

커뮤니티 활성화 이벤트

[T hub Q&A 캠페인]

[개발자 Toolkit 제공]

[T hub Book 작가 모집]

12

12

3. 개발 문화 혁신 사례

Small Successes

다양한 개발자 Toolkit 제공 및 활용을 통해 개발 비용절감, 개발 기간 단축에 기여하고 있으며, 마스터 연재 Tech blog의 도서 출판, 창업 지원 센터에 도서 및 지식 기부 등 사회적 가치 창출 추진

개발 생산성 제고 (개발지원도구)

- **Cloud 形 성능 테스트 플랫폼 (w/ICT Infra 센터)**

: TiDC, AWS, Azure 에 구축한 시스템의 테스트 수행 환경 제공

T Map 경유지 최적화, T Map 대중교통, LBS iHPS, MBP API 등 전사 8개 과제 활용으로 테스트 비용 절감 및 개발 기간 단축

map API
경유지 최적화

- **Open Source Software 취약점 분석 시스템**

살아있는 동화, 자율주행, MEC 플랫폼, TACO 등 전사 7개 과제,

취약점 분석 및 개선 권고안 제공으로 OSS 사용 Risk 사전예방

Social Value 창출

- **T hub Book 도서 집필 및 출판**

: 6월에 선발된 T hub Book 작가들이 Blog 연재 및 출판 추진 중

작가	영역	주제/제목
OOO	프로그래밍	실전 프로젝트로 풀스택 엔지니어로 거듭나기
OOO	UI/UX	모두를 위한 UX
OOO	Cloud	Kubernetes Mesh Service 를 위한 Istio
OOO	AI	PyTorch 튜토리얼
OOO	Media Processing	Gstreamer User Book

※ 작가들은 OOO 출판사와 계약하여 정식 출판 진행

- **T hub Book 활용 지식 기부 (안)**

: 창업지원센터, 소프트웨어 특성화고 등 도서 기부 및 저자 강연

13

13

3. 개발 문화 혁신 사례

Ceremony

자발적인 활동과 협업에 대한 격려의 의미로 연간 우수 활동자들의 네트워킹 행사(마스터의 밤) 초대 및 조그마한 포상을 실시함

[전문가 네트워킹 행사]

[우수 Contributor Top 5 시상]

[Best Practice 상]

[협업상]

14

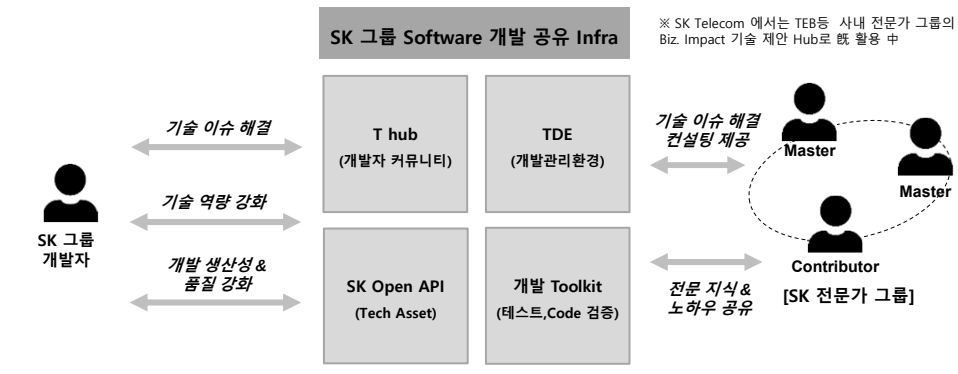
14

3. 개발 문화 혁신 사례

Next Step

"기술과 Practice를 공유하고 협업을 통해 문제 해결을 추구하는 개발 문화 혁신의 場"

Biz. Impact 기술 제안을 통해 상품 경쟁력에 기여하는 기술 기반 성장 촉진 Engine으로 진화



※ SK University의 교육/실습 및 협업 도구로 연계 활용 협의 中

15

15

3. 개발 문화 혁신 사례

Next Step

Enterprise Value

Social Value

2018
'SK telecom
Software Developer'

2019
'SK 그룹
Software Developer'

2020
'T hub for
Korea Software
Developer'

SK 그룹 Software 개발 공유 Infra

대한민국 Software 개발
공유 Infra

stackoverflow

of Korea Software Developer

Software Developer Summit

Software Developer Center
developer.sk.com

16

16

End of Document

17

17

국제표준 개정에 따른 국방 소프트웨어 개발절차 개선방향

박태현[1], 심승배[1*], 김의순[1], 조성림[1], 홍수민[1], 윤웅직[1]

[1]한국국방연구원 군사발전연구센터 전장정보화연구실

목 차

서론

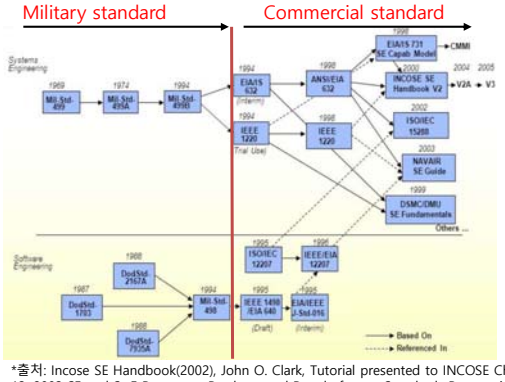
- 국방 소프트웨어 개발절차는 국제표준(ISO/IEC/IEEE 12207)과 미국 국방부의 관련 표준/절차를 참고하고 국방환경을 고려하여 개발되었음
  - 국방 표준 개발 프로세스(공정)
  - 국방 CBD 개발 프로세스(공정)

- 본 연구의 목적
  ①2017년 국제표준 개정에 따른 변화를 살펴보고
  ②국방 소프트웨어 개발절차에 대한 개선 요구사항을 식별하여
  ③개선방향을 제시

KIDA 3

소프트웨어 개발 분야 국제표준 동향 분석

- 소프트웨어 개발 관련 국제표준 현황
  - 미 국방 표준(1968 – 1994) → 산업 표준 (1994 ~)
    - 1969년 미 국방 표준(Mil-Std)이 시스템 공학으로 시작
    - 1995년 소프트웨어 개발 관련 국제표준의 기틀이 된 ISO/IEC 12207 제정

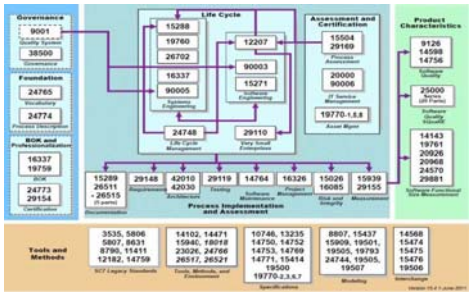


*출처: Incose SE Handbook(2002), John O. Clark, Tutorial presented to INCOSE Chapter Seminar, Nov. 13, 2003 SE and SwE Processes, Products and People from a Standards Perspective

KIDA 4

소프트웨어 개발 분야 국제표준 동향 분석

- 소프트웨어 개발 관련 국제표준 현황
  - **ISO/IEC JTC1의 소프트웨어 및 시스템공학 분과위원회(Software and Systems Engineering subcommittee, SC7)를 중심으로**
  - **ISO/IEC 15288 제정 후, 기존 ISO/IEC 12207 및 표준 및 보고서간 조정이슈 발생**
    - 시스템공학과 소프트웨어공학간 개념/용어/상세수준/프로세스 등 일관성 문제

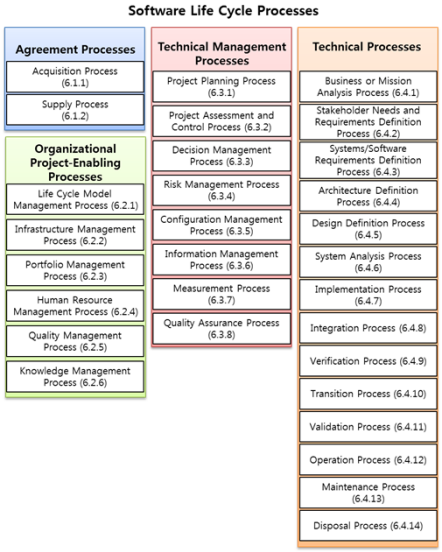


*출처 : ISO/IEC/JTC1/SC1 Webpage
(https://sites.google.com/site/isoiecjtc1sc7/)

KIDA 5

소프트웨어 개발 분야 국제표준 동향 분석

- ISO/IEC/IEEE 12207 표준 요약
  - 4개의 Process
    - 협약 프로세스
    - 조직의 프로젝트 기반 프로세스
    - 기술 관리 프로세스
    - 기술 프로세스
  - 30개의 Sub-process



KIDA 6

소프트웨어 개발 분야 국제표준 동향 분석

- **ISO/IEC/IEEE 12207 표준 요약**
  - **Agreement Processes**
    - 획득과 공급 두 조직간 협약을 수립하는데 필요한 활동 정의
    - 획득자는 제품 또는 서비스를 위해 공급자에게 업무를 부과할 수 있고, 협약을 통해 이뤄짐

<표 1> 협약 프로세스 그룹의 구성

프로세스	목적	핵심 산출물
획득 (Acquisition)	획득자 요구사항을 충족하는 제품 또는 서비스를 확보	공급요청서, 협약서
공급 (Supply)	합의한 요구사항을 충족하는 제품 또는 서비스를 획득자에게 공급	제품 또는 서비스 인도서

KIDA 7

소프트웨어 개발 분야 국제표준 동향 분석

- **ISO/IEC/IEEE 12207 표준 요약**
  - **Organizational Project-Enabling Processes**
    - 조직의 프로젝트 기반 프로세스는 프로젝트 착수, 지원 및 통제를 통해 제품 또는 서비스를 획득하고 공급하는 조직의 능력을 관리

<표 2> 조직의 프로젝트 기반 프로세스 그룹의 구성

프로세스	목적	핵심 산출물
생명주기 모델 관리 프로세스 (Life cycle model management process)	조직에서 사용할 정책, 생명주기 프로세스, 생명주기 모델, 절차를 정의, 유지관리 및 보장	생명주기 정책, 프로세스, 모델
기반구조 관리 프로세스 (Infrastructure management process)	수명주기 전반에 걸쳐 조직과 프로젝트의 목적을 지원하기 위해 프로젝트에 기반구조와 서비스 제공	기반구조 요구사항 정의
포트폴리오 관리 프로세스 (Portfolio Management Process)	조직의 전략적 목적을 충족시키기 위하여 프로젝트를 시작하고, 필요한, 충분한, 적합한 프로젝트를 유지	프로젝트 식별
인력자원 관리 프로세스 (Human Resource Management Process)	필요한 인력자원을 조직에 제공하고, 업무 요구사항과 일치하는 능력을 유지	프로젝트에서 요구되는 역량을 식별
품질경영 프로세스 (Quality Management Process)	조직과 프로젝트의 품질 목표, 고객 만족을 달성하기 위한 제품, 서비스, 품질관리 프로세스의 구현을 보증	품질 평가 기준과 방법
지식 관리 프로세스 (Knowledge Management Process)	조직이 기존 지식을 다시 적용할 기기를 활용할 수 있도록 능력과 자산을 생성	지식 자산 분류체계

KIDA 8

소프트웨어 개발 분야 국제표준 동향 분석

- **ISO/IEC/IEEE 12207 표준 요약**
  - **Technical Management Processes**
    - 프로젝트 계획수립, 평가 및 통제와 관련된 프로세스로 구성
    - 관리 측면 : 프로젝트 진도를 계획/실행/평가/통제에 필요한 프로세스
    - 지원 측면 : 특화된 관리 목적을 지원하기 위한 프로세스

<표 3> 기술 관리 프로세스 그룹의 구성

프로세스	목적	핵심 산출물
프로젝트 계획 (Project Planning)	효과적이고 실행 가능한 계획을 작성하고 이를 조직화	프로젝트 계획서
프로젝트 평가 및 통제 (Project Assessment and Control)	프로젝트가 적절한지 평가하고, 프로젝트의 상태와 기술 및 프로세스 성과를 결정하고, 기술 목표를 만족시키기 위하여 프로젝트 예산 내에서 계획과 일정에 따라서 수행되고 있다는 점을 보증	프로젝트 평가 결과
의사결정 관리 (Decision Management)	생명주기의 어떤 시점에서의 의사결정에 대한 대안을 식별하고 특징짓고 평가하기 위한 구조적이고 분석적인 프레임워크를 제공하고 가장 유익한 행동을 선택	의사결정 기준 및 결과
위험관리 (Risk Management)	지속적으로 위험을 식별, 분석, 처리하고 감시	위험관리 계획
형상관리 (Configuration Management)	생명주기에 대하여 시스템 구성요소와 형상을 관리하고 통제	형상관리 계획 (기준선 포함)
정보관리 (Information Management)	지정된 이해관계자들에게 정보를 제공하기 위하여 정보를 생성, 수집, 변환, 유지, 보관, 분배 및 폐기	정보관리 전략
측정 (Measurement)	제품, 서비스 및 프로세스의 품질에 대한 효과적인 관리를 지원하고 실증하기 위해 객관적인 데이터와 정보를 수집, 분석 및 보고	측정 전략 및 결과
품질보증 (Quality Assurance)	프로젝트에 대하여 조직의 품질 관리 프로세스의 효과적인 적용을 지원	품질보증 전략 및 결과

소프트웨어 개발 분야 국제표준 동향 분석

- **ISO/IEC/IEEE 12207 표준 요약**
  - **Technical Processes**
    - 조직 및 프로젝트 기능을 통한 이익을 최대화하고 기술적 의사결정 및 조치로부터 발생하는 위험을 최소화하기 위한 활동으로 구성
      - ➢ 소프트웨어 시스템을 위한 요구사항을 정의, 요구사항을 효과적인 제품으로 변환, 제품 및 서비스를 사용하고 제공하고 폐기하기까지의 활동을 정의

<표 4-1> 기술 프로세스 그룹의 구성

프로세스	목적	핵심 산출물
비즈니스, 임무분석 프로세스 (Business or Mission Analysis process)	비즈니스 또는 임무에 대한 정의 및 범위 설정, 문제점을 다룰 수 있는 방안 수립	비즈니스, 임무 관련 평가결과 및 대안
이해관계자의 니즈와 요구사항 정의 프로세스 (Stakeholder Needs and Requirements Definition process)	필요한 기능을 제공하는 시스템과 관련한 이해관계자의 니즈와 요구사항 정의	이해관계자의 요구사항 분석 결과
시스템/소프트웨어 요구사항 정의 프로세스 (System/Software requirements Definition process)	이해관계자의 요구사항을 기술적 관점으로 변환하기 위함	시스템/소프트웨어의 요구사항 분석 결과
아키텍처 정의 프로세스 (Architecture Definition process)	시스템 아키텍처 대안을 생성하고, 이해관계자의 관심사를 표현하고 시스템 요구사항을 충족하는 하나 이상의 대안을 선택하고, 이를 일련의 일관된 뷰로 표현	시스템 아키텍처 결과
설계 정의 프로세스 (Design Definition process)	시스템 아키텍처의 모델과 뷰에서 정의한 아키텍처적인 요소들이 구현과의 일관성 유지를 위해 상세한 데이터와 정보를 제공	설계 정의 결과

소프트웨어 개발 분야 국제표준 동향 분석

- **ISO/IEC/IEEE 12207 표준 요약**
  - **Technical Processes**
    - **조직 및 프로젝트 기능을 통한 이익을 최대화하고 기술적 의사결정 및 조치로부터 발생하는 위험을 최소화하기 위한 활동으로 구성**

<표 4-2> 기술 프로세스 그룹의 구성

프로세스	목적	핵심 산출물
시스템 분석 프로세스 (System Analysis process	전 수명 주기 동안 의사결정에 필요한 기술적 데이터와 정보의 기준을 제시	시스템 분석 결과
구현 프로세스 (Implementation process)	특정 시스템 요소를 구현	소프트웨어 구현 결과(소스 코드 등)
통합 프로세스 (Integration process)	일련의 시스템 요소를 시스템/소프트웨어 요구사항, 아키텍처 및 설계를 충족하도록 시스템(제품 또는 서비스)으로 통합	통합 전략 및 결과
검증 프로세스 (Verification process)	시스템 또는 시스템 요소가 지정된 요구 사항 및 특성을 충족한다는 객관적인 증거를 제공	검증 전략 및 결과
이전 프로세스 (Transition process)	시스템이 운영상황에서 이해관계자의 요구사항으로 명시된 서비스를 제공하는 능력을 수립	SW 이전 전략 및 결과
확인 프로세스 (Validation process)	시스템이 사용 중일 때 의도된 운영 환경에서 의도된 대로 사용되고 업무나 임무 목표와 이해관계자의 요구사항을 충족한다는 객관적인 증거를 제공	시스템 확인 전략 및 결과
운영 프로세스 (Operation process)	서비스를 제공하기 위해 시스템을 사용	운영 전략 및 결과
유지보수 프로세스 (Maintenance process)	시스템이 서비스를 제공할 수 있는 능력을 유지	유지보수 전략 및 결과
폐기 프로세스 (Disposal process)	명시된 의도대로 사용하기 위한 시스템 요소나 시스템의 존재를 끝내고, 대체되거나 멈춘 요소들을 적절하게 처리하고, 식별된 중요한 폐기 요구사항들을 적절히 다룸	폐기 전략 및 결과

KIDA 11

국방 소프트웨어 개발절차 분석 및 개선방향

- **국방 소프트웨어 개발절차 분석:** 개발준비, 분석, 설계

개발 단계		개발 작업
개발준비		소프트웨어 수명주기 모델 선택
		개발계획 작성
분석	시스템 요구사항 분석	시스템 요구사항 정의
		시스템 요구사항 검토 및 평가
	시스템 구조설계	시스템 구조설계 수행
		시스템 구조설계 검토 및 평가
	소프트웨어 요구사항 분석	소프트웨어 요구사항 정의
		소프트웨어 요구사항 검토 및 평가
설계	소프트웨어 구조설계	소프트웨어 구조 정의
		인터페이스 설계
		데이터베이스 설계
		사용자 문서 개발
		소프트웨어 통합시험 요구사항 정의 및 계획 작성
		소프트웨어 구조설계 검토 및 평가
	소프트웨어 상세설계	소프트웨어 구성요소 상세설계
		인터페이스 상세설계
		데이터베이스 상세설계
		사용자 문서 갱신
		소프트웨어 단위시험 요구사항 정의 및 계획 작성
		소프트웨어 통합시험 요구사항 정의 및 계획 갱신
		소프트웨어 상세설계 검토 및 평가

*출처: 국방부 (2019), 국방정보화업무훈령

KIDA 12

국방 소프트웨어 개발절차 분석 및 개선방향

- **국방 소프트웨어 개발절차 분석:** 구현 및 테스트, 인도

개발 단계		개발 작업
구현 및 테스트	소프트웨어 코딩 및 단위시험	단위 소프트웨어 코딩 및 데이터베이스 개발
		단위 소프트웨어 및 데이터베이스 시험절차/자료 준비
		사용자 문서 갱신
		소프트웨어 통합시험 요구사항 정의 및 계획 갱신
		소프트웨어 코드 및 단위시험 결과 검토
	소프트웨어 통합 및 자격시험	소프트웨어 통합계획 작성
		소프트웨어 통합 및 자격시험
		소프트웨어 통합결과 검토 및 평가
		사용자 문서 갱신
	시스템 통합 및 자격시험	시스템 통합계획 작성
		시스템 통합 및 자격시험
		시스템 통합결과 검토 및 평가
		사용자 문서 갱신
인도	시스템 설치	시스템 설치계획 작성
		시스템 설치 수행
	시스템 인수 지원	개발시험평가
		운영시험평가
		검수
		발주자 사후지원

*출처: 국방부 (2019), 국방정보화업무훈령

국방 소프트웨어 개발절차 분석 및 개선방향

- **국방 소프트웨어 개발절차 분석:** 무기체계 소프트웨어

*출처: 방위사업청(2018)

국방 소프트웨어 개발절차 분석 및 개선방향

- 국방 소프트웨어 개발절차 개선방향
  - **Agreement Processes**

<표 5> 협약 프로세스 그룹과 국방 소프트웨어 개발절차 비교 분석 결과

기술 관리 프로세스 그룹		국방 정보화업무 훈령(국방 소프트웨어 개발절차)	
프로세스	활동	관련 조항	개선 필요성
획득	· 획득 준비 · 획득 홍보와 공급자 선정 · 협약 수립과 유지 · 협약 감시 · 제품 또는 서비스 수락	제16조(정보화전략계획수립 소요) 제48조(제안요청서 작성) 제49조(입찰공고 및 제안설명) 제50조(제안요청서 변경) 제52조(계약 및 변경) 제56조(사업종결 보고) 제66조(인수 및 확인) 제67조(검수) 제68조(전력화) 제70조(정보화전략계획수립 대상 및 수행시기) 제155조(정보화사업 평가)	O
공급	· 공급 준비 · 제품 또는 서비스의 공급요청서에 대한 응답 · 협약 수립과 유지 · 협약 수행 · 제품 또는 서비스 인도와 지원	관련 조항 없음	X

국방 소프트웨어 개발절차 분석 및 개선방향

- 국방 소프트웨어 개발절차 개선방향
  - **Organizational Project-Enabling Processes**

<표 6> 조직의 프로젝트 기반 프로세스 그룹과 국방 소프트웨어 개발절차 비교 분석 결과

조직의 프로젝트 기반 프로세스 그룹		국방 정보화업무 훈령(국방 소프트웨어 개발절차)	
프로세스	활동	관련 조항	개선 필요성
생명주기 모델 관리	· 프로세스 수립 · 프로세스 평가 · 프로세스 개선	【별표 14】 국방 표준 소프트웨어 개발 공정별 산출물	X
기반구조 관리	· 기반구조 수립 · 기반구조 유지보수	관련 조항 없음	X
포트폴리오 관리	· 프로젝트 정의와 허가 · 프로젝트 포트폴리오 평가 · 프로젝트 종료	제2절 소요 기획 및 결정 제150조(정보화 평가의 구분) 제155조(정보화사업 평가) 제156조(정보화 평가 결과 보고 및 후속조치)	O
인력자원 관리	· 역량 파악 · 역량 개발 · 역량 획득 및 공급	제90조(정보화 교육)	X
품질경영	· 품질경영 계획 · 품질경영 평가 · 시정 및 예방 행동 수행	제131조(적용기간) 제133조(품질보증 활동) 제134조(품질보증 활동 기록)	X
지식 관리	· 지식관리 계획 · 조직차원의 지식과 기술 공유 · 조직차원의 지식자산 공유 · 지식, 기술, 지식자산 관리	관련 조항 없음	O

국방 소프트웨어 개발절차 분석 및 개선방향

- 국방 소프트웨어 개발절차 개선방향
  - **Technical Management Processes**

<표 7> 기술 관리 프로세스 그룹과 국방 소프트웨어 개발절차 비교 분석 결과

기술 관리 프로세스 그룹		국방 정보화업무 훈령(국방 소프트웨어 개발절차)	
프로세스	활동	관련 조항	개선 필요성
프로젝트 계획	· 프로젝트 정의 · 프로젝트 및 기술 관리에 대한 계획 · 프로젝트 활성화	제40조(사업 참여기관 지정) 제41조(사업계획서 작성 대상 및 시기) 제44조(사업계획서(발주) 제46조(사업계획서(발주) 승인)	X
프로젝트 평가 및 통제	· 프로젝트 평가 및 통제에 대한 계획 · 프로젝트 평가 · 프로젝트 통제	제54조(공급자 및 계약이행 관리) 제55조(사업추진 점검 및 조정) 제77조(개발 사업관리 이정표) 제78조(개발 사업관리 이정표별 합동검토)	O
의사결정 관리	· 의사결정 준비 · 의사결정 정보 분석 · 의사결정 및 의사결정 관리	관련 조항 없음	X
위험관리	· 위험관리 계획 · 위험 프로파일 관리 · 위험 분석 · 위험 조치 · 위험 감시	관련 조항 없음	O

기술 관리 프로세스 그룹		국방 정보화업무 훈령(국방 소프트웨어 개발절차)	
프로세스	활동	관련 조항	개선 필요성
형상관리	· 형상관리 계획 · 형상 식별 · 형상 변경 관리 · 배포 통제 · 형상 상태 보고 · 형상 평가	제135조(형상관리 목적 및 대상) 제138조(형상 식별) 제139조(변경 요구) 제140조(변경요구 검토 및 승인) 제137조(형상관리위원회 구성) 제141조(형상이력 관리)	O
정보관리	· 정보관리 준비 · 정보관리 수행	제124조(정보자원의 구분) 제126조(정보자원정보 등록·관리)	O
측정	· 측정 준비 · 측정 수행	관련 조항 없음	O
품질보증	· 품질보증 준비 · 제품 또는 서비스 평가 수행 · 프로세스 평가 수행 · 품질보증 기록 및 보고서 관리 · 인시던트(결함) 및 문제 처리	제130조(품질보증 목적) 제133조(품질보증 활동) 제134조(품질보증 활동 기록)	O

KIDA 17

국방 소프트웨어 개발절차 분석 및 개선방향

- 국방 소프트웨어 개발절차 개선방향
  - **Technical Processes**

<표 8-1> 기술 프로세스 그룹과 국방 소프트웨어 개발절차 비교 분석 결과

기술 프로세스 그룹		국방 정보화업무 훈령(국방 소프트웨어 개발절차)	
프로세스	활동	관련 조항	개선 필요성
비즈니스, 임무분석 프로세스	· 분석 준비 · 문제점 정의 · 대안범위 설정 · 대안 평가 · 비즈니스, 임무분석 관리	제6조(국방정보화기본계획서) 제13조(정보화 소요 대상) 제15조(소요원칙) 제16조(정보화전략계획수립 소요) 제23조(소요 제기) 제24조(소요검토) 제25조(소요 결정) 제42조(소프트웨어 개발 프로세스) 제55조(사업추진 점검 및 조정) 제69조(정보화전략계획수립 목적) 제70조(정보화전략계획수립 대상 및 수행시기) 제71조(정보화전략계획수립 원칙) 제72(정보화전략계획수립 결과 후속조치) 제74조(개발 방법론) 제138조(형상 식별)	O
니즈와 요구사항 정의 프로세스	· 니즈 정의를 위한 준비 · 니즈 정의 · 운영개념 및 그 외 수명주기 개념 설정 · 니즈를 요구사항으로 전환 · 요구사항 분석 · 요구사항 정의 관리	제11조(정보화사업 관련기관 임무) 제23조(소요제기) 제25조(소요결정) 제34조(중기계획 검토 및 반영) 제44조(사업계획서(발주) 작성) 제48조(제안요청서 작성) 제57조(시험평가 구분) 제60조(개발시험평가 실시) 제67조(검수) 제62조(운용시험평가 계획수립) 제69조(정보화전략계획수립 목적) 제74조(개발 방법론) 제75조(개발 산출물 작성 및 관리) 제135조(형상관리 목적 및 대상) 제138조(형상 식별)	O

KIDA 18

국방 소프트웨어 개발절차 분석 및 개선방향

- 국방 소프트웨어 개발절차 개선방향
  - **Technical Processes(Cont.)**

<표 8-2> 기술 프로세스 그룹과 국방 소프트웨어 개발절차 비교 분석 결과

기술 프로세스 그룹		국방 정보화업무 훈령 (국방 소프트웨어 개발절차)	
프로세스	활동	관련 조항	개선 필요성
시스템/소프트웨어 요구사항 정의 프로세스	· 시스템/소프트웨어 요구사항 정의 준비 · 시스템/소프트웨어 요구사항 정의 · 시스템/소프트웨어 요구사항 분석 · 시스템/소프트웨어 요구사항 관리	제16조(정보화전략계획수립 소요) 제42조(소프트웨어 개발 프로세스) 제44조(사업계획서(발주) 작성) 제57조(시험평가 구분) 제67조(검수) 제74조(개발 방법론) 제75조(개발 산출물 작성 및 관리) 제76조(소프트웨어 개발 공정) 제135조(형상관리 목적 및 대상) 제138조(형상 식별)	X
아키텍처 정의	· 아키텍처 정의 준비 · 아키텍처 뷰포인트 작성 · 후보 아키텍처의 모델 및 뷰 작성 · 아키텍처와 설계 연결 · 아키텍처 후보 평가 · 선택된 아키텍처 관리	제75조(개발 산출물 작성 및 관리) 제157조(국방아키텍처 분류 및 도입 대상) 제160조(국방아키텍처 도입 및 운영 기본원칙) 제135조(형상관리 목적 및 대상)	O
설계 정의	· 시스템 설계 정의 준비 · 각 소프트웨어 시스템 요소와 연관된 설계 설정 · 소프트웨어 시스템 요소 확보를 위한 대안 평가 · 설계 관리	관련 조항 없음	O

KIDA 19

국방 소프트웨어 개발절차 분석 및 개선방향

- 국방 소프트웨어 개발절차 개선방향
  - **Technical Processes(Cont.)**

<표 8-3> 기술 프로세스 그룹과 국방 소프트웨어 개발절차 비교 분석 결과

기술 프로세스 그룹		국방 정보화업무 훈령(국방 소프트웨어 개발절차)	
프로세스	활동	관련 조항	개선 필요성
시스템 분석	· 시스템 분석 전략 정의 및 분석 준비 · 시스템 분석 수행 · 시스템 분석 관리	관련 조항 없음	X
구현	· 구현 준비 · 구현 수행 · 구현 결과 관리	관련 조항 없음	X
통합	· 통합 준비 · 통합 수행 · 통합 결과 관리	관련 조항 없음	X
검증	· 검증 준비 · 검증 수행 · 검증 결과 관리	제57조(시험평가 구분) 제58조(시험평가 수행원칙) 제59조(개발시험평가 계획수립) 제60조(개발시험평가 실시) 제61조(개발시험평가 조치)	X
이전 프로세스	· 이전 준비 · 이전 수행 · 이전 결과 관리	제65조(정보시스템 설치) 제66조(인수 및 확인) 제67조(검수)	O

KIDA 20

국방 소프트웨어 개발절차 분석 및 개선방향

- 국방 소프트웨어 개발절차 개선방향
  - **Technical Processes(Cont.)**

<표 8-4> 기술 프로세스 그룹과 국방 소프트웨어 개발절차 비교 분석 결과

기술 프로세스 그룹		국방 정보화업무 훈령(국방 소프트웨어 개발절차)	
프로세스	활동	관련 조항	개선 필요성
확인 프로세스	· 확인 준비 · 확인 수행 · 확인 결과 관리	제67조(검수)	O
운영 프로세스	· 운영 준비 · 운영 수행 · 운영 결과 관리 · 고객 지원	제103조(운영 책임) 제105조(운영 절차)	O
유지보수 프로세스	· 유지보수 준비 · 유지보수 수행 · 물류 지원 수행 · 유지보수와 물류 결과 관리	제80조(유지보수 규모 산정 및 검토) 제81조(유지보수 형태) 제82조(유지보수 책임) 제83조(유지보수 절차) 제84조(유지보수 서비스수준 관리)	O
폐기 프로세스	· 폐기 준비 · 폐기 수행 · 폐기 마무리	제87조(폐기)	O

KIDA 21

결론

- 결론
  - **소프트웨어 개발 프로세스에 대한 국제표준 개정동향을 분석**
  - **현재의 국방 소프트웨어 개발절차를 분석**
  - **국방 소프트웨어 개발절차에 대한 개선 요구사항을 식별하여 개선방향을 제시**

- 향후 연구방향
  - **미 국방부의 소프트웨어 개발 관련 최신 동향 분석**
  - **애자일 방법론의 군 도입방향 모색**

KIDA 22

참고문헌

- 국방부. (2019).『국방 정보화업무 훈령』
- 방위사업청. (2018).『무기체계 소프트웨어 개발 및 관리 매뉴얼』
- 이중윤, 시스템공학 & 소프트웨어공학 국제표준 소개
- ISO/IEC/IEEE 12207 Systems and software engineering - Software life cycle processes (2017)
- ISO/IEC/IEEE 15288(2015) Systems and software engineering - System life cycle processes (2015)

OpenSource와 Web(IDE) Plug-In 중심으로 ALM 시스템 구축

전인복 ○ 백종문
ktds, 한국과학기술원
{jib120, jbaik} @kaist.ac.kr

Building ALM System Based on OpenSource and Web (IDE) Plug-In

Inbok Jeon ○ Jongmoon Baik
korea telecom data systems, Korea Advanced Institute of Science and Technology

요 약

소프트웨어 프로젝트의 성공적인 개발을 위해서는 개발 라이프사이클에서 각각 진행 단계에 따라 상황을 한눈에 파악되어야 하고 손쉽게 관리 되어야 한다. 이를 관리해주는 시스템이 ALM 이며 Application-LifeCycle-Management의 약자로 애플리케이션 개발에 사용되는 모든 생명주기들을 통합적으로 관리하는 시스템을 말한다. 본 연구에서는 이미 만들어진 다양한 Open Source Tool들을 통합하여 하나의 ALM Portal 시스템을 구축해보며, 더불어 Web IDE Plug-in개발과 함께 시스템을 손쉽고 편리하게 구축하는 방향을 제시한다.

1. 서 론

소프트웨어 개발이 점차 갈수록 활발해 지고 복잡해지고 있지만 개발 생명주기관리 과정별로 사용된 Tool의 형태는 대게 개별적이거나 독립적으로 사용하고 있다. 이러한 형태를 개선하기 위해 만들어진 시스템이 ALM (Application lifecycle management) 이다.

ALM 은 개발과정의 프로젝트계획, 요구사항분석 및 소프트웨어 개발 생명주기 (SLDC) 보다 더 광범위한 관점으로 애플리케이션을 개발하는 진행 과정의 모든 생명주기를 관리한다.

관리, 설계, 개발, 테스트, 이슈관리, 코드품질, 배포, 운영의 전반적인 진행과 상태들을 통합적으로 관리할 뿐만 아니라 다양한 정보들을 제공한다. 특히나 자사에서는 한해 SI프로젝트에 참여하고 있는 횟수도 증가하고 있기 때문에 이러한 ALM 시스템의 프로젝트 도입 사용도가 증가하고 있다.

프로젝트에 대한 전반적인 가이드 없이 개발을 시작한다면 고객, 개발자, PM/PL 및 기타 사용자들이 각각 개별적으로 환경에 수동적으로 관련 Tool에 접근해서 사용해야 한다. 프로젝트 이슈관리사이트, 개발환경, 저장소, 빌드, 테스트, 코드분석, 배포뿐만 아니라 MSA 시스템환경에서는 더욱 더 많은 관련 서비스 사이트들에 접속해 진행과정을 접속해서 확인해야 한다. 큰 규모의 프로젝트에서는 더욱더 불편할 수 있으며, 프로젝트를 관리하는 입장에서도 각각 유지보수를 하기가 쉽지 않다.

요구사항에 따른 개발 진행 과정도 별도 연결시켜 주는 절차 없이 사용한다면 이슈 추적하기에도 어려움이 있다. 요구사항에서 생성한 고유번호를 이용해 저장소이력과 연결해서 사용해야 하며 그에 따라 배포도 관리되어야 한다. 그리고 각 사용 Tool마다 접속하는 사용자를 동일하게 가입시켜야만 사용자별 구분이 가능하다. 그렇지 않으면 각각의 Tool들을 개별 로그인해서 사용해야 하는 불편함이 있다.

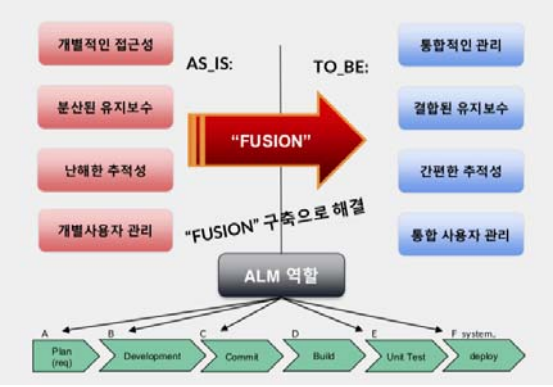


그림1. ALM의 도입효과

따라서 앞에서 언급한 그림 1과 같은 문제점들을 ALM “Fusion”을 도입해 해결하고자 한다. 물론 이러한 문제점을 해결하기 위해 상용화되어 출시되어 있는

ALM 시스템 제품군들이 존재한다. 대표적으로 Microsoft사에서 Azure Devops[1], 그리고 해외에서 유명한 Rommana ALM[2], 국내에서는 SilkRoad가 있으며 이중에 Azure Devops는 소규모 인원의 개발에서는 비용없이 개발이 가능하다. 하지만 일반 규모의 시스템에 도입하여 사용하기에는 제품 비용이 높은 편이며, 설치형이 아닌 클라우드 환경의 경우에는 프로젝트 종료 후에도 운영환경에서의 사용 유지보수 비용과 사용에 따른 과금이 추가적으로 발생 할 수 있다. 그뿐만 아니라 Tool에 대한 사용 적응 기간과 교육도 필요하다. 평소 사용하던 오픈소스 Tool이 아니라 각 솔루션에 맞는 개발 솔루션들을 사용함에 따른 Learning Curve가 높을 수 있으며, 단기 프로젝트에는 사용법 습득하기 위한 교육 및 학습시간이 소요되어 비교적 적합하지 않다.

따라서 최근 1년 동안 자사의 프로젝트에서 사용한 Tool들을 기반으로 ALM 시스템 환경을 구축하고자 하였다.

2. 개선방향

첫번째로 사이트별 개별적으로 접근하여 사용하던 시스템 Tool들을 통합적으로 하나로 연결하기 위해 그림 2의 형태로 ALM Portal 메인 사이트를 구축하였다. 여기서는 모든 사이트의 정보들을 중앙 통제하는 역할을 수행하게 되며 사용자별로 필요한 정보들을 관리 및 저장시키고, 사이트에서 필요한 주요 정보들을 요약하여 메인 대시보드 화면에 출력하거나 사이트들을 연결 시켜 준다.

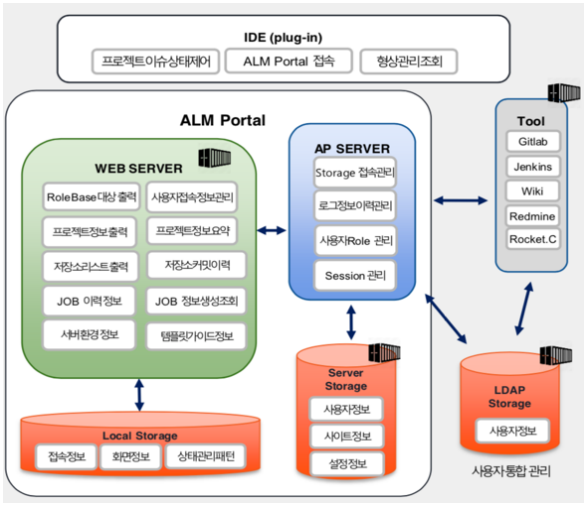


그림 2. ALM Portal 환경구성도

두번째로 모든 시스템들의 사용과 유지보수를 간소화 시키기 위해서 컨테이너로 각각의 사이트들을 모두 이미지화 시켜 관리하도록 생성 하였다. 컨테이너 기반으로 Volume 마운트를 외부로 꺼내어 모두 데이터와 함께 저장되기 때문에 이미지를 이용해 OS나 시스템환경에 상관없이 손쉽게 Docker환경에서 실행 시킬 수 있도록 생성하였다.

그리고 플러그인을 별도 개발하여 추적성의 문제를 해결하고자 하였다. 개발자들이 가장 많이 사용하고 있는 IDE로 Vscode에 플러그인을 개발하였으며 여기에서는 나에게 할당된 일감과 전체일감 그리고 우선순위, 진행상황을 리스트형식으로 한눈에 파악하여 볼 수 있으며 동시에 수정도 가능하도록 구성되어 있다.

그리고 여러 오픈소스 솔루션들을 하나로 통합시키고자 LDAP을 이용해 사용자관리를 통제했으며 솔루션별 로그인의 추가적인 인증절차를 없애기 위해 SSO를 도입해서 해결하였다.

3. 구축내용

ALM Portal 사이트의 구축은 기본 Web 환경으로 구성되어 있으며 그림 3과 같은 구성으로 오픈소스가 사용되었다.

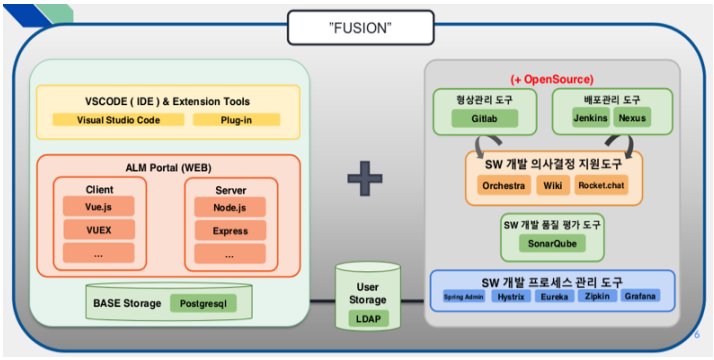


그림3. Fusion 오픈소스구성도

Client에서는 Vue.js Web Framework기반으로 Vuex, element-UI 및 기타 npm module들로 생성하였고, Server 는 Node.js 기반으로 Express로 서버 및 Axios 통신 등으로 구성하였다. Portal내에서의 통신과 사이트 간의 통신은 RESTful 로 API로 데이터를 주고 받고 있으며, 기본 Storage로 Postgresql을 사용하였다.

다음은 Open-Source Tool에 대한 소개로 의사결정 지원도구 Redmine이 프로젝트에 대한 요구사항 이슈관리로 사용되며, 산출물에 사용되는 다이어그램 및 도표는 Xwiki[6] 및 Diagram Extension을 사용한다. 사용자간에 지원도구로는 Rocket.chat Community 버전을 설치했고, 배포관리도구로는 Jenkins를 사용하며 라이브러리 Repository 관리를 위해 Nexus도

구성하였다.

프로젝트 개발에 사용시 제공될 템플릿이 별도 존재하는데 Spring Boot 기반에 MSA Template으로 구성하였다. 서비스별로 모니터링할 수 있도록 개발 프로세스 관리도구로 Spring Admin, Eureka, Hystrix, Zipkin, Grafana 사이트들의 정보를 저장한다.

사용자 권한이 필요한 모든 Tool들의 계정 관리는 OpenLdap을 설치해 LDAP환경으로 구성하였다.

이러한 사용자 관리는 메인 Portal 사이트에서 가입, 통제시키도록 개발하였고 통합된 계정을 이용해 SSO를 위한 Key-Cloak도 함께 구성 하였다. 따라서 Portal 메인화면에서 전체를 통제시키고 통일된 환경구성 역할을 한다. 접속은 일반 웹브라우저를 사용해도 되며

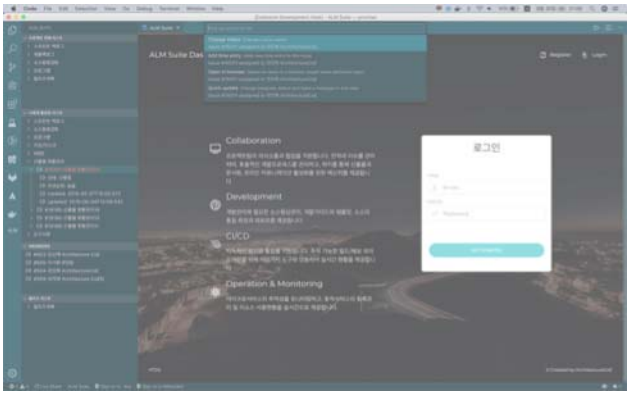


그림4. Fusion 접속화면

그림 4와 같이 Vscode[4] IDE에서도 Portal에 접속하여 사용할 수 있다. IDE에는 다음과 같이 Portal 사이트를 이용하며, 프로젝트 관리 Tool인 Redmine[5] 에 API를 호출해 진행중인 프로젝트에 대한 전체 일감과 본인의 일감 진행상태를 확인하고 수정할 수 있다.

Vscode에서 지원해주는 API를 활용해 IDE에서 Portal 사이트를 접속해서 활용한다.

4 평가 및 결론

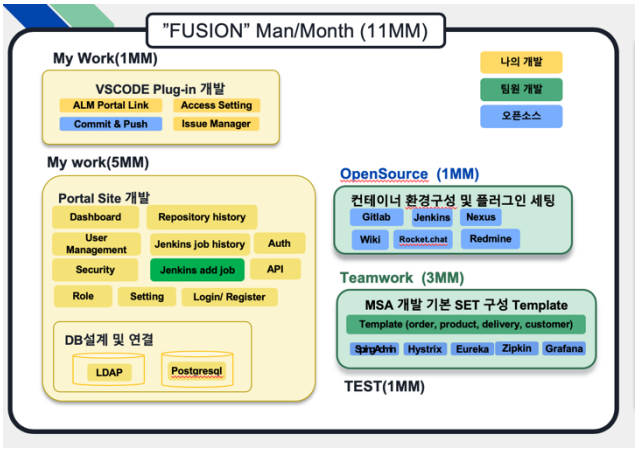


그림 5. Fusion 개발 공수

OpenSource Tool들을 사용하지 않고 ALM의 모든 환경 SET을 만들기에는 많은 공수가 필요 하지만 Opensource를 사용하여 각 사이트의 환경을 잡고 통합시켜 만들면 많은 인력이 투입되지 않아도 프로젝트관리에 효율적인 ALM 환경을 구성해 볼 수 있다.

그림 5와 같이 Fusion의 개발에 투입된 개발 공수는 약 11MM 이며 개발 기간은 약 6개월로 저자를 포함한 총 투입 인원은 3명이다.

최대한 오픈소스 Tool과 NPM Module들을 사용하여 개발하였으며, Portal 사이트에 대한 개발이 5MM으로 가장 많은 부분을 차지하였고 기타 컨테이너 환경구성과 Plug-in 개발이 1MM 가 투입되었다. 테스트는 시나리오별로 요구사항 등록부터 이슈 할당, 개발, 저장소 등록, 이력확인, 자동배포, 코드릴리즈, 테스트 등 다양하게 진행하였으며, 시스템의 모든 구성은 도커로 컨테이너화 관리되도록 개발하였다.

MM당 비용으로는 약 450만원으로 측정하였으며 총 인건비는 약 5000만원이다. 커머셜 ALM제품군을 도입하여 사용했을때와 비교하면 프로젝트당 도입비용은 비슷하지만, 프로젝트에 투입되어 사용되는 입장에서 보면 누적 비용은 꽤 큰 차이를 보인다.

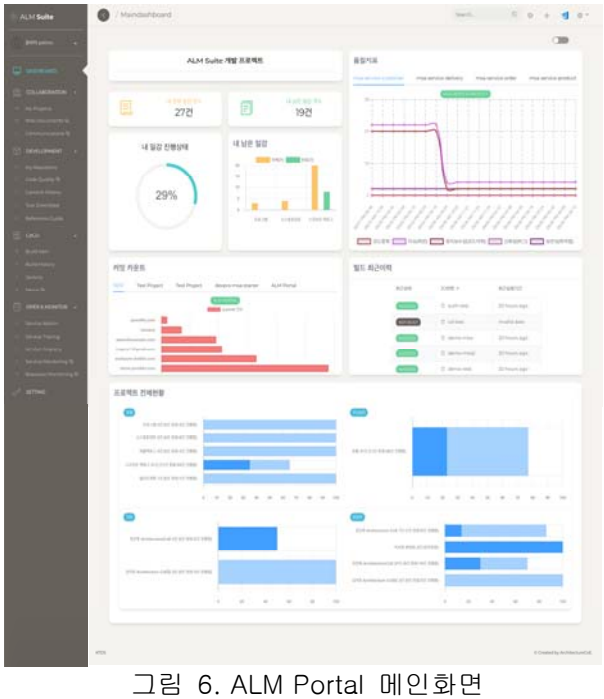


그림 6. ALM Portal 메인화면

5. 추후 과제

ALM 시스템을 수월하게 구축하기 위해서는 프로젝트 개발 프로세스의 전반적인 흐름을 이해하고 환경을 구성해야 한다. 각각의 Tool들이 제공해주는 많은

기능을 이해하고 잘 활용할 수 있어야만 원활한 Packaging 환경을 구성을 할 수 있으며, 그래야만 투입된 프로젝트마다 변경된 요구사항들을 손쉽게 반영하고 솔루션간의 이질감 없이 자연스러운 구성이 가능하다.
이번에 사용한 Tool외에 다른 방법의 솔루션 도입으로 가능한 방법이 있는지 좀 더 알아보고 다른 패키징 구성도 구축해볼 필요가 있다.

참고문헌

[1]azure devpos: https://azure.microsoft.com/ko-kr/services/devops/
[2]rommana : https://rommanasoftware.com/
[3]gitlab:https://docs.gitlab.com/
[4]vscode:https://code.visualstudio.com/api/references/vscode-api
[5]redmine:https://www.redmine.org/projects/redmine/wiki/Rest_api
[6]xwiki:https://www.xwiki.org/xwiki/bin/view/Documentation/

데이터베이스 서버 그룹 관리를 위한 EMS

김효일 ○ 백종문
KT, 한국과학기술원
{jayden, jbaik} @kaist.ac.kr

Enterprise Management System for Database server groups

Hyo-Il Kim ○ Jong-Moon Baik
Korea Telecom, Korea Advanced Institute of Science and Technology

요 약

데이터는 컴퓨터 산업의 발전에서 가장 중요한 자원 중 하나이다. 이러한 데이터는 일반적으로 DBMS (Database Management System)와 같은 관리 시스템을 이용하여 관리하는데 각종 민감한 정보들이 포함되어 있고 관리의 용이를 위하여 온프레미스(On-premise) 환경하에서 관리되고 사용되었다. 그러나 기술의 발전과 더불어 이러한 데이터에 대한 보안 요구사항은 더 높아지면서 운영 및 유지보수 비용을 낮추는 전략을 Cloud에서 찾기 시작했다. 본 연구는 Cloud환경에서 DBMS를 운영할 때 제한된 자원에서 운영되는 EMS 구축 방안을 소개한다. 기존 온프레미스(On-premise) 환경보다 더욱 제한된 환경과 정보만으로 다양하게 사용중인 불특정 다수의 고객 DBMS의 고가용성(HA)을 만족시키며 DBMS운영에 필요한 부가기능도 지원하는 EMS 구축 방안을 제안한다.

1. 서 론

1.1 연구 배경 및 목적

2019년 Gartner의 Future of the Database Market Is the Cloud에 따르면 Cloud 서비스 매출은 2022년까지 총 3,320억달러로 2019년에 비해 총 1.5배 성장한다고 발표했다[1]. 그 이유는 지속적인 IT 산업의 발전으로 사용자들은 인터넷을 통해 Cloud 컴퓨팅에 대한 전문적인 지식과 기술 없이도 애플리케이션, 스토리지, OS 보안 등 자신이 필요한 IT 자원을 원하는 시점에 필요로 하는 만큼만 대가를 지불하고, 서비스를 제공받을 수 있게 되었으며[2,3], 기업은 IT 자원을 보유하고 관리하는 대신, 전기회사에서 공급하는 전기를 이용하 듯 간단하고 저렴한 비용으로 IT 자원을 빌려서 사용함으로써 쉽고 저렴한 서비스를 이용할 수 있게 된 것이다[4,5]. 이처럼 IT 산업의 사용 환경이 기존 온프레미스(On-premise)에서 Cloud로 변화되고 있는데 Gartner는 Cloud 상품들 중 PaaS(Platform as a Service) 상품이 2019년 대비 2022년까지 117%의 상승을 보일 것으로 예측했다. 그 중 DBPaaS (Database Platform as a Service)의 상승이 가장 높았는데 이는 기업 보안의 이슈로 데이터를 외부 기관에 보관하기를 꺼렸던 기업들 조차 비용과 운영 효율화 이슈로 Cloud환경에서의 데이터베이스 사용을 시작했다는 것을 의미한다[1].

그러나 이러한 Cloud DB 서비스가 폭발적으로 증가함에도 운영을 위한 시스템은 기존 온프레미스(On-Premise)환경에 맞추어져 있다. 본 연구는 기존의 운영 환경이 Cloud환경에 적합하지 않는 이유와 여러 제약 사항을 소개하고 Cloud환경에 맞는 EMS 구축 방안을 제안한다.

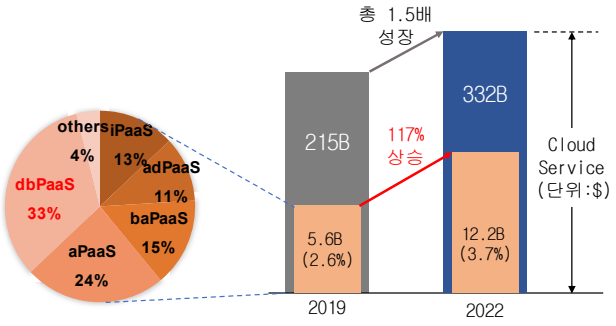


그림 1. Cloud PaaS 매출 예측

1.2 Cloud Gen1과 Cloud Gen2의 비교

Oracle OpenWorld 2018에서 기존 Gen1 Clouds가 자원을 공유 과정에서 User Code와 Cloud Control Code가 컴퓨팅 환경을 공유하도록 설계되어 CSP (Cloud Service Provider)가 고객의 데이터를 보거나 사용자 코드에 접근할 수 있는 취약점을 가졌다고 소개하였다. 이는 기업이 Cloud 환경에서 DBMS 사용을 신뢰하지 못하는 가장의 큰 원인이었다. 그래서 Gen2 Cloud Architecture에서는 CSP들이 고객의 데이터를 보거나 User Code에 절대 접근할 수 없도록 설계했다고 밝혔다[6]. 즉, Cloud 사용자 입장에서는 온프레미스(On-Premise)환경과 동일한 데이터 보호 환경을 제공할 수 있게 되었으나, 운영자들 입장에서는 서비스 장애를 비롯한 특수한 상황을 제외하고는 고객의 장비에 접근하거나 어떠한 행위도 할 수 없게 되었음을 의미한다.

1.3 Cloud환경의 운영 제약 및 요구 사항

Cloud 환경은 기존 온프레미스(On-premise)환경과는 다른 운영 제약사항이 있다.

첫째, 중단 없는 서비스이다. 이는 온프레미스(On-premise)환경과 동일하지만 선택사항이 아닌 필수사항이다. 이에 대한 대비책을 항상 강구해야 두어야 하며 운영자 개입 없이 시스템 컨트롤이 가능해야 한다.

둘째, 접근제한이다. Cloud는 불특정다수의 사용자가 존재한다. 인프라 서비스는 CSP가 제공하지만 사용되는 데이터는 사용자의 소유이기 때문에 CSP들은 운영에 대한 자동화가 필수이다. 또한, 장애와 같은 특수한 상황을 제외하고는 사용자의 자원에 접근할 수 없으며 관련 정보도 알 수 없다. 그리고 보안의 이슈로 특정 IP를 제외하고는 CSP들은 Cloud 자원에 접근할 수 없다.

셋째, Scale-out이다. 자원을 효율적으로 저장하고 사용하는 방법이 필요하다. 이는 Cloud서비스의 기조와 같은데 Cloud서비스가 무한한 자원을 이용할 수 있지만 내부적으로는 물리적인 자원들의 모음이다. 따라서 CSP들은 자원을 효율적으로 관리함으로써 Cloud 서비스의 사용 효율을 높이고 장애 대응을 해야 한다.

2. 관련연구

2.1 Linux-HA

Linux-HA 프로젝트는 신뢰성(Reliability), 가용성(Availability), 내구성(Serviceability)을 위한 고가용성 솔루션을 제공한다[9]. Linux-HA 프로젝트의 핵심 소프트웨어인 Heartbeat[10], DRBD[11]등을 이용하여 시스템에 고가용성을 보장하기 위한 방안을 제시하고 있으며, 이를 이용하여 데이터베이스, 웹, LVS, 메일, DNS, DHCP, Proxy Caching 서버를 구성할 수 있다[8][10].

2.2 Heartbeat

Heartbeat은 Linux-HA 프로젝트의 핵심 프로그램으로 시리얼(Serial line)이나 UDP 통신을 이용하여 고가용성(HA) 시스템에서 특정 노드의 결함을 지속적으로 감시하는 기능을 한다. 고가용성(HA) 시스템 노드들의 전체적인 구성과 운영방법에 따라 Active/Active방식과 Active/Standby방식으로 구분할 수 있다[8][12][13][15].

Active/Active방식은 각각의 노드가 각각 자신의 서비스를 제공하다가 하나의 노드에 장애가 발생하면, 장애가 발생하지 않은 노드가 장애가 발생한 노드의 서비스를 모두 담당한다. 이는 장애발생시에도 중단 없는 서비스를 제공할 수 있는 장점이 있지만 장애가 발생하지 않은 노드가 장애가 발생한 노드의 작업을 모두 수행해야 하므로 오버헤드로 인한 2차적인 장애를 유발할 수 있는 단점이 있다.

Active/Standby 방식은 각 노드가 primary/backup 방식으로 동작하며, primary 노드에 장애가 발생할 경우 대기하고 있던 backup 노드가 모든 서비스를 이전받아 나머지 작업을 수행한다. 이 방식은 구현이 쉽다는 장점이 있지만 primary 노드에 장애가 발생하기 전에 backup 노드는 아무런 처리를 하지 않고 대기해야 하므로 부가적인 노드 운영에 대한 비용 이슈가 존재한다[14].

2.3 DRBD(Distributed Replicated Block Device)

고가용성 시스템에서는 각 노드에서 처리하는 데이터가 항상 일관성과 무결성을 유지하여야 한다. 이를 위해 시스템 설계에 따라 공유디스크를 사용하여 동기화 하는 방법과 로컬 디스크를 사용하여 동기화 하는 방법으로 나눌 수 있다. DRBD는 디스크를 이중화하는 방법으로 각 노드의 로컬디스크를 동기화하는 기법으로 자신의 로컬디스크에 DRBD를 위한 파티션을 생성하여, 처리되는 데이터를 저장하고 실시간으로 다른 노드에 동기화하는 역할을 한다. DRBD의 동작 방법은 TCP/IP네트워크를 통해서, 두 서버 간의 데이터를 실시간으로 동기화 한다[11].

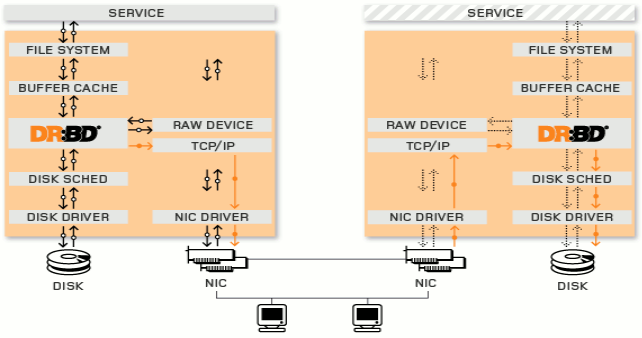


그림 2. DRBD Architecture

2.4 시스템 모니터링

시스템 모니터링은 수집기가 모니터링 대상 시스템의 정보를 가져옴을 의미하는데 데이터 수집 방식에 따라 크게 두가지로 나뉜다.

첫번째로 Agentless방식이다. 이는 Agent설치 없이 모니터링 대상 시스템에 접근해서 모니터링 결과를 가져오는 방식(Pop)으로 Agent설치 없이 사용할 수 있다는 것이 큰 장점이다.

두번째로 Agent 방식이다. 이는 모니터링 대상 시스템에 Agent를 설치하여 Agent를 통해 수집된 결과를 서버로 전송하는 방식(Push)이다. 관리가 용이한 반면 반드시 프로그램을 설치해야하는 단점이 존재한다[7].

3. 데이터베이스 서버 그룹 관리를 위한 EMS

본 EMS는 Cloud환경에서 DBMS를 운영하는 운영자들을 위한 시스템이다. 기존 온프레미스(on-

premise)용 DBMS 운영시스템이 Cloud환경에서 운영되기 어려운 이유는 1.3 Cloud환경의 운영 제약 및 요구 사항에서 확인할 수 있는데 이러한 이유로 본 EMS를 기획하였다. 특히, DBMS 관리만을 위한 간단하고 가벼운 구조를 가져가기 위해 기존 DBMS 관리 시스템에서는 제공하나 Cloud환경에서는 적용되기 힘든 기능들은 제거하고 고가용성(HA) 기능에 중점을 두고 기획하였다. 그리고 DBMS운영에 필요한 부가 기능을 반영할 수 있게 모듈식 방법으로 설계하였다. 본 EMS는 아래 환경으로 구현하여 CloudDB 시스템에 적용하여 현재 4개월째 이상없이 운영중이다.

Environment	Version
Mysql, Mysql utility	5.6
python	2.6
Mysql Connector/python	1.2
GTID	

3.1 고가용성(High availability) Group 관리

고가용성은 사전적의미로 장애가 없는 성질을 의미하는데 본 연구에서는 사용자가 설정한 데이터베이스 그룹을 HA Group이라 칭한다. 사용자는 HA Group을 생성하여 데이터베이스들간의 HA를 구성하여 관리한다.

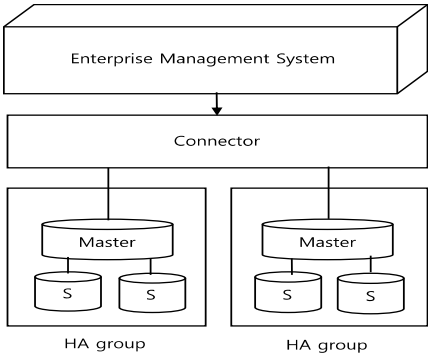


그림 3. System Diagram

3.2 System Working Status

본 EMS에서는 관리하고 있는 데이터베이스의 상태를 아래 4가지 단계로 표현하고 관리한다.

① Primary: 정상동작중인 상태로 MasterDB를 말한다. MasterDB는 Read/Write가 가능하다.
② Secondary: 정상동작중인 상태로 SlaveDB를 말한다. SlaveDB는 Read만 가능하며 장애 시 SlaveDB가 MasterDB가 된다.
③ Spare: 정상동작중이지만 운영에서 제외된 DB를 말한다.
④ Faulty: 장애상태이다.

3.3 System Environment

본 시스템(EMS)은 CLI(Command-Line-Interface)를 기반으로 구현하였다. EMS와 터미널간에 XML 기반의 분산 시스템 통신방법으로 HTTP를 통해서 간단하고 이식성이 높은 원격 프로시저 호출(RPC)하는 XML-RPC를 이용하여 통신한다. 그리고 EMS는 각 데이터베이스 서버 상태를 확인(Heartbeat)하여 어떤 데이터베이스가 사용해야 하는지 선택한다.

데이터베이스의 접속 정보(IP와 ID/PW)를 안다면 직접 접속해서 운영할 수도 있으나 보안정책상 해당 기능은 운영자 옵션으로 처리하여 상황에 맞게 운영한다.

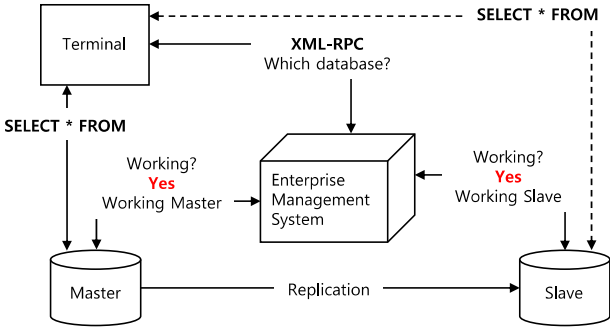


그림 4. System Environment

3.4 System Architecture

본 시스템의 궁극적인 목적은 데이터베이스 서버 그룹 관리를 위한 EMS이다. 시스템이 제공하는 주요 기능은 데이터베이스 상태 관리, HA Group관리, 모듈식 부가기능 반영, 복제(Replication), 샤딩(Sharding) 기능을 지원하며 두가지 방법으로 동작한다.

첫째, 자동 감시 및 운영이다. 본 EMS는 heartbeat과 DRBD기능을 이용해서 3초 간격으로 관리중인 DBMS의 상태를 체크하고 동기화한다. Server monitor는 이러한 상태를 확인하며 이슈가 발생되면 Failure Tracker에게 정보를 전달한다. Failure Tracker는 발생된 이슈에 대해서 히스토리 관리를 하는데 신규로 발생된 오류는 parser->handler->executor->scheduler 순으로 조치하며, 기존 발생된 오류는 EMS내 DBMS와 상태를 비교하고 업데이트하며 관리한다.

둘째, 수동 감시 및 운영이다. 본 EMS는 운영자가 직접 개입하여 수동으로 절체하고 운영할 수 있다. 콘솔을 통해 EMS에 접속하여 관련 명령어를 입력하면 parser->handler->executor->scheduler 순으로 입력한 명령어를 실행한다.

만약 관리중인 DBMS에 이슈가 발생시 자동으로 3.2 System Working Status대로 상태가 변경되어 관리되며 시스템 상태는 수동으로 변경 가능하다.

다양한 DBMS의 부가기능 지원을 위해 Event 모듈을 통해 관리하는데 Parser와 handler에 신규 명령어를 등록하고 Event 모듈에 반영 후 사용한다.

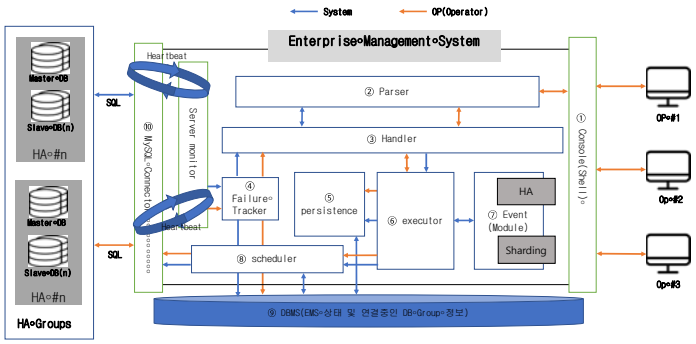


그림 5. System Architecture

이 EMS는 Agentless방식으로 Connector에 해당 코드를 포함하여 배포하며, Agent 설치 없이 CloudDB VM(Virtual Machine) instance 생성시 함께 생성되어 동작될 수 있도록 설정하여 운영한다.

4. 결론

Cloud는 불특정 다수의 사용자들을 위한 서비스이다. 시스템을 사용하는 방식이 다양하고 비용에도 민감하다. 또한, 제약사항 및 보안문제에도 민감하게 대응해야 한다. 이러한 점에서 본 연구에서 구축한 EMS는 Cloud환경에서 관리되는 DBMS의 여러 제약사항을 해결하고 다수의 사용자 및 운영자 만족을 위해 만들어 졌다. CLI에 익숙한 운영자들을 위해 구축된 시스템이기에 UI/UX적인 부분이 부족하지만 향후 보완하여 발전시킬 계획이다.

5. 참고문헌

[1]Gartner, “The Future of the DBMS Market Is Cloud” by Donald Feinberg, Merv Adrian and Adam Ronthal. 2019

[2]김동호, 이정훈, and 박양표. "기업의 Cloud Computing 서비스 도입의도에 영향을 미 치는 Cloud Computing 특성 요인에 관한 연구." 한국전자거래학회지 17.1, 2012

[3]윤용익, 김스베틀라나, “모바일 클라우드 컴퓨팅 기술 동향”, 정보통신산업진흥원 주간기술동향, 통권 제1439호, 2010.

[4]이주영, “클라우드 컴퓨팅의 특징 및 사업자별 제공 서비스 현황”, 정보통신정 책연구원 방송통신정책, 제22권, 제6호, 2010.

[5]Armbrust, M., Fox. A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M., “Above the Clouds : A Berkeley View of Cloud Computing,” UC Berkeley TR 2009,

[6]Oracle, Cloud Generation 2: Larry Ellison Keynote at Oracle OpenWorld 2018

[7]Richter, Wolfgang, et al. "Agentless cloudwide streaming of guest file system updates." 2014 IEEE International Conference on Cloud Engineering (IC2E). IEEE, 2014.

[8]박성종. "임무컴퓨터를 위한 고가용 시스템의 구현 및 성능분석", 충남대학교 석사학위논문, 2009

[9]http://linux-ha.org

[10]http://linux-ha.org/Heartbeat

[11]http://linbit.com/, http://drbd.org/

[12]A, Robertson, “Linux-HA Heartbeat System Design,” USENIX, pp 305-316, 2000.

[13]최종명, 한주현, 최재영, “Diehard:인터넷 서비스를 위한 N-way 고가용성 시스템”, 정보과학회 논문지, 제28권, 제 8호, pp.390-398, 2001.

[14]D.P.Siewiorek, “Architecture of fault-tolerant computers: an historical perspective.” IEEE, Vol79, No.1. pp1710-1734, 1991.

[15]http://download.nust.na/pub6/mysql/doc/mysql-ha-scalability/en/ha-heartbeat.html

헬리콥터 자동비행조종컴퓨터 검증을 위한 시나리오 기반 시험 사례

김재만 ○ 이선아

한국항공우주산업㈜ ○ 경상대학교

Jmkim24@daum.net , saleese@gun.ac.kr

Scenario-based Test Cases for the Verification of an Automatic Flight Control Computer

Kim, Jaeman ○ Lee, Seonah

Korea Aerospace Industries, LTD. ○ Gyeongsang National University.

요 약

자동비행조종컴퓨터는 많은 기술적 이슈의 엄정한 개발 및 시험 단계를 거쳐 개발된다. 이러한 단계에서 도리어 사용자의 관점에서의 사용 시나리오에 대한 기술과 평가는 소홀할 수 있다. 본 연구에서는 자동비행조종컴퓨터 검증을 위한 시나리오 기반 시험 사례를 개발한다. 즉 자동비행조종컴퓨터의 기능흐름에 대하여 시나리오 기반의 시험 사례를 적용하여 시험결과를 도출한다. 시험결과 대부분의 시험 사례는 통과하였다. 그러나, 상식적인 사용 사례로 보았을 때 통과해야 하나 실패한 시험 사례가 있었다. 우리는 우리가 제시한 자동비행조종컴퓨터의 시나리오 기반 시험 사례가 다른 자동비행조종컴퓨터의 검증을 위해서도 유용하게 활용될 수 있다고 기대한다.

1. 서 론

헬리콥터의 자동비행조종시스템에서 가장 중요하며 핵심이 되는 자동비행조종컴퓨터의 시험은 다음과 같다. 첫째, 비행운용프로그램의 소프트웨어 단위에서 검증, 둘째, 하드웨어 단위에서 검증, 제어법칙 단위에서 검증, 셋째, 제어법칙과 비행운용프로그램 통합 검증, 넷째, 하드웨어와 비행운용프로그램의 통합 검증, 다섯째, 시스템 운용을 위해 연동되는 주요 구성품들도 포함하는 시스템 통합 검증 등이다. 이처럼 헬리콥터의 자동비행조종시스템을 위하여 많은 시험들이 단계별로 수행된다. 그런데 만약 넷째 혹은 다섯째 단계의 시스템 통합 검증을 위한 테스트 진행 중 기능적인 결함이 발견되었을 시에는 원인을 파악하고 원인이 되는 단위에서의 보완 및 재 검증이 수행되고 다음 단위에서의 검증을 거쳐서 다시 시스템 통합 검증을 수행하는 등의 반복되는 복잡한 과정을 거치게 된다. 이러한 검증방법은 많은 비행조종컴퓨터를 개발하는 과정에서 적용되는 설계표준에 따른 검증절차이다. 하지만, 합리적인 시험 시나리오를 연구하여 적용한다면 좀 더 앞 단계에서의 효율적인 검증과 사용자 관점에서의 좀 더 체계적인 시험 절차로 발전해 나갈 수 있을 것으로 예상된다.

본 연구에서는 비행운용프로그램이 장입된 자동비행조종텀퓨터가 요구도를 충족함을 검증하기 위한 방법으로 시나리오 기반 시험을 제안한다. 여기서 제안하는 시험 시나리오는 자동비행조종컴퓨터와 비행운용프로그램 단위의 검증과 시스템 통합 검증의 중간 과정이다. 이 과정에서 시스템 전반적인 기능연동에 대한 요구도 충족 확인을 위한 유스케이스 시나리오와 품질속성 시나리오를 작성한 후, 검증을 위한 시험을 수행할 시험 사례를 구체화하였다. 이러한 사용 시나리오 관점에서의 시험 사례는 또한 사용 측면에서의 누락된 부분은 없는지를 검토할 수 있도록 돕는다.

본 연구에서 각 시험 사례별로 시험을 수행한 결과, 99% 수준의 합격률이 나왔다. 그러나 하나의 시험 사례는 실패임을 확인하였다. 본 연구를 통해 도출된 시험 결과와 같이 헬리콥터 자동비행조종컴퓨터의 요구도를 기반으로 선정된 시험 시나리오에 대한 연구사례는 다양한 유사장비의 요구도 검증에 활용이 유용할 것으로 판단된다.

2 관련 연구

2.1 소프트웨어 검증

기존 연구[1~4]에서는 소프트웨어 단위시험과 구조시험 등을 통해 커버리지를 충족하는지 시험하고 분석한 결과를 제시하고 있다. 반면, 본 연구에서는 소프트웨어 단위에서는 수행할 수 없는 시스템 사용 시나리오를 기반으로 한 시험 사례를 제시하여 전반적인 동작 흐름에서의 검증결과를 보여준다.

2.2 통합시험을 통한 시스템 검증

기존 연구[5~9]에서는 비행조종컴퓨터 단위에서의 시험 이후, 시스템을 통합한 시험환경(HILS)을 통해 시스템 레벨에서의 검증한 결과를 제시하고 있다. 본 연구에서는 하드웨어와 비행운용프로그램(OFP)이 포함된 소프트웨어를 통합한 자동비행조종컴퓨터 단위에서 시스템 통합시험(HILS) 시험 진입 이전에 시스템 기능흐름 시험결과를 미리 점검 가능할 수 있는 시나리오 기반 시험 사례를 보여준다.

3 유스케이스 시나리오 시험 사례

3.1 유스케이스 시나리오 명세서

유스케이스 시나리오는 사용자가 어떤 것을 사용한 사례를 수행하기 위한 시나리오로 작성하는 것이다. 시스템 개발자 보다는 사용자 입장에서 이해하기 쉽고 간단하게 작성하는 것이 중요하다.

자동비행조종컴퓨터의 기능흐름을 기반으로한 유스케이스 다이어그램은 그림 1과 같다.

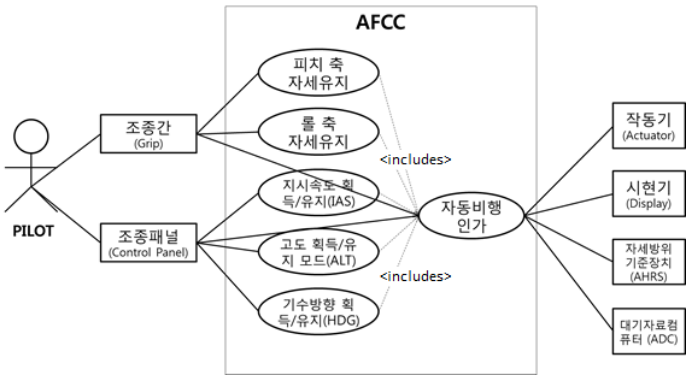


그림 1. 유스케이스 다이어그램

위의 그림에서 'AFCC'는 자동비행조종컴퓨터이며, 'IAS'는 지시속도 획득/유지 모드이며, 'HDG'는 기수방향 획득/유지 모드이며, 'ALT'는 기압고도 획득/유지 모드이며, 'AHRS'는 자세방위 기준장치이며, 'ADC'는 대가자료 컴퓨터이다.

자동비행조종컴퓨터의 유스케이스 간략 설명은 표 1과 같다.

표 1. 유스케이스 간략 설명표

번호	유스케이스명	개략설명
UC01	자동비행 인가 기능	조종사에게 자동비행 기능을 제공
UC02	피치축 자세유지 기능	조종사에게 피치축 자세유지 기능을 제공
UC03	롤축 자세유지 기능	조종사에게 롤축 자세유지 기능을 제공
UC04	IAS 기능	조종사에게 속도 획득/유지 기능을 제공
UC05	ALT 기능	조종사에게 고도 획득/유지 기능을 제공
UC06	HDG 기능	조종사에게 기수방향 획득/유지 기능을 제공

6가지의 유스케이스 중, 피치축 자세유지 기능 유스케이스(UC02)에 대한 세부 명세서는 표 2와 같다.

표 2. 피치축 자세유지 기능 유스케이스 명세서(UC02)

번호	유스케이스	피치 축 자세유지 기능
	개요	조종사에게 피치 축 자세유지 기능을 제공
	관련 엑터	- 주엑터: 조종사(PILOT) - 보조엑터: 조종패널(Control Panel), 자동비행조종 컴퓨터(AFCC), 시현기(Display), 작동기(Actuator), 대기자료 컴퓨터(ADC), 자세방위 기준장치(AHRS), 조종간(Grip)
	선행 조건	자동비행 인가 기능 유스케이스가 실행됨. 자세방위 기준장치가 ON되어 AFCC와 시현기로 피치/롤 자세 및 기수방향에 대한 센싱정보를 지속적으로 보낸다.
1	정상 시나리오	1. 조종사는 조종간의 사이클릭 Beep 스위치를 피치축 전방으로 4초 입력한다. 2. 조종간은 조종입력에 따른 이산신호(GND)를 AFCC로 출력한다. 3. AFCC는 입력 입력신호로 항공기 피치축 자세 기준 값을 변경하고 ARINC429 신호(303 LABEL)를 작동기로 출력한다. 4. 작동기는 받은 입력신호로 명령 값에 해당되는 위치로 작동한다. 5. 시현기는 자세방위 기준장치에서 받은 항공기 피치축 자세 값을 자세 지시계에 각도 값으로 시현한다. 6. AFCC는 자세방위 기준장치에서 받은 항공기 피치축 자세 값이 기준 값과 일치하도록 정상작동되고 있음을 확인하고 ARINC429 신호(061 LABEL)를 시현기로 출력한다. 7. 시현기는 받은 입력신호로 피치축 자세유지 기능이 정상운용되고 있는 상태를 표현하기 위하여 노란색 "P"나 노란색 "OFF"가 사라지도록 시현한다. 8. 조종사는 시현되는 정보를 통해 피치축 자세유지 기능이 정상작동 했음을 확인한다.
	대체 시나리오	해당사항 없음.
시험 사례		해당사항 없음.

3.2 유스케이스 시험 사례

자동비행조종컴퓨터의 피치축 자세유지 기능 유스케이스(UC02)의 시험 사례는 TC7 ~ TC17까지 총 11개로 구성되며 간략 설명은 표 3과 같다.

표 3. UC02 시험 사례

유스케이스 시나리오ID	시험 사례	입력1	입력2	기대 값
피치 축 자세유지 기능 (UC02)				
UC02-1	TC7	TC0 실행	-	피치 기준 값 = 0도 유지
	TC8	TC0 실행	피치축 BEEP 전방 1초 입력	피치 기준 값 = 0도에서 -2도로 변경
	TC9	TC0 실행	피치축 BEEP 후방 1초 입력	피치 기준 값 = 0도에서 2도로 변경
	TC10	TC0 실행	피치축 BEEP 전방 2초 입력	피치 기준 값 = 0도에서 -4도로 변경
	TC11	TC0 실행	피치축 BEEP 후방 2초 입력	피치 기준 값 = 0도에서 4도로 변경
	TC12	TC0 실행	피치축 BEEP 전방 7.5초 입력	피치 기준 값 = 0도에서 -15도로 변경
	TC13	TC0 실행	피치축 BEEP 후방 7.5초 입력	피치 기준 값 = 0도에서 15도로 변경
	TC14	TC0 실행	피치축 BEEP 전방 15초 입력	피치 기준 값 = 0도에서 -30도로 변경
	TC15	TC0 실행	피치축 BEEP 후방 15초 입력	피치 기준 값 = 0도에서 30도로 변경
	TC16	TC0 실행	-	피치 기준 값 = 0도 유지
	TC17	TC0 실행	롤축 BEEP 좌측 1초 입력	피치 기준 값 = 0도 유지

(*) TC0 실행: 시뮬레이션 실행 + 자동비행 ON/OFF 스위치 누름 = 자동비행 기능 인가

4 품질속성 시나리오 기반 시험 사례

4.1 품질속성 시나리오

특정품질에 대한 요구도를 명세하기 위한 것이 품질속성 시나리오이다. 품질속성 6개 항목을 사용한 시라리오이며, 품질속성 6개 요소는 그림 2에서 보인바와 같다.

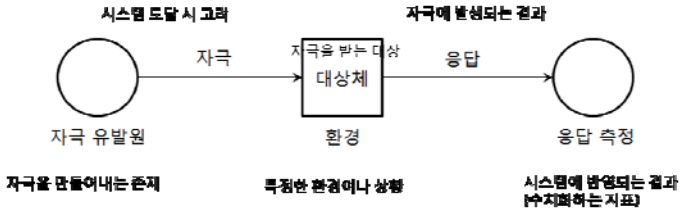


그림 2. 품질속성 시나리오 항목

그림 2에 따라 작성한, 자동비행조종컴퓨터의 우선순위가 높은 3개의 주요 품질속성 시나리오는 아래의 표 4와 같다.

표 4. 품질속성 시나리오 작성 및 순위 결정

품질 속성	속성 상세화	시나리오	우선순위	
			중요도	난이도
안전성	QS1. 알람 속도 최소화	조종사는 정상적으로 운용중인 자동비행조종컴퓨터의 항공기 전원 차단에 따른 결함을 자동비행조종컴퓨터는 신속히 인지하여 알람신호를 1초 이내 출력하고 출력된 신호를 시현기가 입력받아 시현하기를 원한다.	H	M
안전성	QS2. 알람 시간 유지	조종사는 비행상태로 정상 운용중인 자동비행 인가조건에서 인가된 자동비행 기능이 결함에 의해서 해제되면 자동비행조종컴퓨터는 조종사가 충분히 인지할 수 있도록 10초 동안 알람신호를 출력하고 출력된 신호를 시현기가 입력받아 시현하기를 원한다.	H	M
운용성	QS3. 선택된 센서 사용	조종사는 비행 중, 단일 센서 결함 시 원하는 정상상태의 센서를 선택하기 위해 재형성 유닛의 선택 스위치를 돌리고 센서 선택 신호를 시현기가 받아 시현하고 선택된 센서 선택신호를 출력시키고 출력된 신호를 자동비행조종컴퓨터에서 받아 자동비행 기능에 선택된 센서가 사용하기를 원한다.	H	H

① QS1. 알람속도 최소화

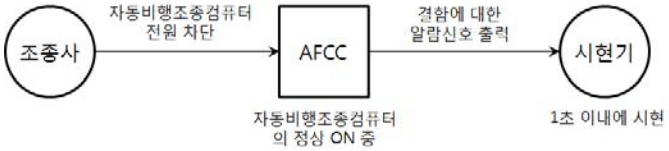


그림 3. QS1 품질속성 시나리오

알람속도 최소화 품질속성 시나리오는 그림 3과 같으며, 조종사 편의를 위해 자동비행을 사용한 비행 중, 조종사가 조종간이나 패달을 조작하지 않고 시현되고 있는 화면만을 모니터링하면서 운용중인 자동비행이 정상작동하고 있는지를 확인한다. 이러한 상황에서 자동비행조종컴퓨터의 결함이 발생되는 경우는 헬리콥터를 치명적인 위험상황을 유발할 수 있기 때문에 결함을 조종사에게 알려주는 알람 속도는 최소화되어 신속하게 조종사가 결함을 인지하여 수동으로 조종간과 패달을 조작하여 헬리콥터를 조종하여 위험 상황이 발생 되지 않도록 해야 한다.

② QS2. 알람시간 유지

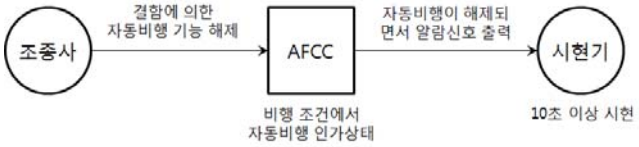


그림 4. QS2 품질속성 시나리오

알람시간 유지 품질속성 시나리오는 그림 4와 같으며, 조종사에게 신속한 알람도 중요하지만, 조종사가 충분히 결함 사항을 확인할 수 있는 10초 동안 유지되어야 한다.

③ QS3. 선택된 센서 사용

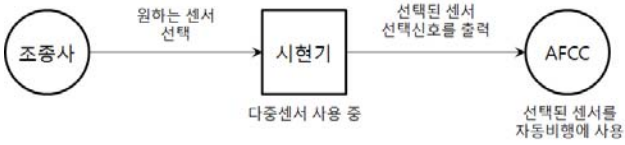


그림 5. QS3 품질속성 시나리오

선택된 센서 사용 품질속성 시나리오는 그림 5와 같으며, 조종사가 헬리콥터 운용중 이중으로 구성된 센서에서 단일 결함이 발생되었을 경우, 조종사가 판단하여 한 대의 센서를 선택하여 시현되도록 하고 자동비행조종컴퓨터는 조종사가 선택한 센서정보를 입력받아 사용하여야 한다.

4.2 품질속성 시험 사례

자동비행조종컴퓨터의 품질속성 시나리오 기반의 시험 사례는 표 5와 같다.

표 5. 품질속성 시나리오 기반 시험 사례

유스케이스 시나리오ID	시험 사례	입력1	입력2	기대 값
알람 속도 최소화 (QS1)				
QS1	QTC1	AFCC 전원 ON	AFCC 전원 OFF	1초 이내 알람 시현
알람 시간 유지 (QS2)				
QS2	QTC2	UC01 실행	자동비행 ON/OFF 스위치 누름	알람 시간 10초 유지
선택된 센서 사용 (QS3)				
QS3	QTC3-1	AHRS 'N' 선택	AHRS-N 신호 출력	AHRS N 사용
	QTC3-2	AHRS '1' 선택	AHRS-1 신호 출력	AHRS 1 사용
	QTC3-3	AHRS '2' 선택	AHRS-2 신호 출력	AHRS 2 사용
	QTC3-4	ADC 'N' 선택	ADC-N 신호 출력	ADC N 사용
	QTC3-5	ADC '1' 선택	ADC-1 신호 출력	ADC 1 사용
	QTC3-6	ADC '2' 선택	ADC-2 신호 출력	ADC 2 사용

① 알람속도 최소화(QTC1.) 시험 사례

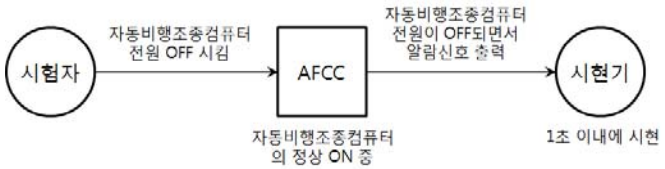


그림 6. QTC1 시험 사례

알람속도 최소화 시험 사례는 그림 6과 같으며, 시험자는 전원이 ON되어 정상적으로 운용중인 자동비행조종컴퓨터의 전원을 차단시켜 자동비행조종컴퓨터로부터의 결함에 대한 알람신호 시간을 점검하여 1초 이내 시현되는 것으로 확인하고자 한다.

② 알람시간 유지(QTC2.) 시험 사례

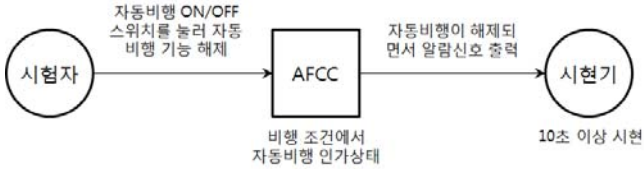


그림 7. QTC2 시험 사례

알람시간 유지 시험 사례는 그림 7과 같으며, 시험자는 비행조건에서 자동비행 기능 인가를 위해

UC01을 실행시킨다. 인가된 자동비행 상태가 해제되도록 다시 자동비행 ON/OFF 스위치를 누르면 자동비행조종컴퓨터로부터의 알람신호 출력시간이 10초인지를 점검하여 충분한 시간 동안 시현되는 것으로 확인하고자 한다.

③ 선택된 센서 사용(QTC3.) 시험 사례

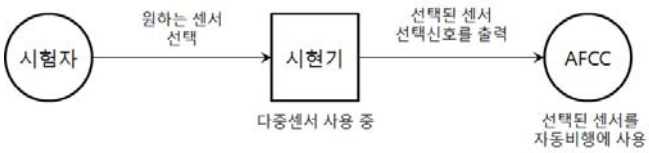


그림 8. QTC3 시험 사례

선택된 센서 사용 시험 사례는 그림 8과 같으며, 시험자는 다중 센서 중 원하는 센서를 선택하고 선택된 신호를 시현기에서 받아 다중 센서 시현에서 선택된 센서 정보를 시현하고 선택된 센서 선택신호를 자동비행조종컴퓨터에서 받은 것을 점검하여 자동비행 기능에 선택된 센서정보를 사용하고 있는 것으로 확인하고자 하였다. 본 시험 사례에서의 센서는 자세방위 기준장치(AHRS)와 대기자료 컴퓨터(ADC)이므로, AHRS와 ADC 2개의 선택스위치를 사용하였다. 두 가지 센서에 대해서 각각 다중 센서를 선택하여 사용하기 위한 'N' 선택, 1번 센서를 선택하여 사용하기 위한'1' 선택, 2번 센서를 선택하여 사용하기 위한'2' 선택을 스위치를 통해서 수행하였다.

5 시험 환경

자동비행조종컴퓨터의 유스케이스와 품질속성 시나리오 기반의 시험 사례별 시험을 위한 시험환경이다. 자동비행조종컴퓨터가 정상적인 비행조건으로 인지되도록 시뮬레이션되고 자동비행을 위해 필요한 입출력 신호의 인터페이스를 위한 환경이다. 전원 공급장치(PSU), 자동비행조종컴퓨터 Breakout Box, Arinc-429 입출력 처리보드, 노트북 등으로 아래의 그림 9와 같이 구성된다.

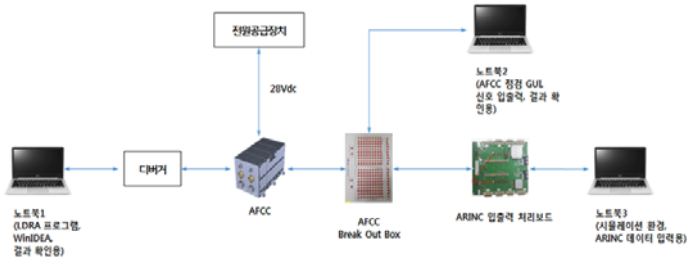


그림 9. 시험환경

6 시험 결과

본 연구에서는 자동비행조종컴퓨터의 요구도 충족여부 검증을 위해 시나리오 기반 기법을 적용한 시험결과, 표6과 같이 총 104개의 시험 사례에 대해 103개의 합격과 1개의 불합격이다. 시나리오 기반 시험 방법을 적용한 결과 99%의 높은 성공율 확인으로 요구도 기반의 시나리오, 시험 사례의 기능흐름이 잘 반영된 결과이다. 1개의 불합격은 비행운용프로그램에 반영된 제어법칙의 피치다운 리미트 사항을 시험 시나리오와 시험 사례 작성 시 충분한 검토되지 않고 미 반영된 사항이다. 따라서 시나리오 기반 시험기법을 사용한 요구도 검증은 높은 성공율이 보장되지만, 하드웨어, 소프트웨어, 제어법칙과 시스템을 구성하는 각 구성요소들에 대한 정보의 정확한 이해와 올바른 반영이 필요하다.

표 6. 시험결과

유스케이스 시나리오 ID	시험 사례	결과 (P=Pass, F=Fail)
자동비행 인가 기능 (UC01)		
UC01-1	TC0 / TC1 / TC2	P / P / P
UC01-2	TC3 / TC4	P / P
UC01-3	TC5 / TC6	P / P
피치 축 자세유지 기능 (UC02)		
UC02-1	TC7 / TC8 / TC9 / TC10 / TC11 / TC12 / TC13 / TC14 /TC15 / TC16 / TC17	P / P / P / P/ P / P / P / F / P / P/ P
롤 축 자세유지 기능 (UC03)		
UC03-1	TC18 / TC19 / TC20 / TC21 / TC22 / TC23 / TC24 / TC25 / TC26 / TC27 / TC28	P / P / P / P / P / P / P / P / P / P / P
IAS 기능(UC04)		
UC04-1	TC29 / TC30 / TC31	P / P / P
UC04-2	TC32 / TC33 / TC34 / TC35 / TC36 / TC37 / TC38 / TC39 / TC40 / TC41 / TC42	P / P / P / P / P / P / P / P / P / P / P
UC04-3	TC43 / TC44 / TC45	P / P / P
UC04-4	TC46 / TC47 / TC48	P / P / P
UC04-5	TC49 / TC50	P / P
ALT 기능(UC05)		
UC05-1	TC51 / TC52 / TC53	P / P / P
UC05-2	TC54 / TC55 / TC56 / TC57 / TC58 / TC59 / TC60 / TC61 / TC62	P / P / P / P / P / P / P / P / P
UC05-3	TC63 / TC64 / TC65	P / P / P
UC05-4	TC66 / TC67 / TC68	P / P / P
UC05-5	TC69 / TC70	P / P
HDG 기능(UC06)		
UC06-1	TC71 / TC72 / TC73 / TC74 / TC75 /	P / P / P / P / P /

	TC76 / TC77 / TC78 / TC79 / TC80 / TC81 / TC82	P / P / P / P / P / P / P
UC06-2	TC83 / TC84 / TC85 / TC86	P / P / P / P
UC06-3	TC87 / TC88 / TC89	P / P / P
UC06-4	TC90 / TC91 / TC92	P / P / P
UC06-5	TC93 / TC94 / TC95	P / P / P
UC06-6	TC96 / TC97 / TC98	P / P / P
UC06-7	TC99 / T100	P / P
알람 속도 최소화 (QC1)		
QS1	QTC1	P
알람 시간 유지 (QC2)		
QS2	QTC2	P
선택된 센서 사용 (QC3)		
QS3	QTC3	P

7 결 론

본 연구에서는 자동비행조종컴퓨터의 요구도 충족 여부 검증을 위하여 시나리오 기반 시험 방법을 적용하였다. 유스케이스 시나리오를 통한 101개의 시험 사례에 품질속성 시나리오를 통한 3개의 시험 사례를 더하여 총 104개의 시험 사례가 도출되었다. 시험 사례별로 시험을 수행한 결과, 총 104개의 시험 사례 중 103개가 합격을 하고 1개의 항목이 불합격되었다.

시나리오 기반 시험 방법을 적용한 결과 약 99%의 높은 성공률이 확인되었고, 이러한 높은 성공률은 요구도 기반의 시나리오 및 시나리오 기반의 시험 사례가 하드웨어, 소프트웨어, 제어법칙 및 시스템의 전반적인 기능 흐름이 잘 반영되었음을 의미한다. 유일하게 실패한 항목은 비행운용프로그램에 반영되어 있는 제어법칙이 피치업 모션과는 다르게 피치다운 모션에 리미트를 적용한 것을 시나리오 및 케이스 작성자가 알지 못하고 피치업과 피치다운 모션 모두 리미트가 없는 것으로 작성한 것이 원인이었다.

시나리오 기반 시험 방법을 검증에 적용하면 높은 성공률이 보장되지만, 위 합격/불합격 분석에서 보듯이 하드웨어, 소프트웨어, 제어법칙 및 시스템을 구성하는 주요 각 구성요소에 대한 정보 (기능/성능, 동작 원리, 시스템 운용 개념, 제한사항 등)의 정확한 이해 및 적절한 반영이 선결되어야 한다.

본 연구에서는 하드웨어와 OFP가 포함된 소프트웨어를 통합된 자동비행조종컴퓨터 단위에서 시스템 통합 시험환경(HILS) 시험 진입 이전에 시스템 기능흐름에 대한 시험결과를 미리 점검 가능할 수 있는 시나리오 기반 시험 사례이다. 이 같은 시나리오를 기반으로 한 시험을 함으로써, 기존의 시스템 통합시험 중 확인된 결함을 고장탐구하고 고장 원인에 대한 오류수정 및 점검을 완료한 후, 다시 시스템 통합시험을 수행하기 위한 4주 이상의 리드타임을 1주 이내로 줄일 수 있는 효율적인 검증절차로 합리적인 시험 시나리오 연구사례라는 것을 확인할 수 있었다.

본 연구를 진행하면서 다양한 시험 시나리오와 시험기법을 이해하고 적용해볼 수 있었던 계기가 되었다. 향후 누구나 쉽게 이해하고 요구도 충족은 물론, 시험 성공률도 높은 새로운 시험 시나리오와 시험기법을 만들기 위한 연구를 향후 진행할 예정이다..

8 참조 문헌

[1] 이영호, 고성희, 기자영, 김보경, 김병현, 2016, “DO-178 절차에 따른 항공용 터보팬 엔진의 EECU 소프트웨어 검증시험에 관한 연구 - Part 2: 요구도 기반 시험”, 한국추진공학회 학술대회논문집, pp 530 - 534

[2] 장정훈, 장주수, 강유선, 2016, “DO-178C(항공용 소프트웨어 감항인증)에 적합한 요구도 기반 시험 절차 및 Test Coverage 달성 방안 연구”, 한국항공우주학회 학술발표회 초록집, pp 1184 - 1185

[3] 이지은, 신선영, 장순용, 2017, “비행제어 소프트웨어 동적시험 커버리지 달성방안 연구”, 항공우주시스템공학회 2017년도 추계학술대회

[4] 윤성훈, 김여울, 황선유, 2015, “모델기반설계 소프트웨어 신뢰성 검증에 관한 연구”, 한국항공우주학회 학술발표회 초록집, pp 1101 - 1104

[5] 변진구, 윤형식, 허기봉, 권종광, 박준현, 2008, “비행조종컴퓨터 하드웨어 검증시험 결과” 한국항공우주학회 학술발표회 초록집, pp 1023 - 1026

[6] 변진구, 허기봉, 이광현, 석진영, 2016, “무인항공기 비행제어 HILS 시험환경 연구”, 한국항공우주학회지, 44(4), pp 316-323

[7] 김진완, 2016, “국내외 저가형 비행제어장치 개발 현황”, 항공우주시스템공학회 2016년도 추계학술대회

[8] 김응태, 성기정, 이상종, 장재원, 최형식, 이장호, 김재은, 이석천, 이승현, 윤한수, 2009, “다중화 FBW 비행제어시스템 핵심기술 연구“, 국가연구개발 보고서

[9] 신성식, 문정호, 노은정, 2011, ”근접감시 무인항공기의 비행제어시스템 개발“, 한국방위산업학회, 한국방위산업학회, 제18권, 제2호

세그멘테이션 기반 검출과 어텐션 기반 인식을 활용한 유통기한 OCR

문형진° 나병진
에스에스지닷컴
{mhjin, nbjin}@ssg.com

OCR of BestBefore Using Detection based on Segmentation and Recognition based on Attention

HyoungJin Moon° Byeongjin Na
SSG.COM

요 약

과거부터 많은 OCR 알고리즘이 연구되었고 딥러닝의 발전과 함께 비약적인 발전이 일어났다. 이제 학문적인 연구를 넘어서 산업에 적용 가능한 수준이 되었다. 이커머스 분야도 사람이 상품을 직접 보고 판단하고 기록하는 작업을 한다. 우리 알고리즘은 이러한 병목현상을 해소할 목적으로 활용할 수 있다. 본 연구에서는 SSG.COM 자동화 물류 센터에서 상품이 입고될 때 상품의 유통기한을 사람이 직접 눈으로 확인하고 입력하는 비효율적인 프로세스의 개선을 위해서, 상품의 이미지로부터 유통기한 패턴의 글자를 검출하고, 인식하여 자동화에 기여할 수 있는 알고리즘을 제안하고자 한다. 우리의 알고리즘은 크게 유통기한 검출과 유통기한 인식으로 구성된다. 검출기가 검출한 유통기한 영역을 인식기가 입력으로 받아 글자를 인식한 후, 후처리를 통해 글자별 년, 월, 일 등의 태그를 출력하여 유통기한 날짜를 추출하게 된다. 이 알고리즘은 유통기한뿐만 아니라 구조화된 텍스트 정보를 가지는 텍스트 이미지에 대해 적용할 수 있다.

1. 서 론

최근 딥러닝의 비약적인 발전으로 신뢰성 있는 결과를 얻게 됨에 따라, 많은 분야의 산업에서 고도화된 기술을 활용한 서비스가 운영되고 있다. 식품을 다루는 산업에서 유통기한 관리는 매우 핵심적인 요소 중 하나다. 자동화 물류 센터의 상품 입고 과정에서 여전히 유통기한 등록은 사람의 눈으로 직접 확인하며 수동으로 입력하도록 되어 있다. 이 과정을 효율적으로 개선하고자 OCR 기술을 개발하였다.

우리는 장면 텍스트 검출과 장면 텍스트 인식 기반으로 연구 개발했고 산업에 효과적으로 적용하기 위해 몇 개의 문제를 고민하였다. 첫째, 다양한 유통기한 텍스트 형식으로부터 유통기한 정보를 유연하게 인식할 수 있어야 한다. 둘째, 원하는 글자들의 위치 정보를 잘 전달하여 구조화된 형식의 텍스트 인식에 집중할 수 있도록 해야 한다.

우리는 위 두 가지를 고려해서 유통기한 텍스트 검출과 유통기한 텍스트 인식의 2 단계로 이루어진 유통기한 OCR 알고리즘을 제안한다. OCR 유통기한 검출은 촬영된 장면으로부터 유통기한 위치를 추출한다. 추출된 유통기한 텍스트의 위치 정보는 유통기한 인식기의 입력으로 들어가 유통기한 텍스트를 인식한 뒤, 후처리를 통해 텍스트에서 유통기한에 필요한 날짜 정보를 추출하는 순서로 진행된다. 후처리는 개체명인식(Named Entity Recognition) 방법으로 글자마다 날짜 태그를 추출한다.

이 외에도 우리는 데이터 셋 구성에 대하여 논의하고자 한다. 텍스트 학습 데이터 셋은 지도 학습의 특성상 어노테이션 (Annotation)자원이 많이 소요된다. 자원 소모를 줄이기 위해 합성 이미지 데이터 셋을 만들었고 이미지를 합성할 때 고려할 점은 섹션 3 에서 기술하도록 한다. 섹션 4 에서는 알고리즘의 구성, 섹션 5 에서는 실험을 기술한다. 마지막으로 간략한 결론과 향후 계획을 섹션 6 에서 기술한다.

2. 관련 연구

이미지로부터 텍스트를 검출, 인식하는 알고리즘은 주로

CNN (Convolutional Neural Networks)를 기반으로 특징을 추출한다. 마찬가지로 우리의 유통기한 OCR 검출, 인식 알고리즘에서 CNN 구조를 사용하며, 이러한 CNN 기반의 특징 추출 네트워크를 이 논문에서는 베이스 네트워크(base network)라고 부르도록 한다. 우리 검출 모델은 영역 검출 알고리즘을 사용하고 인식 모델은 이미지 특징을 글자로 학습하기 위해서 자연어처리에서 유명한 모델 트랜스포머 (Transformer)[1]를 사용한다.

최근 딥러닝을 활용한 OCR 에서 텍스트를 검출하는 방법은 회귀 손실(regression loss) 기반으로 영역의 위치를 값을 추론하는 방법과 영역을 화소 (pixel) 단위로 분류하는 세그멘테이션 방법으로 나누어진다. 많은 연구들이 바운딩 박스 (bounding box)의 네 점을 예측하여 높은 정확도를 보여주었다[2,3,4,5]. 하지만 이러한 방법은 텍스트의 개수를 제한해야 하고, 많은 양의 텍스트와 다양한 형식의 텍스트를 검출하는 데 불리하다. 이러한 이유로 최근에는 세그멘테이션 방법을 활용하여 불규칙적인 형태의 텍스트를 검출하는 연구들이 활발히 진행되었다[6,7,8,9]. 영역 검출 방법은 모든 텍스트 영역을 커버할 수 있지만, 각각의 글자를 적절하게 연결 혹은 분리하기 위한 노력이 필요하다. 이커머스에서 OCR 검출은 실생활에서 보여지는 다양한 형식의 글자를 검출할 수 있어야 하며, 많은 양의 글자를 검출해야 한다. 따라서, 회귀 방법보다 영역 방법이 더 적합하다. 또한, 우리는 유통기한 인식 이후의 글자인식을 고려해서 각각의 글자 영역 검출 방법을 선택하였다. 각각의 글자 단위로 텍스트가 검출이 되면 다양한 위치로 구성된 글자를 잘 연결할 수 있기 때문이다. 글자 단위의 텍스트 검출은 부족한 학습 데이터 셋, 글자 단위 추론 정확도 등의 이슈가 존재한다. 최근 SOTA 모델 중 하나인 CRAFT[10]는 이 문제점들을 잘 다루고 있다. 우리는 이 구조를 차용하여 여기에 유통기한에 필요한 영역 검출 네트워크를 추가하였다.

OCR 인식모델은 이미지의 특징을 텍스트로 생성하는 “Show and Tell” [11]의 개념을 같이한다. [11] 모델은 이미지 인식과 자연어처리 기반의 모델 구조가 더해져서 설계되었다. 자연어처리는 시계열 기반 데이터로 RNN, GRU, LSTM 의 모델에 대한 성능이 검증되었다.

유통기한의 데이터의 특징은 형식이 구조화 되어있다는 것이다. 우리는 구조화된 형식을 다루기 위해서 Zbigniew Wojna et al. [12] 내용을 참고하였다. [12]에는 공간적인 특징을 어텐션에 추가하면 구조화된 텍스트를 인식할 수 있다고 한다. 우리는 [11, 12]를 기반으로 모델을 설계했고 [12] 에서 다루는 시계열 데이터를 학습하는 어텐션 RNN 모델을 트랜스포머 [1]로 대체하여 실험하였다.

후처리는 자연어처리의 개체명인식을 이용했다. 개체명인식은 문장에서 워드 (word)등 토큰의 유형을 인식하는 알고리즘이다. 우리는 딥러닝 모델을 이용한 개체명인식으로 접근했다. 모델은 자연어처리에서 이슈가 된 BERT [13] 모델에서 아이디어를 받아 단방향 트랜스포머의 인코더 층 (Encoder layer)를 사용하였다. 텍스트의 고정된 길이를 학습하기 때문에 트랜스포머 인코더로 학습 시간, 성능에서 매우 효과적 이었다. BERT 는 양방향 특징을 갖고 있지만 우리 후처리 모델은 비교적 단순한 텍스트의 태그 패턴을 인식하기 때문에 단방향으로 구성되어 있다.

3. 학습 데이터 셋

OCR 알고리즘은 지도 학습이기 때문에 학습에 필요한 데이터 셋과 정답이 필요하다. 실제 상품 이미지의 수집은 짧은 시간 내에 가능하지만, 수집한 사진을 사람이 라벨링 하기에는 한계가 있다. 특히나 글자 단위의 라벨링은 기존의 일반적인 워드 단위 라벨링보다 훨씬 더 많은 코스트가 소비된다. 워드 단위 또는 글자보다 상위 단위의 묶음으로 구성하면 검출과 인식에서 다양한 형식을 인식하기에 어려움이 있기 때문에, 우리는 글자 단위의 학습을 진행하였다.

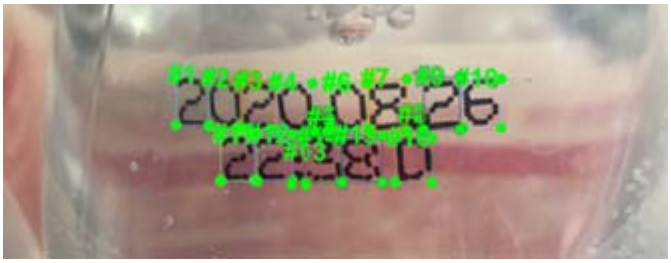


그림 1 유통기한 이미지 라벨링 예시.

OCR 학습 이미지 라벨링 작업은 [그림 1]과 같은 예시 이미지에 대해서 [표 1]과 같이 생성한다. 기본적으로 글자의 라벨과 위치(네 점의 좌표)로 구성되고, 추가적으로

유통기한 데이터 셋을 사용하기 위해서 날짜 태그가 추가되었다. 날짜 태그는 년(Y), 월(M), 일(D), 구분자(SEP), 기타 로 구성된다. 유통기한 상품을 모으는데 자원이 많이 소비되기 때문에 매장에서 유통되는 실제 상품 이미지와 합성데이터로 구성하였다.

표 1: 유통기한 라벨 표기 예시.

label	Loc (x1,y1,...x4,y4)	seq	tag
2	356,516,395,516,395,574,356,574	1	Y
8	585,520,630,520,630,581,585,581	7	M

3.1 OCR 합성 데이터 셋

현재 서비스 중인 카메라 OCR 알고리즘은 기본적으로 한글, 영어, 숫자, 몇개의 기본 기호를 인식하고자 한다. 일어, 중국어는 추후 추가할 예정이다. 무료로 사용 가능한 폰트를 수집하였고, 그 중 한글이 깨지는 폰트를 간단한 영상처리 작업으로 제외시켰다. 유통기한 합성데이터를 만들기 위해서 많이 사용하는 형식으로 22 개 틀을 만들었다. 합성에 사용한 기본적인 폰트는 한글 영어 폰트를 합쳐서 약 1800 개, 추가로 유통기한에서 사용하는 특수한 점자 폰트 2 가지를 영상처리 기법으로 생성하였다. OCR 합성 셋을 구성할 때 이미지 크기와 폰트 크기 각각 생성하는 크기의 구간을 설정 하였다. 이미지의 크기는 너비, 높이 각각 40~1024 를 무작위로 생성하였다. 폰트 크기는 실험으로 20~70 까지 무작위로 생성하였다.

3.2. 유통기한 합성 데이터 셋

우리는 유통기한에서 상당히 빈번하게 사용되는 점자 폰트를 학습시키기 위해 간단한 이미지 프로세싱을 적용하여 텍스트 이미지를 렌더링 했다. 모든 글자에 대해서 점선으로 이루어진 자체 폰트를 다양한 형태로 직접 제작하는 것은 비효율적이므로, 기존의 폰트로 생성된 텍스트를 선정해서 이미지에 추가적인 후처리를 진행하여 점자 폰트와 유사한 패턴의 텍스트 이미지를 생성했다. [그림 2] 왼쪽은 일정한 간격으로 배치된 점 마스크를 이용하여 텍스트 이미지를 렌더링. 오른쪽은 에코폰트를 반전시켜 점으로 이루어진 텍스트 이미지 렌더링 예제다. 검출을 강화하기 위해 텍스트의 글자 배치를 다양하게 하려고 노력했다. 텍스트는 라인단위와 커브드로 구성되고 한 이미지에 각각 반반의 확률로 구성되게 설정하였다.3.3. 실제 상품 데이터 셋

SSG.COM 에서 유통되는 실제 이미지를 촬영하기 위해 가이드 라인을 작성하였다. 빛 반사, 포커싱, 흔들림, 배경(투명한 아이템) 등의 요소를 고려해서 아이템 별 촬영 방법을 작성하였다. 물류 자동화 센터와 이마트 점포에서 신선품, 약품 등 유통기한 관련 아이템 약 100 개 상품으로 유통기한이 표기된 부분을 직접 촬영하여 이미지 데이터 셋을 만들었다. 실제 촬영 데이터 셋은 사람의 수작업을 동반한 라벨링 작업의 한계상 합성 데이터 셋에 비해 적은 수로 이루어져있다. 이를 보완하기 위해, 실제 촬영 데이터 셋에 오그멘테이션 (augmentation) 작업을 진행하였다. 작업은 회전, 어핀 변환(Affine Transformation), 크롭을 무작위로 조합하였다.

그림 2 점자 폰트 생성 예시.

4. 제안기법

OCR 은 검출과 인식으로 구성된다. 검출 모델은 이미지에서 텍스트의 위치를 검출하고 인식 모델은 검출된 영역의 텍스트를 추출한다. 검출 모델은 기존의 회귀 검출방법과 세그멘테이션 방법으로 접근한다. 과거에는 회귀기반 방법이 대세였지만 현재는 세그멘테이션이 대세다. 복잡한 텍스트 형식을 인식할 때 화소 기반 세그멘테이션 방법이 더 효율적이기 때문이다. 회귀검출 방법의 경우 검출하는 개수가 많아 지면 후보 좌표도 늘어나기 마련이다. 이것은 머신 자원 문제로 직결되기 때문에 학습 속도에 영향을 준다. 이미지에서 글자 단위로 검출을 하면 검출의 개수가 비약적으로 늘어나기 때문에 회귀방법은 적합하지 않다고 생각했다. 우리는 회귀기반

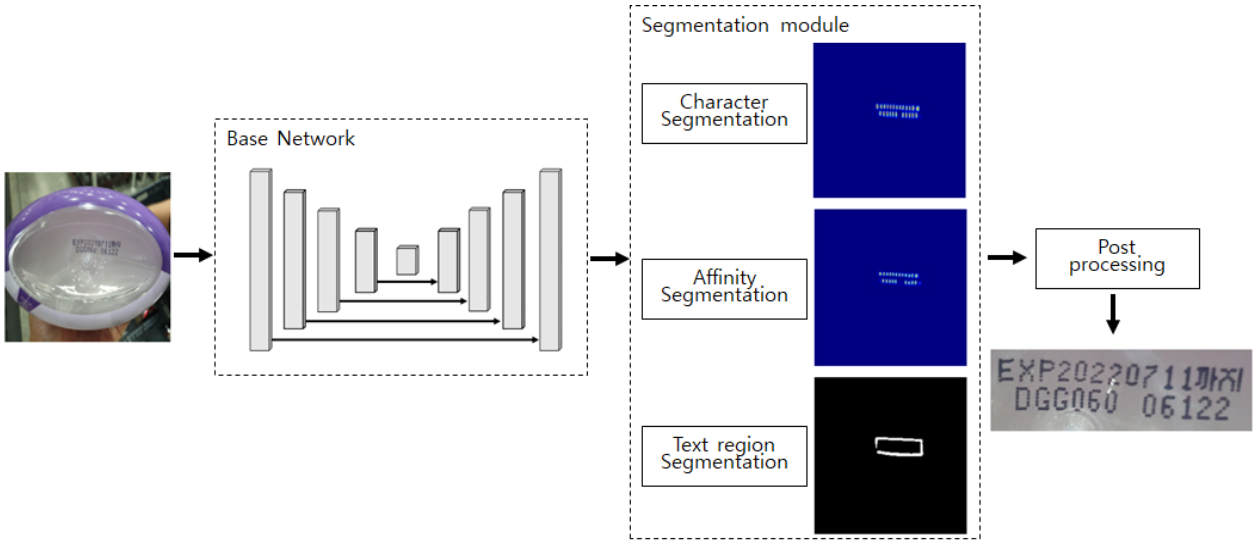


그림 3: 글자 기반 세그멘테이션 검출 모델 구조.

Mask RCNN 과 영역기반 CRAFT 를 학습하면서 평가했다. CRAFT 는 글자기반 세그멘테이션의 속도와 정확도가 좋기 때문에 우리 유통기한 검출에 맞게 네트워크를 수정하였다. 검출의 기본적인 알고리즘은 이미지의 특징을 추출하는 인코더 (base network) 와 특징을 영역으로 재구성하는 디코더 (Decoder)로 구분되어있다.

우리는 다양한 베이스 네트워크를 디코더에 추가해서 실험하였다. 인식 모델 구조는 베이스 네트워크와 시계열 인식으로 구분된다. 베이스 네트워크에서 이미지의 특징을 추출하고 이 특징을 시계열 특징으로 변환한 후 트랜스포머를 학습하였다. 후처리 (post processing)를 적용해서 유통기한 날짜 태그를 추출하였다.

4.1. 검출 모델

텍스트 검출기는 [그림 3]과 같이 크게 베이스 네트워크와 세그멘테이션 모듈, 후처리단으로 구성된다. 베이스 네트워크에서는 이미지를 입력으로 받아 이미지의 특징을 학습하며, 세그멘테이션 모듈에서는 베이스 네트워크를 통해 나온 특징을 이용하여 텍스트 영역인지 아닌지를 판단한다. 세그멘테이션 결과는 후처리 작업을 통해 텍스트 영역의 좌표를 추출한 후 각각의 유통기한 영역을 크롭 (crop)하고 이 결과를 텍스트 인식기로 전달한다.

검출기는 크게 베이스 네트워크와 세그멘테이션 모듈, 후처리단으로 구성된다. 베이스 네트워크에서는 이미지를 입력으로 받아 이미지의 특징을 학습하며, 세그멘테이션 모듈에서는 베이스 네트워크를 통해 나온 특징들을 이용하여 텍스트 영역인지 아닌지를 판단한다. 세그멘테이션 결과 후처리 작업을 통해 텍스트 영역의 좌표를 추출한 후 각각의 영역을 크롭하고, 크롭 이미지들은 텍스트 인식기로 전달한다.

4.1.1. 베이스 네트워크

텍스트 검출기에서는 CNN 기반의 베이스 네트워크를 설계했으며, 정확도 높은 이미지 세그멘테이션을 위해 CNN 의 중간 특징들을 업샘플링 (upsampling)하여 합쳐주는 U-net 의 구조를 채택하였다. 이 구조는 CNN 이 순방향으로 진행됨에 따라 이미지가 축소되어 손실되는 공간 정보를 중간 단계의 특징들과 결합해주면서 얻어낼 수 있기 때문에 보다 정확한 세그멘테이션 맵을 얻을 수 있다.

4.1.2. 세그멘테이션

베이스 네트워크에서 나온 피쳐 맵 (feature map)은 세그멘테이션 모듈에 들어가면서 세 개의 세그멘테이션 맵을 얻는다. 세그멘테이션 모듈은 Character Map 과 Affinity Map 을 출력하는 모듈과 텍스트 영역 맵을 출력하는 모듈로 이루어져 있으며, 각각의 모듈이 동일한 피쳐 맵을 입력으로 받아 세그멘테이션 맵을 출력한다. 앞서 소개한 바와 같이 Character score 와 Affinity score 는 Text region score 와는 다르게 Ground Truth 를 생성하기 때문에 모듈을 분리하여 학습을 진행한다. 두

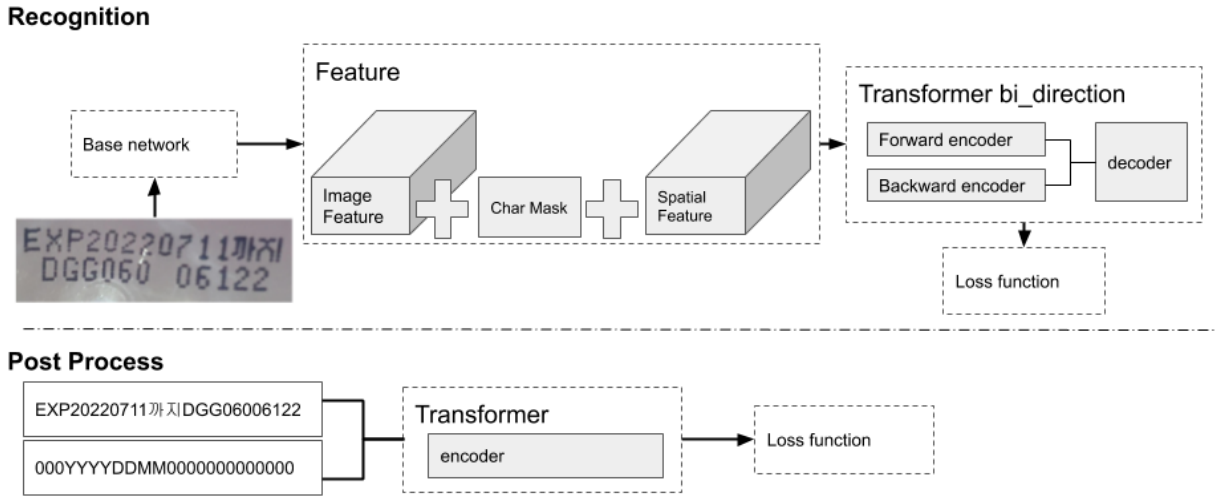


그림 4: 상단 인식 모델 구조와 하단 인식 후처리 모델 구조.

모듈은 층 (layer) 구성이 모두 동일하며 마지막 CNN 층에서 출력되는 채널의 수만 다르다. Text region score 는 유통기한 글자들의 영역 자체를 세그멘테이션 하는 점수로, 유통기한 글자 영역을 유통기한이 아닌 글자들과 명확하게 분리시키기 위해서 추가하였다. 유통기한 이미지의 특성상 이미지 내에 유통기한 뿐 아니라 다양한 글자들이 존재하며 유통기한 정보 외에는 불필요한 글자들이기 때문에 유통기한 영역 내의 글자만 검출할 수 있게 하여 이후에 진행되는 글자 인식에서 불필요한 영역의 글자를 인식하지 않도록 하여 속도 향상에 기여한다.

화소 기반의 Segmentation score 는 OHEM(Online Hard Example Mining)을 적용하여 글자 영역 면적의 불균형을 해소하고 학습이 원활히 진행될 수 있도록 하였다. Score map 을 각각 S 라고 가정한다면, Segmentation score 에 대한 손실 함수는 다음과 같다.

$$Loss = \frac{1}{n}\sum\left(s(p) - s'(p)\right)^2$$

s' 는 Ground-Truth Segmentation Score 를, p 는 pixel 을 나타낸다. 최종 손실 함수 L_s는 다음과 같다.

$$L_s = \alpha L_c + \beta L_a + \gamma L_t$$

L_c 는 Character Loss, L_c 는 Affinity Loss, L_c 는 Text Region Loss 를 나타내며, 각각의 Loss 는 가중치가 부여된 뒤 합산된다.

4.1.3. 검출 후처리

검출 네트워크의 결과로 나온 세그멘테이션 맵으로부터 원본 이미지 내의 글자 영역을 추출하기 위해 후처리 과정을 진행해야 한다. 출력된 맵에서 임계치를 적용해 신뢰성 있는 화소만 골라낸 뒤, 추가적인 영상처리를 통해 글자를 라벨링 하고 글자 영역의 좌표를 구한다.

4.2. 인식 모델

OCR 인식은 [그림 4]와 같다. 전체적으로 [11]과 비슷한 구조지만 시계열 특징 학습 구간에 [1]알고리즘을 적용했다. 현재 OCR 서비스 버전은 단일 라인 텍스트 인식만 가능하다. 우리는 단일 라인 인식이 제한적이라 느꼈기 때문에 다양한 형식으로 구성된 글자를 인식하기 위해서 연구했다. 과거 연구 중 구조화된 텍스트 (structured text)인식 Zbigniew Wojna et al. [12] 을 보면 “Spatially weighted combination” 을 적용해서 문제를 해결하였다. 우리도 Spatial feature 가 유통기한 인식에 도움이 된다고 판단했다. 추가로 글자기반 마스크를 적용하였다. 이 마스크는 검출 모델에서 추출한 글자 세그멘테이션 정보로 0,1 로 구성되어 있다. 이 마스크 정보는 트랜스포머가 각각 글자를 연결하기 위한 어텐션 (attention) 학습에 집중하도록 도와준다. 또한 마스크 적용은 우리의 다음 연구로 각 글자를 크롭 후 인식하여 트랜스포머로 연결하고 재구성 할 수 있다는 확신을 주었다. 인식 모델에서 트랜스포머는 인코더와 디코더로 구성되어 있고, 이미지 특징을 텍스트로 전환하는 기능을 한다. 트랜스포머의 디코더는 고정된 추론 글자 길이를 지정하여 학습 및 추론하였다.

4.2.1. 베이스 네트워크

CNN 기반 베이스 네트워크를 설계할 때 보통 5 단계 영역으로 구성된다. 입력 이미지의 크기를 1/2 로 줄이면서 리셉티브 필드 (receptive field)를 키우고 채널을 늘리면서 특징을 만들어가는 과정인데, 5 단계로 이미지의 크기가 줄어든다. VGG, Resnet, Inception 등등 이미지 분류 모델 베이스 네트워크는 이 5 단계 구조를 갖고 있다. 우리는 여기서 4 번째 단계 이미지 특징을 사용하였다. 다른 단계의 이미지 특징과 2,3,4,5 특징을 연결하여 비교해본 결과 4 번째 단계의 결과가 제일 좋았다. 입력 이미지의 크기는 352, 512 를 테스트 해보고 역시 성능이 좋은 512 로 선택했다.

위에서 언급한 [12]을 참고하여 이미지 특징에 공간특징 (spatial feature) 좌표 정보를 추가하면 인식에 도움이 되기 때문에 X,Y 2 차원 좌표 정보를 공간 정보로 사용했다.

우리는 베이스 네트워크를 선택할 때 속도와 정확도를 고려했다. 속도는 학습 가중치 (weights)수와 비례하고 정확도와 반비례 관계다. 최신 연구 Mingxing Tan et al. [14]는 강화학습을 이용해서 가중치 (weights)수가 비약적으로 줄어든 네트워크 모델을 소개했다. 우리는 이 네트워크를 베이스 네트워크로 학습 및 테스트 진행했다. 이 네트워크는 1~7 까지 시리즈로 구분한다. 우리는 2, 3, 4 시리즈로 학습하였다. 5, 6, 7 시리즈는 4 와 비교할때 정확도의 비약적인 향상이 없기 때문에 테스트를 하지 않았다.

4.2.2 트랜스포머

트랜스포머는 번역에서 탁월한 효과를 발휘한 딥러닝 모델이다. RNN 모델의 단점인 병렬학습이 불가능한 약점을 개선했고 인코더, 디코더 각각 층에서 셀프-어텐션을 학습하기 때문에 보다 긴 문장에서 어텐션을 유지할 수 있는 장점을 갖고 있다. Rami Al-Rfou et al. [15]을 참조하면 LSTM, RNN 은 Long-Term Context 에 대해서 “강력한” 효과를 발휘하지 못한다고 기술한다. 이미지 특징 4 단계 32x32 를 시계열 순서로 변환하면 1024 크기가 된다. 이렇게 매우 긴 벡터를 인식 학습해야 되기 때문에 트랜스포머를 선택하였다.

인코더의 특징은 이미지 특징을 셀프-어텐션 학습했기 때문에 이미지의 특징이라 볼 수 있다. 이미지의 특징을 잘 추출하기 위해서 양방향(bi-directional)로 인코더로 구성하였다. 2 차원 이미지로 설명하면 이미지 좌상단부터 우하단까지 정방향으로 특징을 추출하고 우하단부터 좌상단까지 역방향으로 특징을 추출한다. 이 두 특징을 연결한 후 디코더의 학습 특징으로 제공한다. 양방향과 단방향 학습 결과를 비교해 보면 양방향이 높은 성능을 보였다.

인코더의 특징을 전달받은 디코더는 라벨 (label)의 특징을 셀프-어텐션 학습하고 이 결과를 인코더의 특징과 어텐션을 다시 학습한다. 이렇게 학습한 특징으로 글자를 순서대로 생성하게 된다. 손실 함수는 글자 사전의 카테고리에 대해서 소프트맥스 크로스 엔트로피 (Softmax Cross Entropy)를 사용하였고 손실 함수에 라벨 스무딩 (label smoothing)을 적용하였다. 우리는 시계열 순서에 대해서 (t0, t1, …, tn) 결합 분포를 사용한다.

$$logp(S|I) = \sum_{t=0}^{N} logp(S_t|I, S_0, \ldots, S_{t-1})$$

디코더의 결과를 후처리하여 우리가 필요한 유통기한 정보만 추출한다. 모델은 BERT 를 참고하여 트랜스포머의 인코더 층을 기본으로 사용하였다. 차이점이 있다면 BERT 는 양방향 모델이지만 우리 후처리 모델은 단방향 모델이다. 입력 데이터로 각각 글자와 태그를 쌍으로 전달받아 학습한다. 각 글자의 상관 관계를 셀프-어텐션으로 학습하여 글자와 맵핑되는 태그 유형을 학습한다.

태그 데이터 셋을 생성할 때 [년, 월, 일, 시, 분, 구분자, 기타] 로 구분했다. 실제로 우리가 필요한 태그는 [년, 월, 일, 기타] 이기 때문에 이 네 가지만 추출했다. 추론 결과로 글자마다 네 가지 경우 중 하나로 맵핑된다. 이 태그를 기반으로 우리가 사용할 유통기한을 추출한다.

5. 구 현

우리는 검출, 인식 모델을 Tensorflow 2.0 버전으로 구현했다. 전체 학습 이미지 셋은 오그멘테이션 셋을 포함한 실제 상품 데이터와 합성 데이터로 총 약 10 만장이고 검증셋은 100 장이다. 합성 데이터와 SSG 이미지의 구성은 9.5:0.5 비율이다.

5.1. 검출 모델

머신은 Tesla v100 GPU 세 장을 사용하여 학습을 진행하였다. 베이스 네트워크는 VGG16, ResNet, EfficientNet 을 바꿔가며 학습해 본 결과, EfficientNet 이 가장 좋은 성능을 보였다. 학습 시에 입력 이미지는 비율을 유지한 채 긴 쪽을 1024 로 리사이징하였고, 미니 배치 크기는 (batch size) 8, 웨이트 디케이 (weight decay)는 0.0001, 학습률 (learning rate)은 0.0001 으로 학습하였다.

5.2. 인식 모델

Tesla v100 GPU 한장을 사용하여 30 에포크(epochs) 학습하였고 하이퍼 파라미터 (hyper parameter)는 미니 배치 사이즈 16, 학습률 0.0003, 학습률 디케이 (learning rate decay) 0.94 로 학습하였다. 트랜스포머에서 디코더의 고정된 길이는 100 으로 지정했고 넘으면 사용하지 않았다. 학습셋이 적기 때문에 정규화 (Regularization)를 고려해서 웨이트 디케이 0.0001 로 학습하였다. 입력 이미지의 크기는 512x512 로 패딩 (padding)을 포함한 리사이즈 (resize) 중앙정렬로 만들었다. 학습 데이터 셋의 30%는 이미지의 높이, 너비 각각 0.85~1 로 무작위 크롭을 적용했다.

5.3. 결 과

유통기한 정확도 검증 셋은 전체 데이터 셋에서 무작위로 추출 했다. 검증 셋은 합성 셋 1000 장, SSG 상품 셋 100 장으로 구성되었다.

유통기한 검출 모델은 검출에서 사용하는 평가 방법인 mAP (Mean Average Precision)로 측정했고 유통기한 인식 모델은 검증 셋의 크롭 이미지 기준으로 두 문자열 편집 거리 (levenshtein distance)를 계산 하였다. 검출 결과는 합성 셋 93%, SSG.COM 상품 셋 85% 이다. 인식 결과는 합성 셋 97%, SSG.COM 상품 셋 83% 이다.

OCR 프로세스를 통해 년, 월, 일을 추출한 테스트 결과 중 일부를 [그림 5]에 나타내었다. 실제 상품 데이터 셋에 많은 비중을 차지하는 유통기한 형식, 예를 들어 {년,월,일,까지} 같은 형식은 잘 인식한다. 반면 {exp,년,월,일,prod,년,월,일} 두 줄로 있는 경우와 같이 비중이 적은 형식은 잘 인식 못하는 경우가 있다. 그림에서 보듯이 유통기한 영역을 정확히 검출해내고, 글자를 인식해낸 케이스가 있는 반면, 그렇지 못한 케이스도 다수 존재하였다. 기본적으로 한 줄로 쓰여진 유통기한 같은 경우가 대부분이어서 오탐, 오타없이 잘 인식이 되었다. 특히, 상단의 이미지와 같이 유통기한 주변에 혼동을 줄 수 있는 많은 글자들이 있는 경우에도 유통기한을 잘 인식해낼 수 있었다. 하지만 두 번째 줄과 같이 음각 유통기한과 우측 이미지 같이 육안으로 인지하기 어려운 정도는 검출 정확도가 상당 부분 떨어지면서 글자의 인식에도 혼동을 주는 것을 확인할 수 있다.

6. 결론 및 Future work

유통기한 검출, 인식 모델은 유통기한의 특유한 폰트가 아니면 검출이 안 되는 경우가 있다. 인식도 년, 월, 일 은 유통기한에서 다른 글자에 비해 많기 때문에 인식률이 좋지만 년, 월, 일 이외 시리얼 넘버 또는 유통기한 정보는 잘 인식이 안 되는 경우가 있다. 학습셋에서 적거나 다루지 못한 특이한 형식의 유통기한 또한 인식이 잘 안 된다. 이런 문제를 해결하기 위해서 이미지의 전체 글자를 검출, 인식하고 후처리로 유통기한 태그를 학습해서 추출하는 방법으로 연구할 것이다. 또한 유통기한 모델은 속도와 성능을 고려해서 종단간 (end to end)을 연구 중이다. 검출의 베이스 네트워크와 인식의 베이스 네트워크가 학습 웨이트 (weights)를 공유함으로 시너지가 날 것이라 생각하고, 검출과 인식이 상호 도움을 주어서 정확도가 높아질 것이라 예상한다. 학습 데이터 관련해서 글자 기반 데이터 셋을 만들 때 시간이 많이 소비 때문에 시나리오 또는 부분 자동으로 이 부분을 개선할 연구도 필요하다.

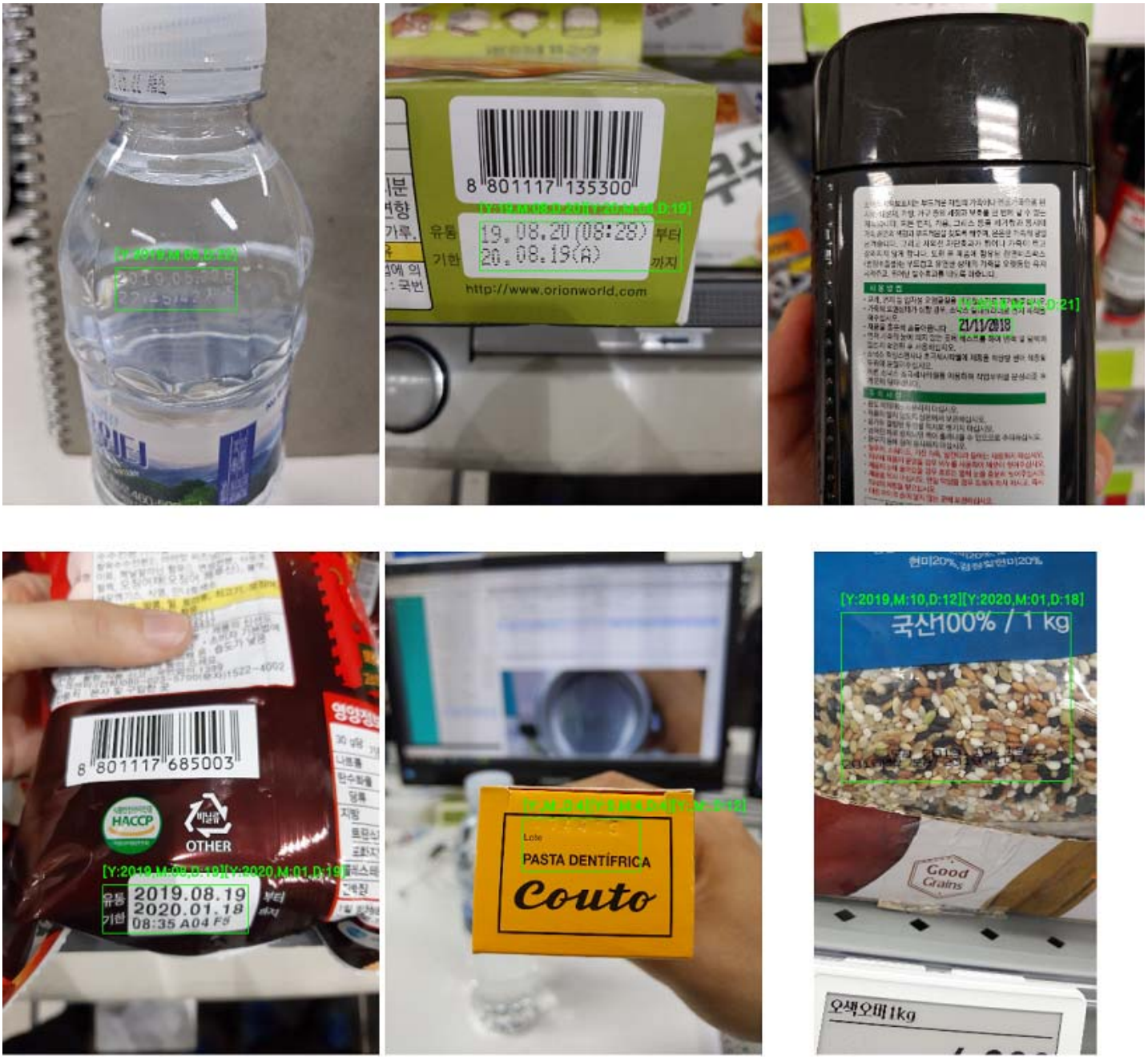

8 801117 135300
19.08.20(08:28)
20.08.19(A)
http://www.orionworld.com
21/11/2018
8 801117 685003
HACCP
OTHER
2019.08.19
2020.01.18
08:35 A04 F8
PASTA DENTÍFRICA
Couto
[Y:2019,M:10,D:12][Y:2020,M:01,D:18]
국산100% / 1 kg
Good Grains
오색오버1kg

그림 5: 검출 및 인식 결과.

참고문헌

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. "Attention Is All You Need" Advances in Neural Information Processing Systems 30 (NIPS), 2017.

[2] Liao, Minghui, Baoguang Shi, and Xiang Bai. "Textboxes++: A single-shot oriented scene text detector." IEEE transactions on image processing 27.8 (2018): 3676-3690.

[3] Liao, Minghui, et al. "Rotation-sensitive regression for oriented scene text detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[4] Deng, Linjie, et al. "Detecting multi-oriented text with corner-based region proposals." Neurocomputing 334 (2019): 134-142.

[5] Zhong, Zhuoyao, Lei Sun, and Qiang Huo. "An anchor-free region proposal network for Faster R-CNN-based text detection approaches." International Journal on Document Analysis and Recognition (IJDAR) 22.3 (2019): 315-327.

[6] Deng, Dan, et al. "Pixellink: Detecting scene text via instance segmentation." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[7] Xu, Yongchao, et al. "Textfield: Learning a deep direction field for irregular scene text detection." IEEE Transactions on Image Processing (2019).

[8] Liu, Zichuan, et al. "Towards robust curve text detection with conditional spatial expansion." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

[9] Wang, Wenhai, et al. "Shape Robust Text Detection With Progressive Scale Expansion Network." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

[10] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, Hwalsuk Lee "Character Region Awareness for Text Detection" The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 9365-9374.

[11] Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan. "Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge" IEEE Transactions on Pattern Analysis and Machine Intelligence, 2016, 652 - 663.

[12] Zbigniew Wojna, Alex Gorban, Dar-Shyang Lee, Kevin Murphy, Qian Yu, Yeqing Li, Julian Ibarz. "Attention-based Extraction of Structured Information from Street View Imagery" International Conference on Document Analysis and Recognition (ICDAR), 2017, 2379-2140.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" arXiv preprint arXiv:1810.04805, 2018.

[14] Mingxing Tan, Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" Thirty-seventh International Conference on Machine Learning (ICML), 2019.

[15] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, Llion Jones. "Character-Level Language Modeling with Deeper Self-Attention" Association for the Advancement of Artificial Intelligence (AAAI), 2019, Volume 33 Number 1.

군집개체 강화학습을 위한 시뮬레이션 환경 기술동향

배정호 ○, 김성호, 김용덕, 성영화

국방과학연구소 국방첨단기술연구원

{jhbae, cocktail, yd.kim, yhsung}@add.re.kr

A Survey on Simulation Environments for Swarm Reinforcement Learning

Jung Ho Bae○, Sung Ho Kim, Yongduk Kim, Young Hwa Sung

Agency for Defense Development

요 약

구글 딥마인드가 아타리 게임을 스스로 플레이하는 인공지능을 발표한 이후로 알파고, 알파스타, ResNet, OpenAI Five 등이 각각 바둑, 스타크래프트2, 이미지 인식, 도타2에서 인간을 뛰어넘으면서 인공지능의 효과성이 검증되었다. 하지만 기존의 방식은 단일 개체를 고려한 학습이거나 컴퓨터내에서 동작하는 게임환경에서의 학습 위주로 수행되었다. 실 세계의 운송, 방산, 스포츠 등 협업과 경쟁이 필요한 분야에서 인공지능을 적용하기 위해서는 여러 개체들 간의 관계를 고려한 강화학습 기법이 필요하다. 이를 위해서 실 장비와 환경에 대한 충실도 높은 시뮬레이션 환경과 이와 연동하는 학습 프레임워크가 필수적으로 선행되어야 한다. 본 논문에서는 무인차량, 드론, 무인수상정 등 군집 무인수송체 강화학습을 위한 시뮬레이션 환경들 조사하고 비교하였다. 조사 대상은 드론과 로봇을 위한 시뮬레이터들과 군집개체 강화학습을 위한 프레임워크들을 포함한 15개의 시뮬레이션 환경들이다.

1. 서 론

최근 첨단 와해적 신기술로서 저비용 단주기 군집체계에 대한 연구가 활발히 진행되고 있다. 평창 동계올림픽에서 인텔(Intel)은 자체 개발한 슈팅스타(Shooting Star) 드론 1,218대를 동시 운용하여 다양한 퍼포먼스를 선보였다[1]. 뿐만 아니라 사우디 아라비아에서는 570만 배럴의 오일 탱크가 테러리스트의 군집드론 의해 공격받았고[2], 미국방성의 SCO(Strategic Capabilities Office)는 자체 개발한 고정익 드론인 Perdix를 103대 동시 운용하면서 무인기들간 집단 의사결정, 적응형 군집비행등을 시연하였다[3].

기존 대부분의 군집개체 운용은 중앙집중식으로 수행되며 이 방식은 통신, 운용, 제어 등의 중요 임무가 중앙 통제소에 집중되어 있다. 이에따라 군집개체 운용의 확장성, 자율성, 강건성이 제한되며 중앙 통제소 불능 시 군집 전체가 마비되는 단일취약성이 존재한다. 이 문제를 극복하기 위하여 중앙 통제를 최소화하여 자율 임무수행이 가능한 비중앙집중식 군집 운용에 대한 연구가 필요하다.

또한 임무의 종류가 다양해지고 복잡해질수록 기존 방식의 한계는 더욱 명확해진다. 기존의 전통적인 규칙기반 방법은 가능한 모든 조건을 개발자가 입력해야하며 고려되지 않은 조건에 대해서는 대처가 어려운 문제가 있다. 또한 개발자가 정의한 규칙에 대한 대응이 효율/효과적인 것인가에 대한 근본적인 문제도 존재한다. 복잡하고 다양한 임무에 대해서 효과적이고 효율적으로 대응하기 위하여 다양한 강화학습기반 지능형 군집체계에 관한 연구가 진행되고 있다. 구글의 딥마인드(Deepmind)가 개발한 알파스타(AlphaStar)는 다양한 유닛을 제어해야하는 전략 시뮬레이션 게임인 스타크래프트2에서 승률 99.8%를 달성하고 최고 레벨인 그랜드마스터를 달성하였으며[4], 오픈AI(OpenAI)가 개발한 강화학습 알고리즘인 OpenAI Five는 5:5 전략배틀 게임인 도타2(DoTa2)에서 세계 챔피언 팀에게 승리하였다[5].

효과적인 강화학습 기법을 게임이 아닌 무인차량, 드론, 무인수상정 등 실제 무인수송장비에 적용하기 위해서는 충실도 높은 시뮬레이션과 강화학습 프레임워크의 지원이 필수이다. 충실도 높은 시뮬레이션을 위해서는 객체와 환경을 실제의 특성을 잘 반영할 수 있도록 높은 해상도로 모델링 할 수 있어야 하며, 신뢰할 수 있는 3D 물리엔진을 이용하여 시뮬레이션을 실행하고, 영상처리 등을 위한 실제와 유사한 가시화 기술을 제공해야 한다. 또한 학습을 원활하게 수행하기 위하여 학습 프레임워크와의 연동을 지원해야 하며, 소프트웨어적으로 개발한 학습 모델들을 실 장비에 적용하기 위한 실 장비에 사용되는 미들웨어와의 연동 지원 등을 고려해야한다. 효과적인 학습을 위해서는 효율적으로 시나리오를 저장/실행/학습할 수 있는 기능이 있어야 하며, 학습시간 단축을 위한 분산/병렬 처리가 가능하고, 분석 및 결과 확인 기능을 제공해야 한다.

본 논문에서는 군집 무인수송장치의 강화학습을 위한 시뮬레이션 환경들을 조사하고 비교한다. 조사한 시뮬레이션 환경은 드론 및 로봇에 대한 시뮬레이터들과 이 시뮬레이터와 연동 가능한 강화학습 프레임워크들이다.

본 논문의 구성은 다음과 같다. 2장에서는 연구배경으로 시뮬레이션과 강화 학습을 설명하고, 3장에서 조사한 시뮬레이션 실행 환경을 비교 설명한다. 4장에서는 결론을 논의한다.

2. 연구배경

군집 드론, 로봇 등의 실장비의 강화학습을 고려한 시뮬레이션을 위해서는 다음 사항들을 고려해야 한다. 실제 장비를 독립적으로 정확하게 모델링하고, 실장비와 연동한 HILS(Hardwar-In-the-Loop Simulation) 또는 PILS(Processor-In-the-Loop Simulation)를 수행하기 위하여 많이 사용되는 미들웨어나 통신 프로토콜을 지원해야한다. 로봇이나 드론 등을 위한 미들웨어로는 ROS(Robot Operating System)[6], MavLink(Micro Air Vehicle Link)[7] 등이 있다. 실제 환경을 높은 충실도로 표현하기 위해서는 3D 물리엔진을 적용해야 한다. 3D 환경에서 사용되는 대표적인 물리엔진으로는 언리얼 4[8], Unity[9], MuJoCo[10], Blender[11] 등이 있다.

강화학습을 위한 프레임워크로는 구글에서 개발한 텐서플로우(TensorFlow)[12], 페이스북에서 개발한 파이토치(PyTorch)[13], 텐서플로우 기반의 케라스(Keras)[14], 매트랩에서 개발한 Caffe[15], 마이크로소프트에서 개발한 CNTK[16] 등이 있다.

심층 강화학습 알고리즘으로는 Q 러닝 기반의 DQN(Deep Q Network)[17]과 DQN 파생 알고리즘[18], 정책 경사 기반의 PPO(Proximal Policy Optimization)[19]와 병렬처리를 위한 A2C[20] 알고리즘 등이 사용된다. Q 러닝 기반의 강화 학습은 현재 환경에서 수행 가능한 액션에 대한 최대 기대 값을 학습을 통하여 갱신한 후, 학습된 최대 기대 값에 따라 액션들을 선택하는 방식이다. 정책 경사 기반 강화학습은 최대 기대 값을 가지는 액션들을 결정적인 아닌, 확률적으로 선택하도록 하고 해당 확률을 학습하는 방식이다. 이는 가위바위보 등의 게임은 결정적인 선택보다는 확률적인 선택이 최적이라는 개념을 적용한 것이다. 다양한 정책 경사 기반의 알고리즘들이 발표되었지만 현재는 PPO가 거의 표준적으로 사용되고 있다.

군집개체를 위한 강화학습 (Multi-agent Reinforcement Learning, MARL) 알고리즘은 기존의 단일 개체를 위한 강화학습 알고리즘에서 협업 또는 경쟁하는 다른 개체들에 대한 고려가 추가된다. 대표적인 MARL 알고리즘으로는 Q 러닝 기반의 QMix[21], Q러닝과 정책 경사를 결합한 액터-크리틱(actor-critic) 기반의 MADDPG(Multi-Agent Deep Deterministic Policy Gradient)[22] 등이 있다. QMix 알고리즘은 기존의 완전 분산형 DQN 알고리즘과 모든 에이전트의 상태 및 전역 상태를 중앙에서 통제하는 완전 중앙집중식의 절충안이다. 이 알고리즘은 개별 에이전트가 관찰한 지역 값들로 신경망을 1차 학습하고, 모든 개별 신경망의 결과값을 종합하여 공동 행동 값을 2차로 학습하는 방식을 사용한다. 결과적으로 완전 분산형보다 안정적으로 학습이 가능하고 완전 중앙집중식보다 확장성이 높다는 장점을 가진다. MADDPG 알고리즘은 다른 에이전트와의 협업/경쟁을 고려한 액터-크리틱 기반 알고리즘으로, 학습 시에는 다른 에이전트의 실시간 변화하는 정책을 고려한 중앙집중식으로 진행하고, 실행 시에는 자신의 측정값만을 가지고 다른 에이전트의 정책을 추론하는 형태로 진행된다. Q-함수를 사용하는 중앙집중식 학습 시에 다른 에이전트의 액션들을 입력으로 받기 때문에, 다른 에이전트의 정책 변화에도 안정적으로 학습이 가능하다는 장점이 있다.

3. 시뮬레이션 실행 환경

이 장에서는 본 논문에서 조사한 강화학습 시뮬레이션 환경들을 설명한다. 표 1은 각 시뮬레이션 환경들의 구현 언어(Language), 지원 OS(OS), 라이선스(License), 시뮬레이션 차원(Dim.), 지원하는 물리엔진(Phy.Eng.), 기본으로 제공하는 심층 강화학습 알고리즘(DeepRL) 등을 비교하여 보여준다. 또한 그림 1은 각 시뮬레이터의 실행 화면을 보여준다.

3.1 AirSim

Aerial Informatics and Robotics Platform (AirSim)[23]은 차량과 드론의 딥러닝, 컴퓨터 비전, 강화학습등의 자율주행 알고리즘을 개발하고 시험하기 위하여 2017년 마이크로소프트에서 개발하였다. 물리엔진은 언리얼 4 (Unreal Engine 4, UE4) 엔진을 기반으로 하고 있으며, MavLink 등의 널리 사용되는 프로토콜들을 지원함으로써 PX4 등의 UAV 컨트롤러와 연동하여 실시간 HILS (Hardware-in-the-Loop Simulation)가 가능하다. 또한 모듈화 구조를 채택하여 확장성을 강조하였다.

3.2 Unity ML-Agent

Unity ML-Agent[24]는 유니티 사에서 제공하는 머신러닝을 위한 툴킷이다. 기존 3D 모델링 도구인 유니티 상에서 모델링한 객체들을 훈련하기 위한 PPO등의 심층 강화학습 알고리즘을 제공한다. 강화학습을 수행하기 위하여 초기화, 환경변수 입력, 행위 수신, 리워드 입력 등을 위한 파이썬 API를 제공한다. 물리엔진으로 유니티 엔진을 사용한다.

아카데미 객체를 통하여 병렬학습을 수행하여 학습속도를 향상 시킬 수 있다.

3.3 OpenAI Gym

OpenAI Gym[25]은 통합 환경 인터페이스를 제공하고 새로운 환경을 재구축하기 위한 플랫폼을 제공한다. 대표적인 시스템으로는 아타리사의 고전게임인 벽돌깨기가 있으며 이 외에도 그리드 월드, 보드게임, 2D 및 3D 로봇과 같은 환경을 제공한다. 물리엔진은 MuJoCo, Robotics, Box2D, Doom 등의 다양한 물리엔진을 기반으로 하는 환경을 제공한다. 강화학습을 위하여 공통으로 사용하는 캡슐화된 함수들을 제공함으로써 사용성을 높였다. OpenAI Baseline은 강화학습 시뮬레이션 환경인 OpenAI Gym의

표 1 군집개체 강화학습을 위한 시뮬레이터 비교

No.	Name	Language	OS	License	Dim.	Phy.Eng.	DeepRL
1	AirSIM	C++, Python	Windows, Linux	MIT	3D	UE4	DQN
2	Unity ML-Agent	C#, Python	Windows, Linux	Apache V2.0	3D	Unity	PPO, SAC, BC, etc.
3	OpenAI Gym	Python	Linux, MacOS	MIT	2D, 3D	MuJoCo, Robotics, etc.	PPO, DQN, etc.
4	Gazebo	C++, Python	Linux, MacOS	Apache V2.0	3D	ODE, Bullet, etc.	PPO
5	MORSE	Python	Linux	BSD	3D	Blender	
6	jMAVSim	Java	Linux, MacOS, Windows	BSD 3	3D		
7	NPS	C	Linux, MacOS	GPLv2	2D	JSBSim	
8	HackflightSim	C++	Windows, Linux	GPL	3D	UE4	
9	Webots	C/C++, Java, Python, Matlab	Linux, MacOS, Windows	Apache V2.0	3D	ODE	
10	CoppeliaSim	C/C++, Java, Python, Matlab, Lua	Linux, MacOS, Windows	GPL, LGPL, MIT	3D	ODE, Bullet, etc.	
11	MAgent	Python	Linux, MaxOS	MIT	2D	N.A.	parameter-sharing DQN, DRQN, A2C
12	SLMLab	Python	Linux	MIT	2D, 3D	MuJoCo, Unity, etc.	DDQN, A2C, etc.
13	Deep RL for Swarm Systems	Python	Linux		2D	N.A.	TRPO
14	Multi-Agent-Reinforcement-Learning-Environments	Python	Linux, MacOS, Windows		2D	N.A.	
15	TensorSwarm	Python	Linux	MIT	3D	Argos3	PPO

확장 패키지 개념으로, 다양한 강화학습 알고리즘을 제공하는 툴킷이다. 현재 DQN, DDPG, TRPO, PPO 등 10개 이상의 최신 강화학습 알고리즘이 구현되어 있으며, MuJoCo와 Atari 벤치마크를 제공하고 있다.

3.4 Gazebo

Gazebo[26]는 2002년 서던 캘리포니아 대학교(Southern California)에서 초기 개발하고 현재 OSRF(Open Source Robotics Foundation)에서 개발하는 ROS (Robot Operating System)에 포함된 시뮬레이터이다. Gazebo는 Open Dynamics Engine(ODE) 등 다양한 물리 엔진과 센서 모델을 사용할 수 있으며, 손쉬운 3D 환경 생성과 로봇 설계 및 알고리즘 테스트, 회귀 테스트, 인공지능 훈련 등을 지원한다. 물리 시뮬레이션, 렌더링, 사용자 인터페이스, 통신, 센서 생성 등을 위한 독립된 라이브러리를 지원하는 분산 아키텍처를 적용하고 있다. ROS를 사용하는 쿼드로터 프레임워크 중 하나인 Hector Quadrotor는 비행 동역학, IMU와 같은 탑재 센서, 외부 이미지 센서, 복잡한 환경 등의 다양한 측면을 모사 할 수 있는 Gazebo ROS 패키지를 제공한다. Iris와 Solo와 같은 Quad부터 Typhoon H480과 같은 Hex, Tailslitter Plane 등 다양한 UAV를 지원한다. 하지만 Hector 패키지는 MavLink 프로토콜이나 Pixhawk 플랫폼 등에 대한 지원은 부족한다.

강화학습을 위하여 ROS 기반으로 OpenAI를 실행시키기 위한 패키지인 openai_ros package를 제공하며, 최소한의 구현으로 강화학습을 수행할 수 있도록 설계되었다. 현재는 ROS 2와 연동하고, 설치, 설정, 학습 편의성 등의 향상을 위하여 OpenAI Gym에 독립적인 gym-gazebo2 패키지를 공개하였다.

3.5 MORSE

Modular Open Robots Simulation Engine (MORSE)[27]는 LAAS(Laboratory for Analysis and Architecture of Systems)와 ONERA(Office National d'Etudes et de Recherches Arospatiales)가 로봇연구를 위하여 공동 개발한 범용 시뮬레이터이다. 3D 시뮬레이션 엔진으로는 Blender를 사용하고 있으며, 다양한 환경에서 동작하는 로봇을 시뮬레이션 하기 위하여 컴포넌트 기반의 모듈 방식으로 설계되었다. 실제 HW와 연동을 위한 다양한 ROS, MavLink 등 다양한 미들웨어를 지원한다. 각 MORSE 컴포넌트는 Blender와 파이썬 파일로 구성되며, Blender 파일은 특정 객체의 물리적 시각적 특성들을 표현하고 파이선 파일은 해당 객체 클래스를 정의한다. MORSE 컴포넌트 라이브러리는 단순한 쿼드로터 강체 모델 예제를 제공한다.

3.6 jMAVSim

Java Micro Air Vehicle Simulator (jMAVSim)[28]는 단순하고 가벼운 멀티로터 시뮬레이터로 PIXHAWK 공학팀에서 개발하였다. jMAVSim은 MavLink 프로토콜을 지원하며, Java3D 라이브러리를 사용하여 시각화하고, 시리얼 통신을 통해 HILS를 수행하거나 UDP를 통해 SILS(Software-In-the-Loop Simulation)를 수행 할 수 있다. 상대적으로 간단한 객체지향 설계를 적용하여 단일 컴포넌트로 제작되었다.

3.7 NPS

New Paparazzi Simulator (NPS)[29]는 ENAV (Ecole Nationale de l'Aviation Civil) UAV Lab. 에서 개발한 UAV 시뮬레이터로 기본 탑재된 비행모델은 JSBSim이며, 이 모델은 다소 복잡한 기체들과 센서모델을 지원 가능하다. 다른 시뮬레이터와의 차이점은 사용자가 비행역학 모델(FDM)을 개발하여 적용할 수 있다는 점이다. FDM 모델은 MATLAB Simulink로 구현하여 시뮬레이터와 연동할 수 있다. FlightGear, Morse 등의 다양한 다른 시뮬레이터로 시각화 할 수 있는 옵션이 존재한다. 회전 날개 항공기와 고정익 기체 UAV들을 지원한다.

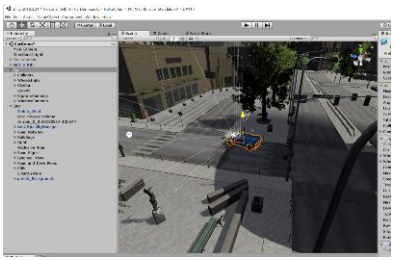
3.8 HackflightSim

HackflightSim[30]는 단순한 크로스 플랫폼 쿼드콥터 시뮬레이터로 미국 Lexington 대학에서 개발하였다. C++로 개발되었으며 언리얼 4 엔진을 사용하였으며 아두이노용 Hackflight 쿼드콥터 비행 제어 펌웨어 기반으로 개발되었다. 해당 프로젝트는 AirSim과 유사하지만, 쿼드콥터만을 지원하는 한계가 있다. HackflightSim은 JMAVSim과 유사하게 단일 컴포넌트로 구현되었다.

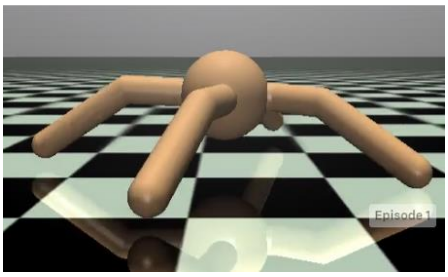
3.9 Webots

Webots[31]은 가장 유명한 모바일 로봇 시뮬레이터 중 하나로, VRML (Virtual Reality Modeling Language) 언어를 사용하여 시뮬레이션 환경을 모델링한다. Webots의 장면 트리는 3D 환경의 필요한 모든 시각적 표현과 시뮬레이션 요소들을 담고 있으며, VRML 노드 특성들과 유사하게 Webots 장면 트리를 구성함으로써 직관적인 모델링이 가능하다. 장면 트리는 Webots가 자동 생성하는 VRML 형태의 '.wbt' 파일로 저장된다. 객체와 환경의 시뮬레이션은 scratch를 이용하여 수행할 수 있다.

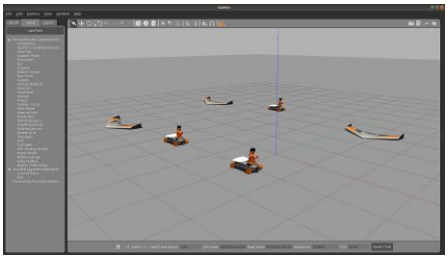
(a) AirSim



(b) Unity ML-Agent

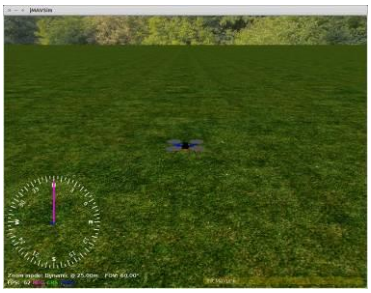


(c) Open AI Gym

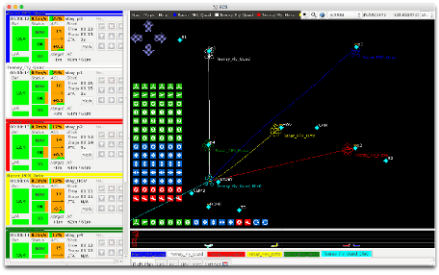


(d) Gazebo

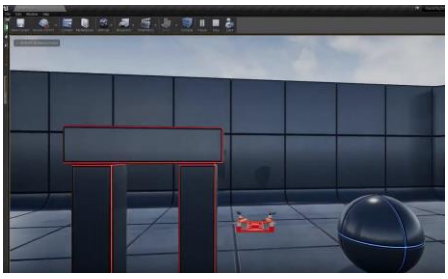
(e) Morse



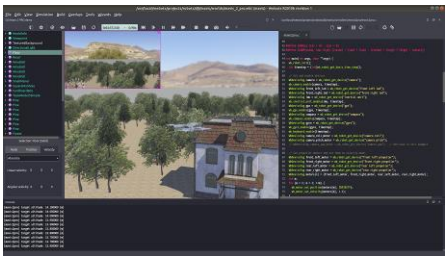
(f) jMAVSim



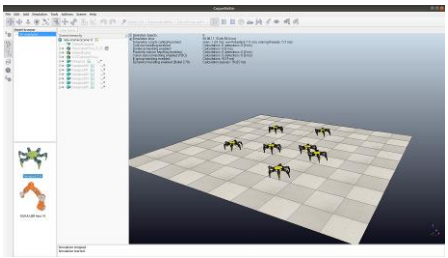
(g) NPS



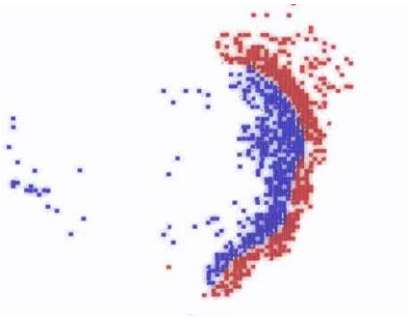
(h) HackflightSim



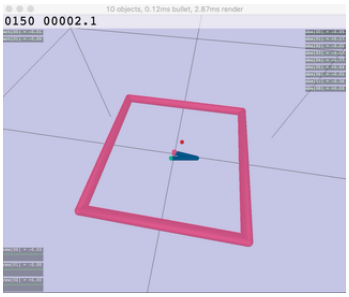
(i) Webots



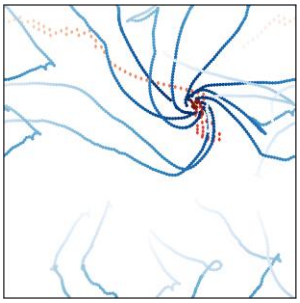
(j) CoppeliaSim



(k) MAgent

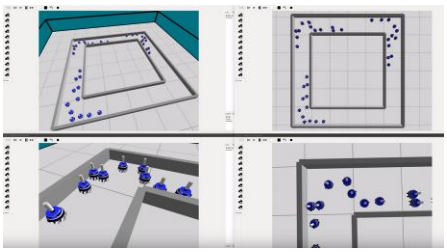


(l) SLMLab



(m) Deep RL for Swarm Systems

(n) Multi-Agent-Reinforcement-Learning-Environments



(o) TensorSwarm

그림 1 군집개체 강화 학습을 위한 시뮬레이터 실행 화면

3.10 CoppeliaSim

CoppeliaSim(구, V-REP)[32]은 멀티 로봇 시뮬레이션을 위한 통합 개발 환경으로 분산 제어 구조 기반으로 되어 있으며, 각 객체는 탑재 스크립트, 플러그인, ROS 노드, BlueZero 노드, 원격 API 등으로 개별적으로 제어 가능하도록 설계되었다. ODE, Bullet, Vortex, Newton의 4개의 물리엔진을 지원하며, 비전 센서, 근접 센서 등을 지원한다. 로봇 학습을 위하여 파이썬으로 구현된 PyRep 패키지를 제공하며 간단한 강화학습 예제를 제공한다. Universal Robots UR10, Franka Emika Panda Gripper, Kuka YouBot 등의 로봇팔, 이동 로봇 등을 지원한다.

3.11 MAgent

MAgent[33]는 UC 버클리에서 개발한 다객체 강화학습 연구 플랫폼이다. 다른 학습 시뮬레이션 플랫폼과의 차이점은 최대 수백만개의 다수 객체를 위한 강화학습을 지원하는 것이다. 객체의 타입, 시야 범위, 액션, 리워드 등을 정의할 수 있고 지형지물을 표현가능하다. 물리 엔진은 포함하고 있지 않으며 2D환경 시뮬레이션이다. 학습 프레임워크로는 텐서플로우와 MXNet을 사용하였고, 제공하는 강화학습 알고리즘으로는 parameter-sharing DQN, DRQN, A2C 등이 있다.

3.12 SLM Lab

SLM Lab[34]은 재현가능한 강화학습 연구를 목적으로 파이토치 기반의 모듈화 구조로 설계된 심층 강화학습 프레임워크이다. 이 프레임워크는 알고리즘, 메모리, 네트워크의 3 종류의 컴포넌트와 파일기반으로 구성하여 하이퍼 파라미터 검색, 결과 분석, 결과 벤치마크 등을 손쉽게 수행 할 수 있도록 하였다. OpenAI Gym, Unity 등의 다양한 2D, 3D 시뮬레이션 환경 등과 연동된다. 공식 지원 심층 강화학습 알고리즘으로는 DQN, Double-DQN, A2C, PPO, SAC 등이 있다.

3.13 Deep RL for Swarm Systems

Deep RL for Swarm Systems[35]는 링컨 대학과 다름슈타트 공과대학교에서 공동개발한 군집객체 강화학습 환경이다. 100 x 100의 2D 격자 배경을 지원하며 다중 객체의 움직임을 시각화하여 보여준다. 물리엔진은 미포함되어 있으며 객체들이 동일하다고 가정하여 모델링 되었다. 기존의 객체간 통신기반이나 영상기반의 강화학습 방법들을 개선한 MFE(mean feature embeddings) 기법을 사용한 강화학습 알고리즘을 직접 개발하여 제공하고 있으며, TRPO(Trust Region Policy Optimization)를 사용하여 다수 객체 조우 문제인 랑데뷰 문제와 다수 객체 포위 문제 등의 예제를 제시하였다.

3.14 Multi-Agent-Reinforcement-Learning-Environment

Multi-Agent-Reinforcement-Learning-Environments [36]는 노스이스턴 대학교에서 개발한 단순한 구조의 군집 객체 강화학습 플랫폼으로 swig, opencv 등의 최소한의 패키지만을 이용하여 다양한 환경에서 실행해 볼 수 있는 장점을 가진다. 2D환경에서 동작하고 물리엔진은 포함하고 있지 않으며, 13개 이상의 다양한 예제 환경을 제공하고 있다.

3.15 TensorSwarm

TensorSwarm[37]은 중국 Dorabot사에서 개발한 군집 로봇을 위한 강화학습 프레임워크이다. 객체들 간의 통신이 현실적으로 불안정하기 때문에 통신을 배재하고 각 객체의 전방 센싱 값과 현재 위치 값만을 이용한 완전 분산형 군집개체 강학습 기법을 제안하였다. 100개 이상의 로봇에 대하여 강화학습을 수행하였으며 7가지 이상의 시나리오에서 의미 있는 학습결과를 보였다. 미들웨어로는 ROS를 사용하고, 물리엔진 및 3D 시뮬레이션 도구로는 Argos 3를, 강화학습 알고리즘은 PPO를 적용하였다.

4. 결론

드론, 무인수상정, 로봇 등의 실 세계에 적용 가능한 군집 강화학습 연구를 위해서는 충실도 높은 시뮬레이션 및 해당 시뮬레이션과 연동한 군집 강화학습 프레임워크의 지원이 필수적이다. 본 논문에서는 이를 위한 15개의 오픈소스 시뮬레이션 도구 및 강화학습 프레임워크를 비교하였다. 우리는 본 조사가 실 장비에 대한 강화 학습을 연구에 기여하기를 기대한다. 향후 본 연구를 기반으로 시뮬레이션 도구와 강화학습 프레임워크를 선정하여 군집 무인수송장치의 학습을 수행할 예정이다.

참고문헌

[1] N. Kshetri and D. Rojas-Torres, “The 2018 Winter Olympics: A Showcase of technological advancement,” *IT Professional*, 2, pp. 19-25, 2018.

[2] B. Huddard, P. Karasz and S. Reed, “Two Major Saudi Oil Installations Hit by Drone Strike, and US Blames Iran,” *The New York Times*, Sept. 14, 2019.

[3] D. Furness, “The Sound of 103 Micro Drones Launched from an F/A-18 Will Give You Nightmares,” *Digital Trends*, Jan. 11, 2017.

[4] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W.M. Czarnecki, and T. Ewalds, “AlphaStar: Mastering the Real-Time Strategy Game StarCraft II,” *DeepMind Blog*, 2019

[5] J. Raiman, S. Zhang and F. Wolski, "Long-Term Planning and Situational Awareness in OpenAI Five," *arXive preprint arXive:1912.06721*, 2019.
[6] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs and A. Y. Ng, "ROS: An Open-Source Robot Operating System," In *ICRA workshop on open source software*, vol.3, no.3.2 p.5, 2009.
[7] J. Li, Y. Zhou and L. Lamont, "Communication Architectures and Protocols for Networking Unmanned Aerial Vehicles," In *2013 IEEE Globecom Workshops*, pp. 1415-1420, 2013.
[8] B. Karis, E. Games, "Real shading in Unreal Engine 4," *Proc. Physically Based Shading Theory Practice*, vol. 4, 2013
[9] U. Techologies, Unity, https://unity3d.com/.
[10] E. Todorov, T. Erez and Y. Tassa, "Mujoco: A Physics Engine for Model-based Control," In *2012 IEEE/RSJ International Conferenece on Intelligent Robots and Systems*, PP. 5026-5033, Oct, 2012.
[11] B.R. Kent, *3d Scientific Visualization with Blender*, Morgan & Claypool Publishers, 2014.
[12] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean and M. Kudlur, "Tensorflow: A System for Large-Scale Machine Learning," In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265-286, 2016.
[13] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito and A. Lerer, "Automatic Differentiation in Pytorch," *Journal of Machine Learning Research*, vol. 18, pp. 1-43, 2018.
[14] A. Gulli and S. Pal, *Deep Learning with Keras*, Packt Publishing Ltd., 2017.
[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girschick and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," In *Proceedings of the 22nd ACM International conference on Multimedia*, pp. 675-678, Nov. 2014.
[16] F. Seide and A. Agarwal, "CNTK: Microsoft's Open-Source Deep-Learning Toolkit," In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2135-2135, Aug, 2016.
[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXive preprint arXiv:1312.5602*, 2013.
[18] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney and D. Silver, "Rainbow: Combining Improvements in Deep Reinforcement Learning," In *Thirty-Scond AAAI Conference on Artificial Intelligence*, pp. 3215-3222, Feb. 2018.
[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[20] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley and K. Kavukcuoglu "Asynchronous Methods for Depp Reinforcement Learning," In *International Conference on Machine Learning*, pp. 1928-1937, Jun, 2017
[21] T. Rashid, M. Samvelyan, C.S. De Witt, G. Farquhar, J. Foerster and S. Whiteson, "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning," *arXive preprint arXive:1803.11485*, 2018.
[22] R. Lowe, Y. Wu, A. Tamar, J. Harb, O.P. Abbeel and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," In *Advances in Neural Information Processing Systems*, pp. 6379-6390, 2017.
[23] Microsoft, AirSim web page, https://github.com/Microsoft/AirSim.
[24] A.Juliani, V.Berges, E.Vckay, Y.Gao, Y.Henry, M.Mattar and D.Lange, Unity: A General Platform for Intelligent Agents. *arXiv preprint arXiv:1809.02627*, 2018
[25] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba, OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016
[26] V.M. Vilches, A.H. Cordero, A.B. Calvo, I.Z. Ugarte and R. Kojcev, Robot_gym: Accelerated Robot Training through Simulation in the Cloud with ROS and Gazebo. *arXiv preprint arXiv:1808.10369*, 2018
[27] O. LAAS and labs, Morse web page, https://github.com/morse-simulator/morse.
[28] P. engineering team, jMAVSim web page, https://github.com/PX4/jMAVSim.
[29] E.N. de l'Aviation Civil, Paparazzi web page, https://github.com/paparazzi/paparazzi/tree/master/sw/simulator.
[30] S.D. Levy, HackflightSim web page, https://github.com/simondlevy/HackflightSim.
[31] O. Michel, "Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation," *International Journal of Advanced Robotics*

Systems, vol. 1, no. 1, pp. 39-42, 2004.

[32] E. Rohmer, S.P. Singh and M. Freese, "CoppeliaSim (formerly V-REP): A Versatile and Scalable Robot Simulation Framework," Coppelia Robotics, http://coppeliarobotics.com.

[33] L. Zheng, J. Yang, H. Cai, M. Zhou, W. Zhang, J. Wang and Y. Yu, "MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence," In *Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 1-2, 2018

[34] W.L. Keng and L. Graesser, SLM Lab, https://github.com/kengz/SLM-Lab.

[35] M. Hüttenrauch, S. Adrian and G. Neumann, "Deep Reinforcement Learning for Swarm Systems," *Journal of Machine Learning Research*, val. 20, no. 54, pp. 1-31, 2019.

[36] J. Shuo, Multi Agent Reinforcement Learning Environments Compilation, https://github.com/Bigpig4396/Multi-Agent-Reinforcement-Learning-Environment.

[37] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang and J. Pan, "Towards Optimally Decentralized Multi-Robot Collision Avoidance via Depp Reinforcement Learning," In *2018 IEEE International Conference on Robotics and Automation*, pp. 6252-6259, 2018.

ADD
AGENCY FOR DEFENSE DEVELOPMENT

국방 SPL 프레임워크 기반의 무기체계 항법 SW 플랫폼 요구사항 분석

2020. 2.

노성규, 박병수, 남성호, 성창기, 백승준, 박삼준
국방과학연구소

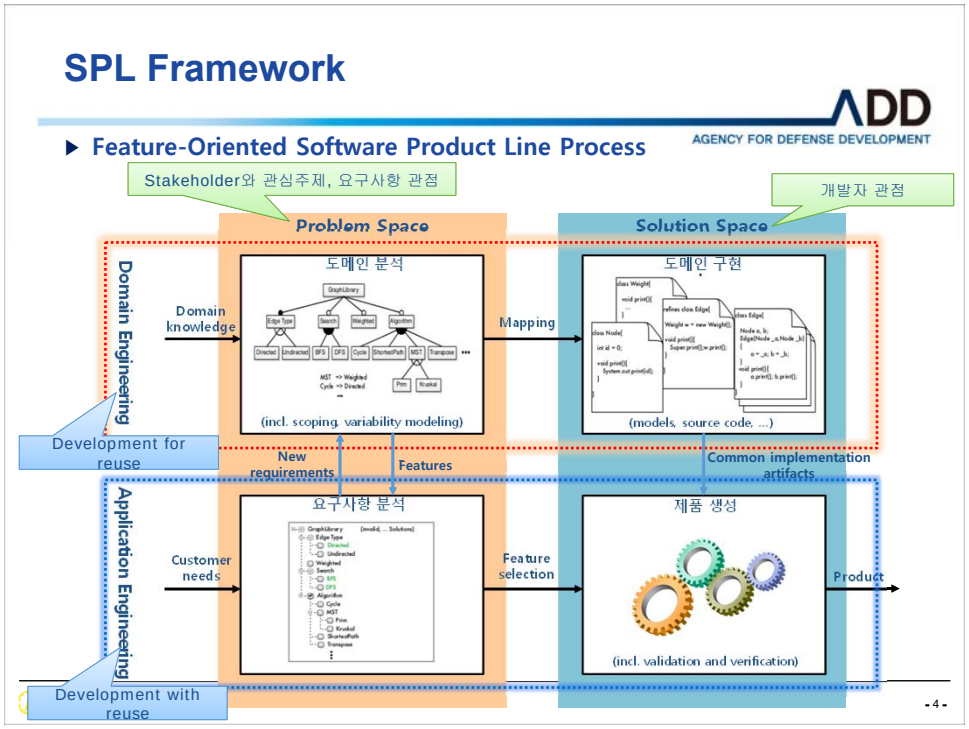
국방과학연구소
Agency for Defense Development

Agenda

- 개요
- 플랫폼 요구사항 개발
- 결론 및 향후 계획

국방과학연구소
Agency for Defense Development


Chapter
개요
국방과학연구소
Agency for Defense Development


SPL Framework
ADD
AGENCY FOR DEFENSE DEVELOPMENT
Feature-Oriented Software Product Line Process
Stakeholder와 관심주제, 요구사항 관점
개발자 관점
Problem Space
Solution Space
Domain Engineering
Domain knowledge
도메인 분석
(incl. scoping, variability modeling)
Mapping
도메인 구현
(models, source code, ...)
Development for reuse
New requirements
Features
Common implementation artifacts
Application Engineering
Customer needs
요구사항 분석
Feature selection
제품 생성
(incl. validation and verification)
Product
Development with reuse
-4-

국방 SPL 프레임워크 배경 – 무기체계 SW 특징 (1/2)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 무인기 체계의 SW 플랫폼 필요성 : 형태, 고도, 성능별 요구 기능은 다르지만 공통으로 활용 가능한 SW 존재



형태별	고정익형 / 회전익형 / 수직 이착륙형
크기별	초소형(0.3m이하) / 소형(0.3-10m) / 중형(10-20m) / 대형(20m이상)
체공별	단기체공(1시간이하) / 중기체공(1-12시간) / 장기체공(12시간이상)
고도별	저고도(3km이하) / 중고도(3-10km) / 고고도(10km이상)
성능별	근거리(50km이하) / 단거리(50-200km) / 중거리(200-500km) / 장거리(500km이상)
기능별	관측 / 탐사, 정찰 / 감시, 표적, 기타

국방과학연구소 Agency for Defense Development

-5-

국방 SPL 프레임워크 배경 – 무기체계 SW 특징 (2/2)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

항법 소프트웨어는 항체의 위치를 찾아내는 기능을 수행하는 소프트웨어로
주요 무기체계에 필수적으로 탑재되는 핵심 소프트웨어임

현상황

- ✓ 탑재되는 무기체계의 특성에 따라 개별적으로 개발
- ✓ 공통적인 요소와 개별적으로 개발되어야 할 요소를 식별하여 전략적인 재사용 필요
- ✓ 다양한 요구사항에 대비 및 품질 향상이 필요

다양한 무기체계에 필수적인 공통기능

무기체계 사업별 독자적인 개발

개발 용이성 및 성능 향상 필요

품질, 상호운용성, 유지보수성 제고필요

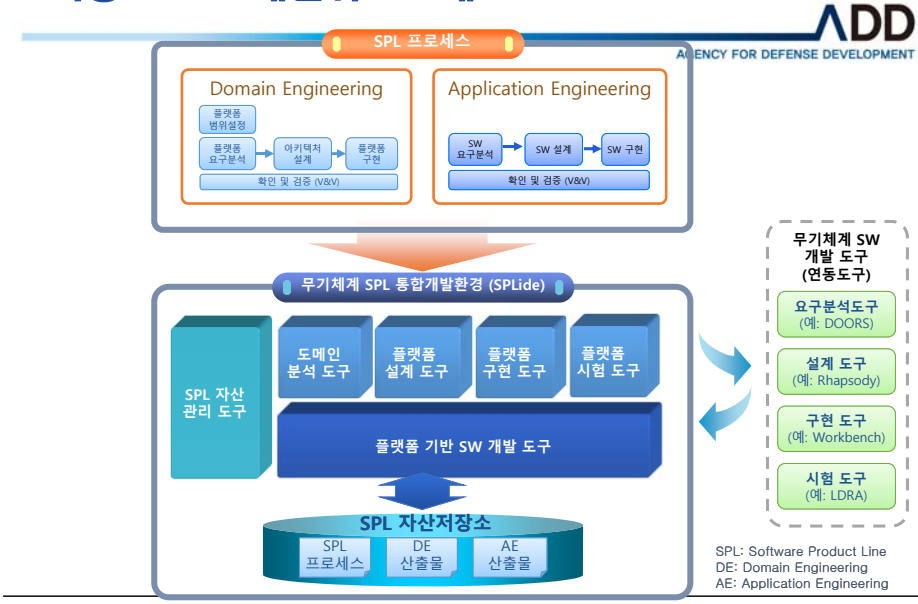
검증된 소프트웨어 아키텍처 및 공통 컴포넌트의 확보로 무기체계 소프트웨어의 신뢰성 향상

국방과학연구소 Agency for Defense Development

-6-

국방 SPL 프레임워크 – 개요

ADD
AGENCY FOR DEFENSE DEVELOPMENT



SPL: Software Product Line
DE: Domain Engineering
AE: Application Engineering

국방 SPL 프레임워크 – 프로세스

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 표준에 근거한 무기체계 SPL 프로세스 정의
- ✓ "ISO/IEC 26550:2015"의 SPL 엔지니어링 프로세스
- ✓ 무기체계 소프트웨어 개발 및 관리 매뉴얼의 획득 프로세스

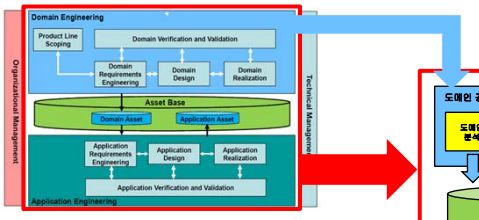


그림. ISO/IED 26550:2015의 SPLE 프로세스

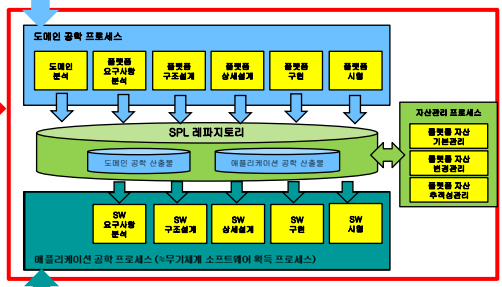
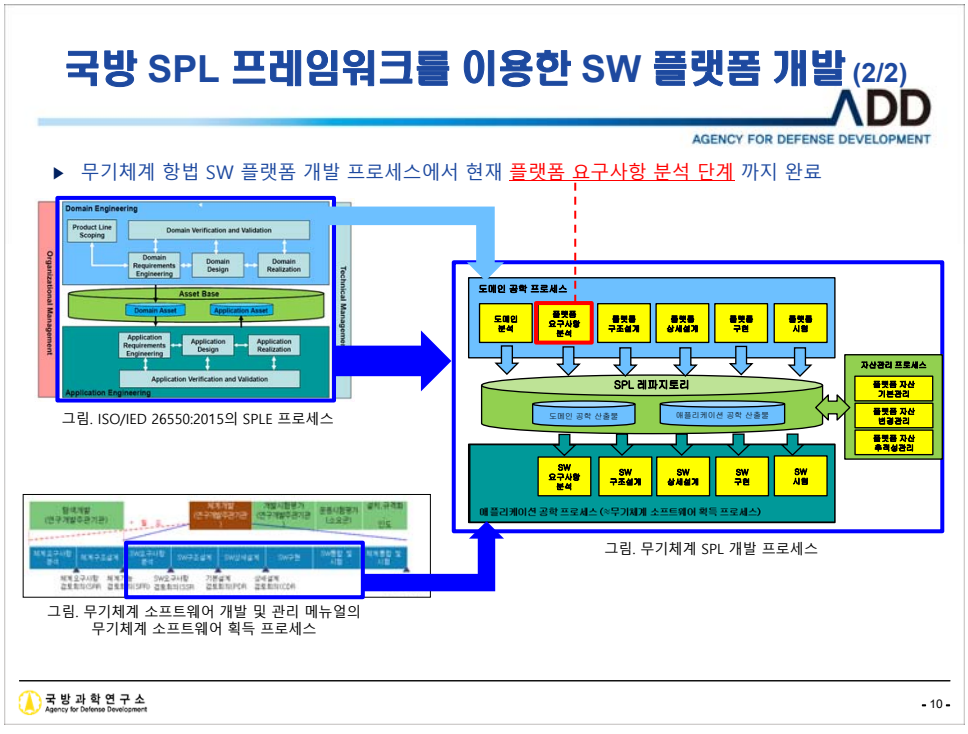


그림. 무기체계 SPL 개발 프로세스

그림. 무기체계 소프트웨어 개발 및 관리 메뉴얼의 무기체계 소프트웨어 획득 프로세스


국방 SPL 프레임워크를 이용한 SW 플랫폼 개발 (1/2)
ADD
AGENCY FOR DEFENSE DEVELOPMENT
무기체계 적용 관성항법시스템
유도
전술지대지 항법SW
천궁 지상항법SW
전술함대지 항법SW
130mm 항법SW
항공
MUAV 항법SW
차기군단무인기 항법SW
공통성·가변성 분석
무기체계 공통 관성항법 SW 플랫폼
피처 모델
항법 SW
항법방식
기준좌표계
센서
상대항법
복합항법
절대항법
기계식 자이로
광학식 자이로
진동형 자이로
플랫폼 요구사항
플랫폼 컴포넌트
위치계산
속도계산
자세계산
동적오차 보상
자체정렬
운항중 정렬
전달정렬
상태감시
플랫폼 설계
체계별 요구사항 분석
체계별 요구사항 분석
AA체계 관성항법시스템 SW
위치계산
자체정렬
보정정보 전처리
고장검출 /분리
속도계산
자세계산
연동SW
연동SW
OS
BB체계 관성항법시스템 SW
위치계산
자체정렬
보정정보 전처리
고장검출 /분리
속도계산
자세계산
연동SW
연동SW
OS
주기
명칭
플랫폼 공통 컴포넌트
명칭
적용 무기체계 특화 컴포넌트
국방과학연구소
Agency for Defense Development
- 9 -


국방 SPL 프레임워크를 이용한 SW 플랫폼 개발 (2/2)
ADD
AGENCY FOR DEFENSE DEVELOPMENT
▶ 무기체계 항법 SW 플랫폼 개발 프로세스에서 현재 플랫폼 요구사항 분석 단계 까지 완료
Domain Engineering
Product Line Scoping
Domain Verification and Validation
Domain Requirements Engineering
Domain Design
Domain Realization
Asset Base
Domain Asset
Application Asset
Application Requirements Engineering
Application Design
Application Realization
Application Verification and Validation
Application Engineering
Organizational Management
Technical Management
그림. ISO/IED 26550:2015의 SPLE 프로세스
도메인 공학 프로세스
도메인 분석
플랫폼 요구사항 분석
플랫폼 구조설계
플랫폼 상세설계
플랫폼 구현
플랫폼 시험
SPL 레파지토리
도메인 공학 산출물
애플리케이션 공학 산출물
SW 요구사항 분석
SW 구조설계
SW 상세설계
SW 구현
SW 시험
애플리케이션 공학 프로세스 (=무기체계 소프트웨어 획득 프로세스)
자산관리 프로세스
플랫폼 자산 기본관리
플랫폼 자산 변경관리
플랫폼 자산 추적성관리
그림. 무기체계 SPL 개발 프로세스
그림. 무기체계 소프트웨어 개발 및 관리 메뉴얼의 무기체계 소프트웨어 획득 프로세스
국방과학연구소
Agency for Defense Development
- 10 -

항법 SW 플랫폼 요구사항 분석 (1/6)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

■ 항법 SW 플랫폼 Context(요약) 정의

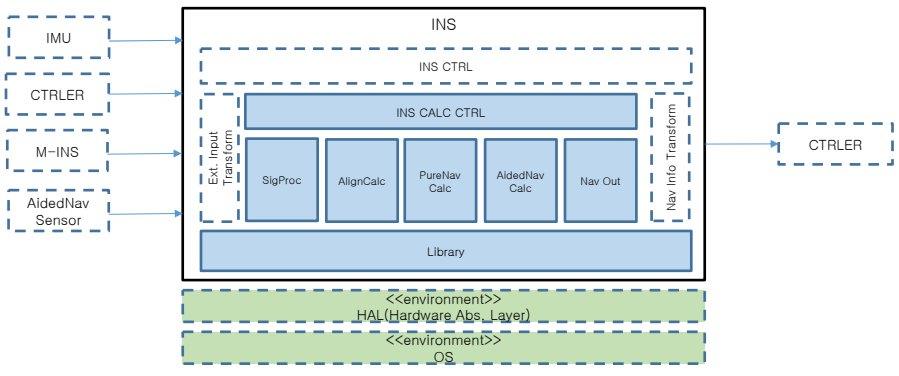


그림. 항법 SW 플랫폼 Context (요약)

항법 SW 플랫폼 요구사항 분석 (2/6)

ADD AGENCY FOR DEFENSE DEVELOPMENT

■ 개발자 관점 개념도 기준으로 요구사항 정의

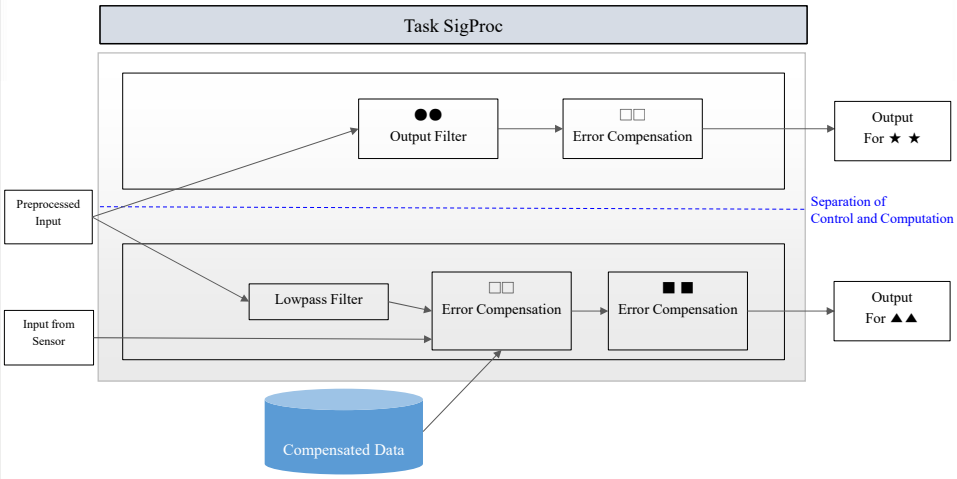
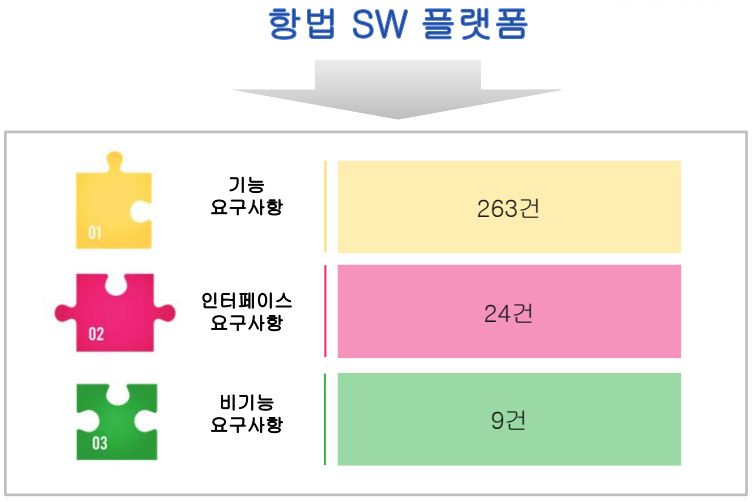


그림. 개발자 관점 개념도 예시

항법 SW 플랫폼 요구사항 분석 (3/6)

ADD AGENCY FOR DEFENSE DEVELOPMENT

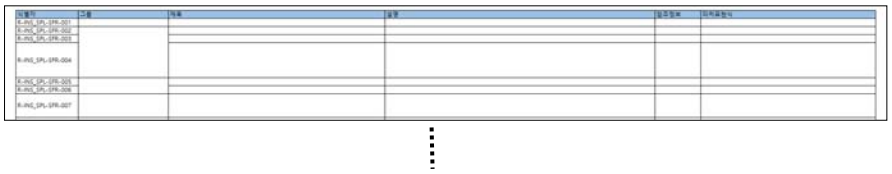


항법 SW 플랫폼 요구사항 분석 (4/6)

ADD AGENCY FOR DEFENSE DEVELOPMENT

- 관성항법 소프트웨어 피처별 요구사항
  - 식별자 : 요구사항 식별자
  - 그룹 : 요구사항 그룹 (대분류)
  - 제목 : 요구사항 제목 (소분류)
  - 설명 : 요구사항에 대한 설명
  - 피처표현식 : SPL통합개발환경(SPLide)에서 통용되는 가변성 표현 기법
    - VNOT 표기법
    - 피처의 생존 조건을 작성
  - 참조정보 : 요구사항 정의서에 표현되지 않는 내용에 대해 참조할 수 있는 (문서) 정보 (예. Use Case 식별자, Interface Control Document 등)

<요구사항 정의서 양식>



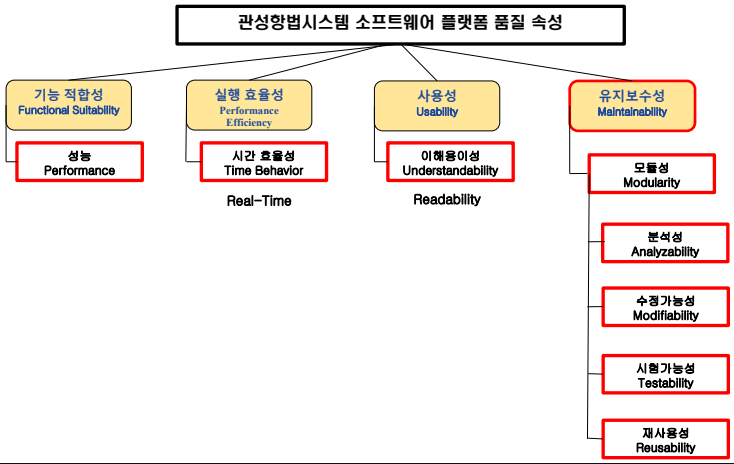
국방과학연구소 Agency for Defense Development

- 15 -

항법 SW 플랫폼 요구사항 분석 (5/6)

ADD AGENCY FOR DEFENSE DEVELOPMENT

- 항법소프트웨어 요구사항 정의
  - 관성항법 소프트웨어 비기능 요구사항(1/2)



국방과학연구소 Agency for Defense Development

- 16 -

항법 SW 플랫폼 요구사항 분석 (6/6)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

항법소프트웨어 요구사항 정의

관성항법 소프트웨어 비기능 요구사항(2/2)

* 중요도(Importance) 및 난이도(Difficulty)는 Low, Medium, High로 구분함.

품질 속성 (대분류)	품질 속성 (소분류)	요구사항 명세	중요도*	난이도*
기능 적합성 (Functional Suitability)	성능 (Performance)	플랫폼은 Task별 실행 주기 이내에 알고리즘 연산이 완료되어야 한다.	High	Medium
실행 효율성 (Performance Efficiency)	시간 효율성 (Time Behavior)	플랫폼의 각 Task들은 실시간성이 보장되어야 한다.	High	Medium
사용성 (Usability)	이해용이성 (Understandability)	플랫폼 코드 가독성을 위해 기본적으로 무기체계 SW 개발 및 관리 매뉴얼 코딩규칙을 따르며, 내부 관리 차원에서 변수 및 번호 체계가 추가로 있을 시 이를 명시해서 모든 플랫폼 개발자는 따른다.	High	Medium
-	강건성 (Robustness)	자동화 도구를 이용하여 코드를 분석(정적 분석)하여 잠재적 오류 제거 활동을 수행한다. 정적 분석은 구현 완료 이후가 아닌 구현 중에 수행한다.	High	Low
유지보수성 (Maintainability)	모듈성 (Modularity)	플랫폼은 모듈화해서 구현해야 한다. 모듈화의 목적은 시험, 디버깅, 분석, 수정, 재사용 등을 용이하게 하기 위함이다.	High	High
유지보수성 (Maintainability)	분석성 (Analyzability)	개발된 플랫폼은 잠재적 오류를 가지고 있다고 가정해서 분석이 용이하도록 설계 및 개발이 이루어져야 한다. 문제가 발생하여 디버깅 시 관련 모듈 또는 함수를 식별할 수 있어야 한다.	High	High
유지보수성 (Maintainability)	수정가능성 (Modifiability)	개발된 플랫폼은 모듈화를 통해 수정이 용이해야 하고, 플랫폼 수정 시 Side Effect를 최소화 하는 방향으로 설계 및 개발이 이루어져야 한다.	High	High
유지보수성 (Maintainability)	시험가능성 (Testability)	플랫폼 안의 세부 모듈은 모두 시험 가능해야 하고, 입출력이 명확히 정의되어야 한다.	High	High
유지보수성 (Maintainability)	재사용성 (Reusability)	플랫폼 모듈의 재사용이 용이하도록 모듈은 필요한 만큼 최대한 작게 분할한다. 단, 가장 작은 모듈도 시험 가능해야 한다.	High	High

국방과학연구소 Agency for Defense Development

- 17 -

국방과학연구소 Agency for Defense Development

결론 및 향후 계획

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 플랫폼 요구사항 분석 완료 (보완 사항 식별됨)

- ✓ 기능 요구사항 263건, 인터페이스 요구사항 24건, 비기능 요구사항 9건
- ✓ 외부 전문가 자문회의('19.12.26.)를 통해서 플랫폼 요구사항 분석 단계 그리고 플랫폼 개발 관련 고민해야 하는 부분들에 대해 의견 수렴 후 보완 사항 식별 (보완 예정)

▶ 플랫폼 통합(Integration) 및 시험(Testing)에 대한 전략/계획 수립

- ✓ 플랫폼 요구사항 분석 단계부터 통합 및 시험에 대한 고민 시작
- ✓ 플랫폼 시험 환경 및 방법(안) 수립 : '19.12.30.

▶ 다음 단계 준비 : 플랫폼 구조 설계, 플랫폼 상세 설계, 플랫폼 구현

- ✓ 플랫폼 구조 설계 : '20년 1분기 완료 (안)
- ✓ 플랫폼 상세 설계 : '20년 2분기 완료 (안)
- ✓ 플랫폼 구현 : '20년 4분기 완료 (안)

국방과학연구소 Agency for Defense Development

- 19 -

ADD
AGENCY FOR DEFENSE DEVELOPMENT

무기체계 항법소프트웨어 플랫폼 피처모델링

2020. 2.

박병수, 백승준, 이인섭, 서강선, 노성규, 박삼준
국방과학연구소

국방과학연구소 Agency for Defense Development

Agenda

- 개요
- 기술적 접근방법
- 피처모델링
- 결언

국방과학연구소 Agency for Defense Development

무기체계 SW 특징 (예: 무인기)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 무인기 체계의 SW 플랫폼 필요성 : 형태, 고도, 성능별 요구 기능은 다르지만 공통으로 활용 가능한 SW 존재



형태별	고정익형 / 회전익형 / 수직 이착륙형
크기별	초소형(0.3m이하) / 소형(0.3-10m) / 중형(10-20m) / 대형(20m이상)
체공별	단기체공(1시간이하) / 중기체공(1-12시간) / 장기체공(12시간이상)
고도별	저고도(3km이하) / 중고도(3-10km) / 고고도(10km이상)
성능별	근거리(50km이하) / 단거리(50-200km) / 중거리(200-500km) / 장거리(500km이상)
기능별	관측 / 탐사, 정찰 / 감시, 표적, 기타

무기체계 SW 특징 (예: 항법SW)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

항법 소프트웨어는 항체의 위치를 찾아내는 기능을 수행하는 소프트웨어로
주요 무기체계에 필수적으로 탑재되는 핵심 소프트웨어임

현상황

- ✓ 탑재되는 무기체계의 특성에 따라 개별적으로 개발
- ✓ 공통적인 요소와 개별적으로 개발되어야 할 요소를 식별하여 전략적인 재사용 필요
- ✓ 다양한 요구사항에 대비 및 품질 향상이 필요

다양한 무기체계에 필수적인 공통기능

무기체계 사업별 독자적인 개발

개발 용이성 및 성능 향상 필요

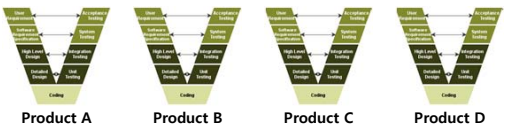
품질, 상호운용성, 유지보수성 제고필요

검증된 소프트웨어 아키텍처 및 공통 컴포넌트의 확보로 무기체계 소프트웨어의 신뢰성 향상

SPL vs Software Reuse

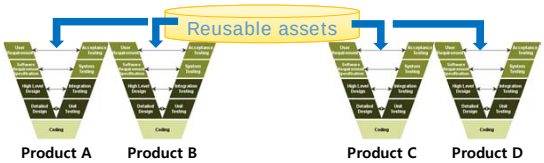
ADD
AGENCY FOR DEFENSE DEVELOPMENT

독립적 개발



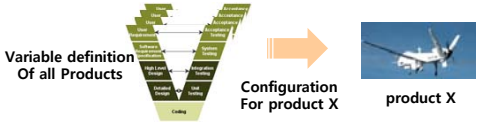
모든 제품은 각각의 개발환경에서 독자적으로 개발

고전적 방식의 Software reuse



재활용 가능한 자산을 활용하여 개발 (각 제품별 규격정의)

SPL(Software Product Line)



공통성과 가변성이 정의된 단일 공통 플랫폼 기반의 제품개발


목표
ADD
AGENCY FOR DEFENSE DEVELOPMENT
무기체계 적용 관성항법시스템
유도
전술지대지 항법SW
천궁 지상항법SW
전술함대지 항법SW
130mm 항법SW
항공
MUAV 항법SW
차기군단무인기 항법SW
공통성·가변성 분석
무기체계 공통 관성항법 SW 플랫폼
피처 모델
항법 SW
항법방식
기준좌표계
센서
상대항법
복합항법
절대항법
기계식 자이로
광학식 자이로
진동형 자이로
플랫폼 요구사항
플랫폼 컴포넌트
위치계산
속도계산
자세계산
동적오차 보상
자체정렬
운항중 정렬
전달정렬
상태감시
플랫폼 설계
체계별 요구사항 분석
체계별 요구사항 분석
AA체계 관성항법시스템 SW
위치계산
자체정렬
보정정보 전처리
고장검출 /분리
속도계산
자세계산
연동SW
연동SW
OS
BB체계 관성항법시스템 SW
위치계산
자체정렬
보정정보 전처리
고장검출 /분리
속도계산
자세계산
연동SW
연동SW
OS
주기
명칭
플랫폼 공통 컴포넌트
명칭
적용 무기체계 특화 컴포넌트
국방과학연구소
Agency for Defense Development
-7-

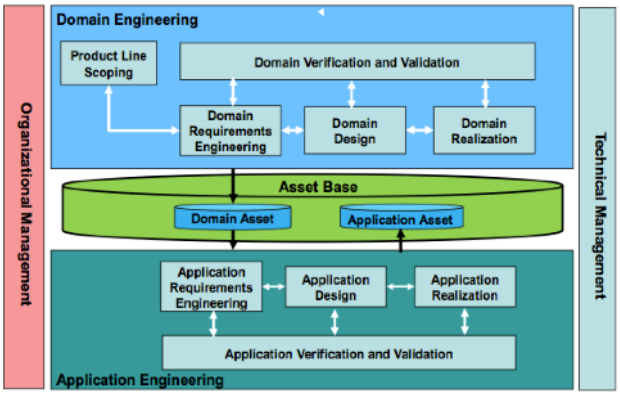

Chapter
기술적 접근방법


국방과학연구소
Agency for Defense Development

ISO/IEC 26550 SPL Reference Model

ADD
AGENCY FOR DEFENSE DEVELOPMENT

ISO/IEC 26550 Software and systems engineering
– Reference model for Product Line Engineering and Management.

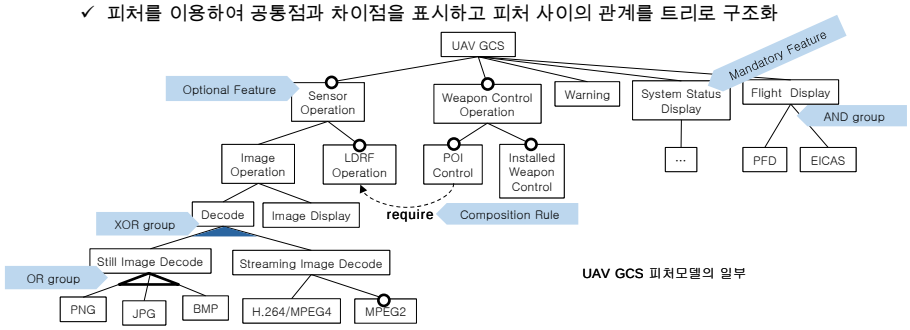


피처모델

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ **피처 (Feature)**
- ✓ 성공적인 SW 제품라인(SPL)을 만들기 위한 핵심은 공통점과 차이점을 이해하는 것
- ✓ 피처는 시스템/소프트웨어의 기능이나 서비스
- ✓ 공통점과 차이점을 식별하는 최소 단위로 사용

▶ **피처모델 (Feature Model)**
- ✓ CMU-SEI[1] 에서 고안된 SW 제품라인(SPL)의 가변성 모델링 기술
- ✓ 피처를 이용하여 공통점과 차이점을 표시하고 피처 사이의 관계를 트리로 구조화

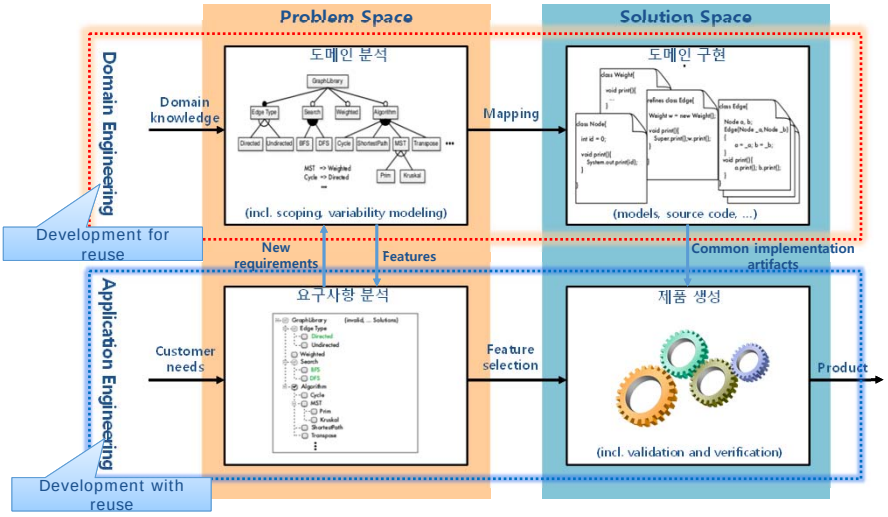


UAV GCS 피처모델의 일부

1) CMU-SEI: Carnegie Mellon Univ. – Software Engineering Institute. FODA(Feature Oriented Domain Analysis) Report, 1990

SPL 운용개념

ADD
AGENCY FOR DEFENSE DEVELOPMENT



무기체계 SPL 프레임워크 확보

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 무기체계 SW를 플랫폼 기반으로 개발할 수 있는 기술/환경

어떤 절차로?

공통 활용할 수 있는 산출물은 무엇을 어떻게 만들 것인가?

효율적으로 개발하고 관리할 수 있는 방법은?

프로세스

SPL 기반 무기체계SW 개발 및 관리프로세스

- SPL 플랫폼 개발 프로세스 · 무기체계 SW 플랫폼을 만들기 위한 절차
- SPL 플랫폼 기반 소프트웨어 개발 프로세스 · 플랫폼 기반의 SW를 개발하기 위한 절차
- 단계별 산출물 정의
- 단계별 업무수행 지침

IDE

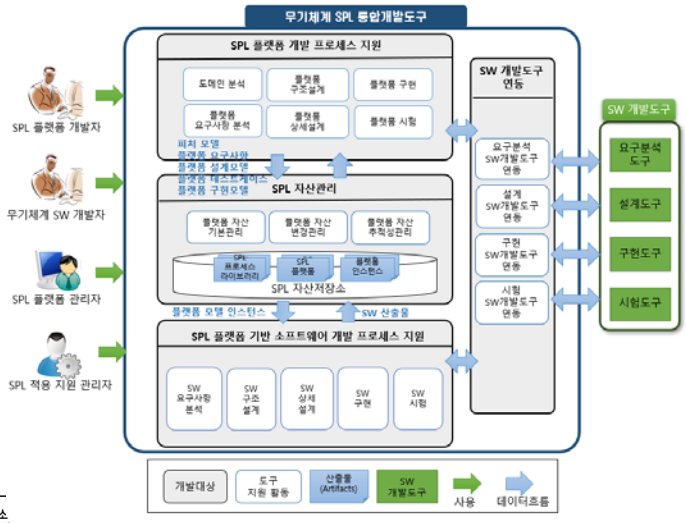
무기체계 SPL 통합개발도구

- SPL 플랫폼 개발 지원도구
- SPL 플랫폼 기반 소프트웨어 개발 지원도구
- 자산 관리 도구
- · SPL 프로세스 지원 도구

무기체계 SPL 통합개발도구

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 무기체계 SPL 통합개발도구(SPLide) 운용개념



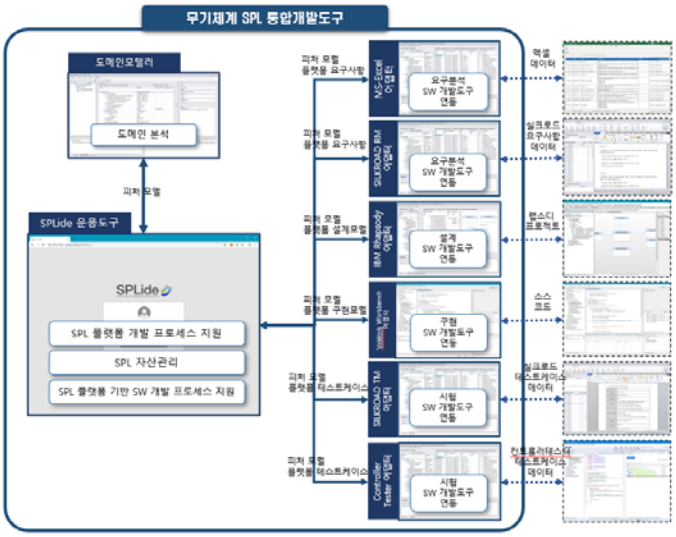
국방과학연구소 Agency for Defense Development

- 13 -

무기체계 SPL 통합개발도구

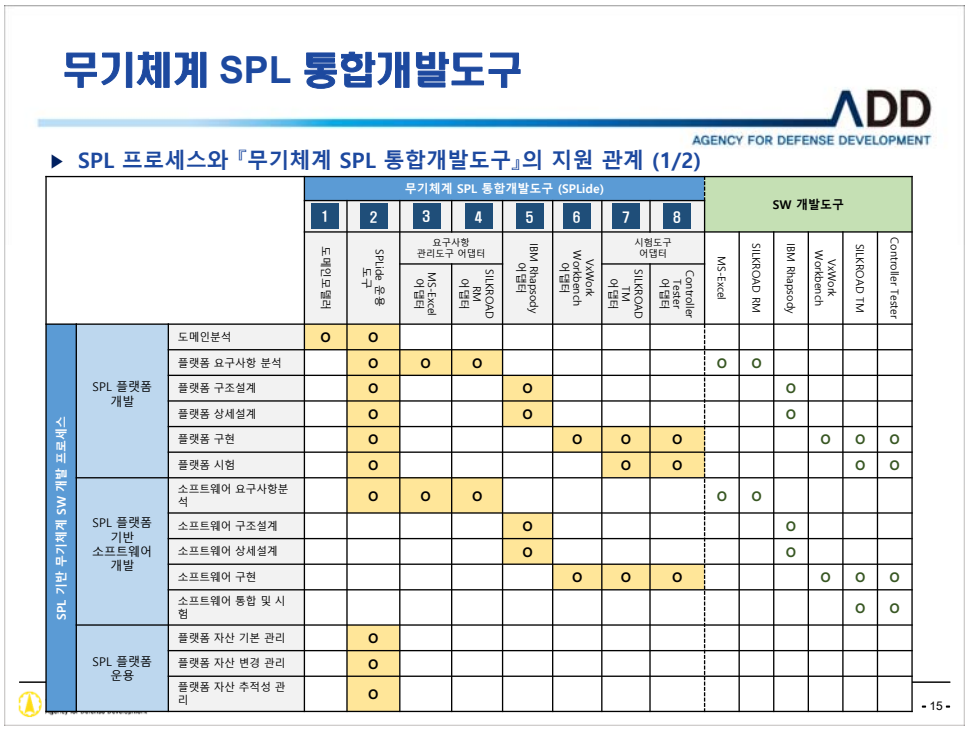
ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 무기체계 SPL 통합개발도구(SPLide)의 구성



국방과학연구소 Agency for Defense Development

- 14 -

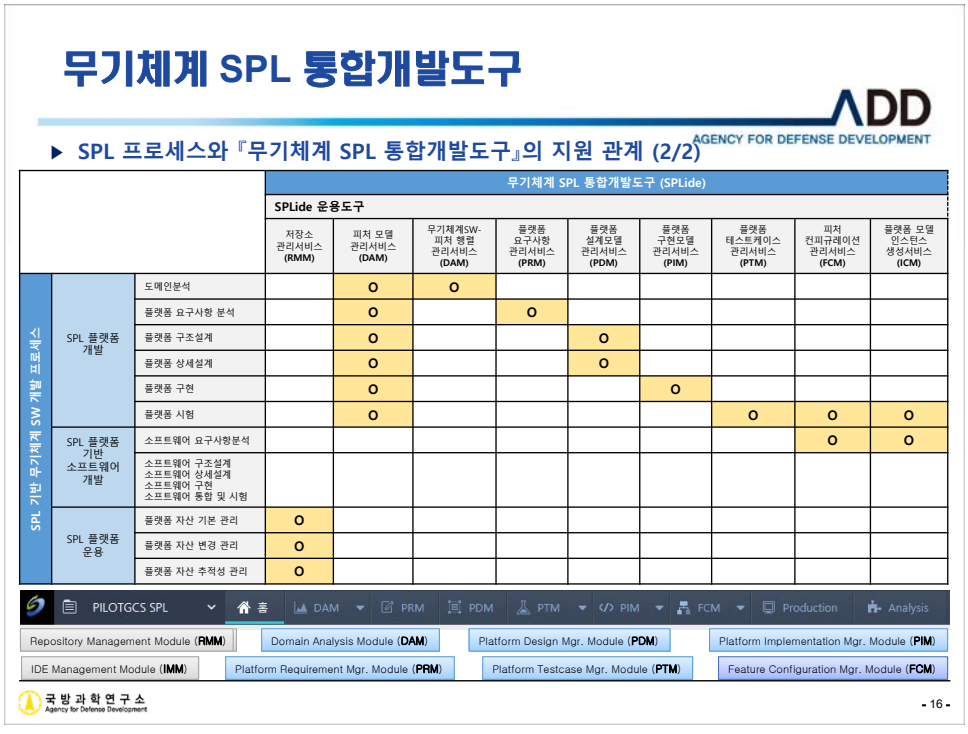


무기체계 SPL 통합개발도구

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ SPL 프로세스와『무기체계 SPL 통합개발도구』의 지원 관계 (1/2)

			무기체계 SPL 통합개발도구 (SPLide)								SW 개발도구					
			1	2	3	4	5	6	7	8						
			도메인모델러	SPLide 운용 도구	요구사항 관리도구 어댑터: MS-Excel 어댑터	요구사항 관리도구 어댑터: SILKROAD RM 어댑터	IBM Rhapsody 어댑터	VxWork Workbench 어댑터	시험도구 어댑터: SILKROAD TM 어댑터	시험도구 어댑터: Controller Tester 어댑터	MS-Excel	SILKROAD RM	IBM Rhapsody	VxWork Workbench	SILKROAD TM	Controller Tester
SPL 기반 무기체계 SW 개발 프로세스	SPL 플랫폼 개발	도메인분석	O	O												
		플랫폼 요구사항 분석		O	O	O					O	O				
		플랫폼 구조설계		O			O						O			
		플랫폼 상세설계		O			O						O			
		플랫폼 구현		O				O	O	O				O	O	O
		플랫폼 시험		O					O	O					O	O
	SPL 플랫폼 기반 소프트웨어 개발	소프트웨어 요구사항분석		O	O	O					O	O				
		소프트웨어 구조설계					O						O			
		소프트웨어 상세설계					O						O			
		소프트웨어 구현						O	O	O				O	O	O
		소프트웨어 통합 및 시험													O	O
	SPL 플랫폼 운용	플랫폼 자산 기본 관리		O												
		플랫폼 자산 변경 관리		O												
		플랫폼 자산 추적성 관리		O												



무기체계 SPL 통합개발도구

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ SPL 프로세스와『무기체계 SPL 통합개발도구』의 지원 관계 (2/2)

			무기체계 SPL 통합개발도구 (SPLide)								
			SPLide 운용도구								
			저장소 관리서비스 (RMM)	피처 모델 관리서비스 (DAM)	무기체계SW-피처 형평 관리서비스 (DAM)	플랫폼 요구사항 관리서비스 (PRM)	플랫폼 설계모델 관리서비스 (PDM)	플랫폼 구현모델 관리서비스 (PIM)	플랫폼 테스트케이스 관리서비스 (PTM)	피처 컨피규레이션 관리서비스 (FCM)	플랫폼 모델 인스턴스 생성서비스 (ICM)
SPL 기반 무기체계 SW 개발 프로세스	SPL 플랫폼 개발	도메인분석		O	O						
		플랫폼 요구사항 분석		O		O					
		플랫폼 구조설계		O			O				
		플랫폼 상세설계		O			O				
		플랫폼 구현		O				O			
		플랫폼 시험		O					O	O	O
	SPL 플랫폼 기반 소프트웨어 개발	소프트웨어 요구사항분석								O	O
		소프트웨어 구조설계 소프트웨어 상세설계 소프트웨어 구현 소프트웨어 통합 및 시험									
	SPL 플랫폼 운용	플랫폼 자산 기본 관리	O								
		플랫폼 자산 변경 관리	O								
		플랫폼 자산 추적성 관리	O								

PILOTGCS SPL | 홈 | DAM | PRM | PDM | PTM | PIM | FCM | Production | Analysis

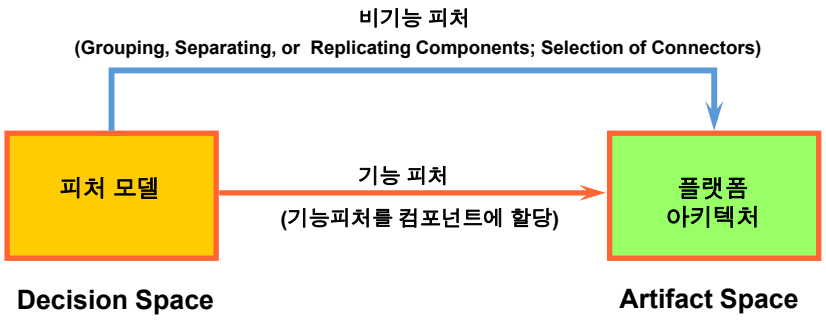
Repository Management Module (RMM) | Domain Analysis Module (DAM) | Platform Design Mgr. Module (PDM) | Platform Implementation Mgr. Module (PIM)

IDE Management Module (IMM) | Platform Requirement Mgr. Module (PRM) | Platform Testcase Mgr. Module (PTM) | Feature Configuration Mgr. Module (FCM)

국방과학연구소 Agency for Defense Development

Feature-based Architecture Modeling

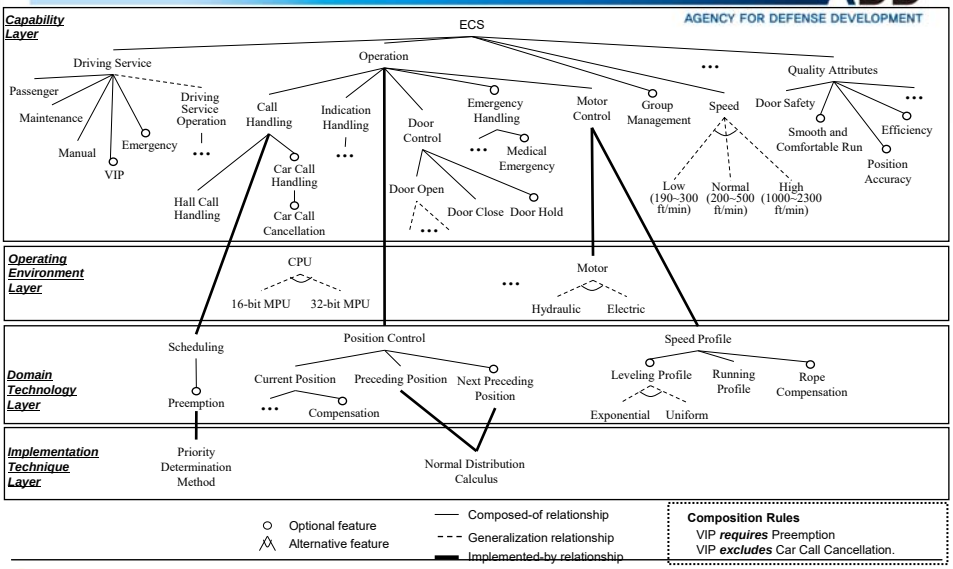
ADD
AGENCY FOR DEFENSE DEVELOPMENT



Mapping Feature Model to Platform Architecture

Feature model for ECS

ADD
AGENCY FOR DEFENSE DEVELOPMENT



Architecture Modeling Process

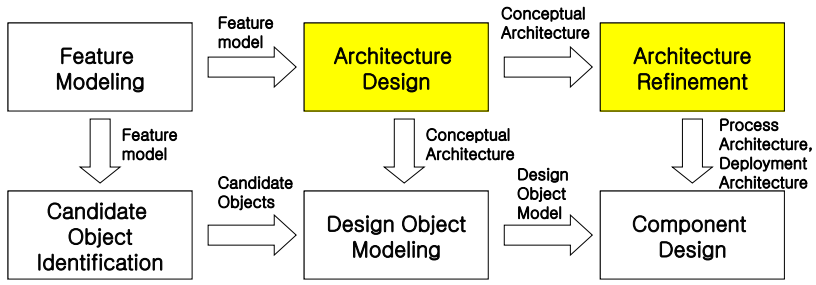
ADD AGENCY FOR DEFENSE DEVELOPMENT

▶ Architecture Design

✓ 피처들을 아키텍처 컴포넌트에 할당하고 해당 컴포넌트들 간의 데이터 및 컨트롤 관련성을 정의 : "conceptual architecture."

▶ Architecture Refinement

✓ 컴포넌트들 간의 상호작용 분석, 각 프로세스 및 네트워크에 할당 등을 통해 conceptual architecture를 "process and deployment architectures"로 정련(refine)



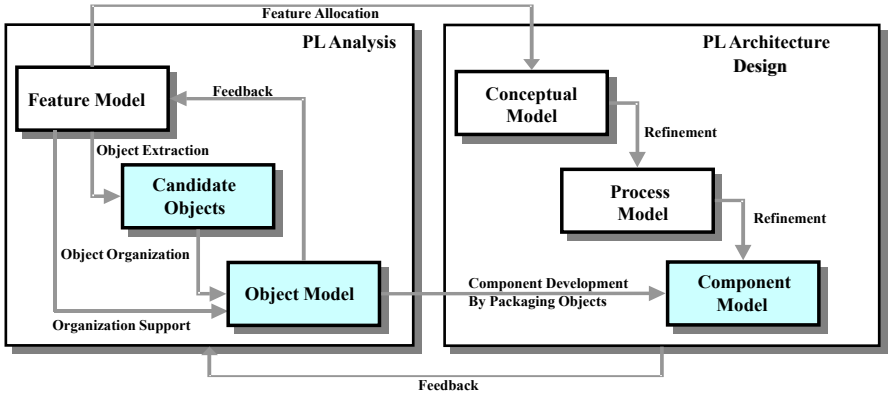
국방과학연구소 Agency for Defense Development

Feature-Based Component Development

ADD AGENCY FOR DEFENSE DEVELOPMENT

▶ 피처 모델로부터 객체 모델을 도출

▶ 객체를 패키징하여 재사용 가능한 컴포넌트로 개발



국방과학연구소 Agency for Defense Development

항법소프트웨어 특징 분석 (1/2)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

- 관성항법장치(INS: Inertial Navigation System)

정 의

관성센서 정보를 활용하여 항체의 자세와 속도, 위치를 제공하는 장치

구 성

- ✓ 자이로 : 각속도 측정
- ✓ 가속도계 : 선형가속도 측정
- ✓ 항법컴퓨터, 신호처리부, 전원공급부, 기구부(제진기능)

원 리

- ✓ 자이로스코프와 가속도계를 이용하여 각속도와 가속도를 측정한 뒤,
- ✓ 각속도, 가속도를 각각 적분하여 자세와 속도를 구하고
- ✓ 구한 속도를 적분하여 위치를 계산한다

특 징

속적인 측정 오차, 알고리즘 오차의 누적에 의한 항법오차의 발산

항법소프트웨어 특징 분석 (2/2)

ADD AGENCY FOR DEFENSE DEVELOPMENT

- 관성항법소프트웨어

구 성 : **정렬, 순수항법, 복합항법**

- ✓ 정렬 : 항법을 수행하기 위한 항법장치의 초기 자세를 찾는 과정
- ✓ 순수항법 : 자세, 속도, 위치 계산
- ✓ 복합항법 : 관성항법장치와 비관성 항법센서(GPS, 영상센서 등) 결합

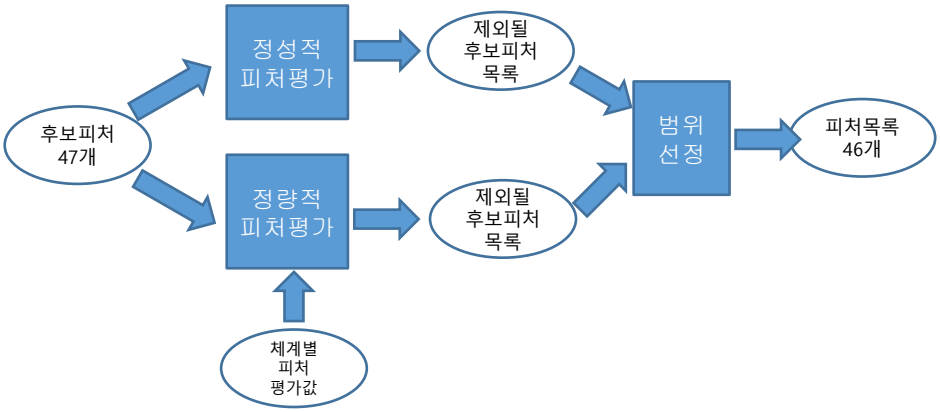
특 징

특 성	내 용
소프트웨어의 다양성	● 관성항법시스템 SW는 대공유도무기, 대함유도무기 그리고 무인기 등 다양한 무기체계에서 운용
소프트웨어 사이의 공통점	● 대부분의 관성항법시스템 SW는 기본적으로 정렬, 순수항법, 보정항법 기능으로 구성
소프트웨어 사이의 차이점	● 관성항법시스템 SW는 무기체계의 요구사항에 따라서 필요한 성능을 만족시키기 위한 다양한 알고리즘 및 기법 사용

SPL 범위 결정 (Scoping) (1/3)

ADD AGENCY FOR DEFENSE DEVELOPMENT

- 분석결과를 바탕으로 식별된 후보 피처를 평가하여 SPL 범위 결정
  - 관성항법시스템 SPL플랫폼 개발 목표를 고려한 범위선정을 진행
  - 정량적 피처평가결과는 범위선정시 참조자료로만 사용



SPL 범위 결정 (Scoping) (2/3)

ADD AGENCY FOR DEFENSE DEVELOPMENT

- 정량적 후보피처 평가결과
  - 47개의 후보 피처 가운데 23개의 피처를 제외 피처 후보로 평가함
  - 특정 체계에만 사용되는 피처는 제외 피처로 식별됨
  - 하드웨어 등 운영환경과 종속적인 피처는 제외 피처로 식별됨
- 플랫폼 적용 범위(Scoping) 평가 결과
  - 총 5개의 무기체계군 가운데 'OO유도무기'를 제외 대상으로 식별
  - 후보 피처목록에 OO유도무기에만 종속적인 피처가 많아서 제외대상으로 식별

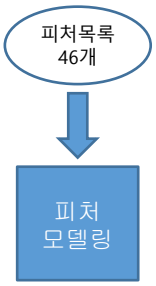
국방과학연구소 Agency for Defense Development

- 25 -

SPL 범위 결정 (Scoping) (3/3)

ADD AGENCY FOR DEFENSE DEVELOPMENT

- 범위 선정 결과
  - 47개의 후보 피처 가운데 하드웨어와 운용환경에 종속적인 '플랫폼제어' 부분만 제외
  - 적용 무기체계별 종속적인 피처들도 '표준화'의 목적으로 SPL 범위에 포함 시킴
- 범위에 포함된 주요피처 목록
  - 신호처리 (SignalProcessing)
  - 정렬 (Alignment)
  - 순수항법 (PureNavigation)
  - 고도안정화 (VerticalDamping)
  - 보정항법 (AidedNavigation)
  - 항법공통라이브러리 (NavigationCommonLib)



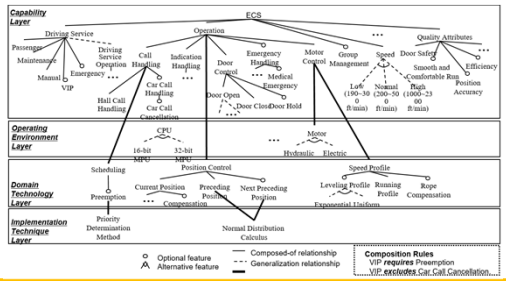
국방과학연구소 Agency for Defense Development

- 26 -

피처 모델링 (1/5)

ADD AGENCY FOR DEFENSE DEVELOPMENT

▶ 피처 모델 레이어 정립

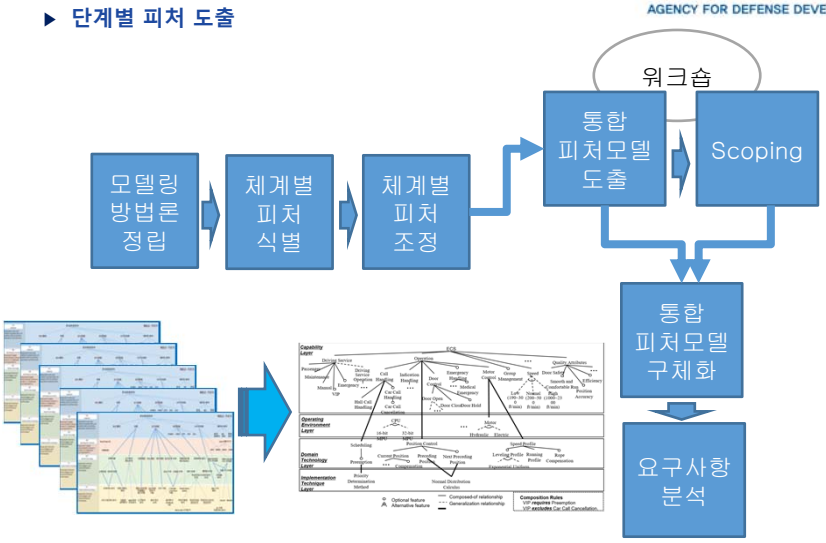


- Capability Layer : 항법 소프트웨어가 제공하는 기능 분류
- Operating Environment Layer : 항법소프트웨어가 탑재되어 운용되는 환경
- Domain Technology Layer : 항법 소프트웨어가 제공하는 기능 구현에 필요한 알고리즘, 로직, 방법론, 기법 등(항법 분야에서만 사용되는 것들)
- Implementation Technique Layer : 여러 분야에서 사용되는 통신기술, 암호화, 범용 알고리즘, 표준 등

피처 모델링 (2/5)

ADD AGENCY FOR DEFENSE DEVELOPMENT

▶ 단계별 피처 도출



피처 모델링 (3/5)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 피처의 가변성 모델링 기법

- ✓ **피처 사이의 구조적 관계를 부모-자식(또는 상위-하위)으로 표현**
- ✓ **카디날리티(Cardinality)를 이용한 가변성 표기법**
- ✓ 택일 피처**그룹과** 다중선택 **피처그룹 표현**
- ✓ 피처 종속적인 속성값의 가변성 표기를 위한 피처어트리뷰트(Feature Attribute) 지원

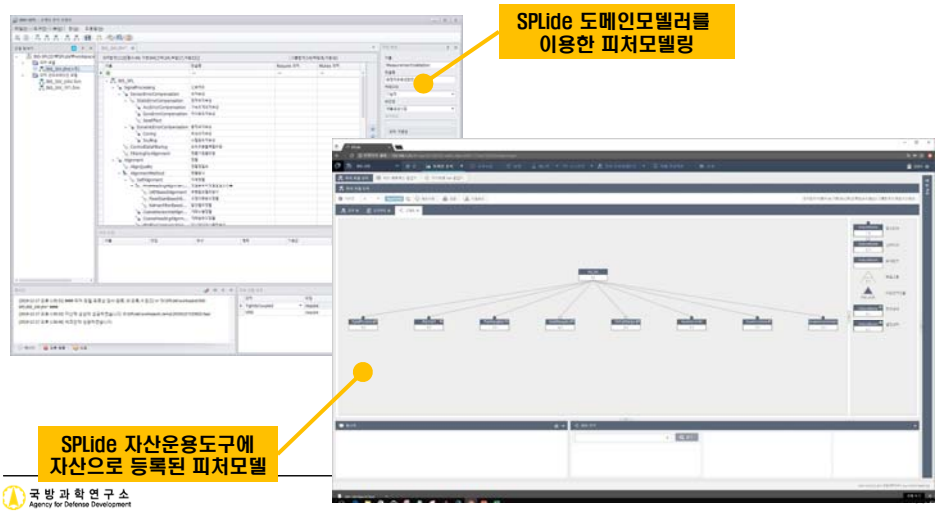
구분	가변성 유형	가변성 표기	의미
피처 가변성	필수 피처 (Mandatory)	**[1..1]**	필수 피처
		[1..m] - m: 최대 피처 인스턴스 개수, 2이상의 자연수	최대 m개의 필수 피처 인스턴스가 생성될 수 있는 필수 피처
	선택적 피처 (Optional)	**[0..1]**	선택적 피처
		[0..n] - n: 최대 피처 인스턴스 개수, 2이상의 자연수	최대 n개의 선택 가능한 피처 인스턴스가 생성될 수 있는 선택적 피처
선택적 피처그룹의 가변성	택일적 피처그룹 (Alternative)	**<1..1>**	그룹에 포함된 선택적 피처를 가운데 반드시 하나는 선택되어야 함
	다중선택 피처그룹	**<n..m>** - n: 0 이상의 자연수 - m: m <= 자식피처의 최대개수	그룹에 포함된 선택적 피처를 가운데 최소 n개에서 최대 m개가 선택되어야 함
피처 어트리뷰트	범위 값	**F1.attrname = {min..max}** - 정수, 실수 유형의 최소/최대값	최소와 최대값 사이에 존재하는 값이 선택될 수 있음
	나열 값	**F1.attrname = {e1, e2, e3}** - 문자열, 정수, 실수 유형의 값	나열 된 값에 존재하는 값이 선택될 수 있음

피처 모델링 (4/5)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 피처모델 정의

- ✓ **SPLide (무기체계SPL 통합개발도구)를 이용**



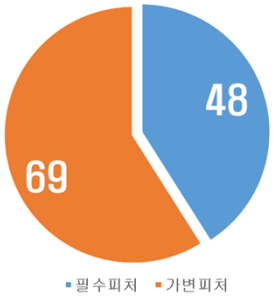
피처 모델링 (5/5)

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ 피처모델 요약

- ✓ 필수피처와 가변피처의 비율이 4:6
- ✓ 체계 종속적인 요소를 SPL 범위에 포함하고 있으므로 가변성이 더 많은 SPL 플랫폼
- ✓ '정렬'과 '보정항법' 가변성의 복잡도가 높은 것으로 분석됨

관성항법시스템 SPL의 피처구성



국방과학연구소 Agency for Defense Development - 31 -

국방과학연구소 Agency for Defense Development

맺음말

ADD
AGENCY FOR DEFENSE DEVELOPMENT

▶ SPL 개념 도입 필요

- ✓ 항법소프트웨어는 주요 무기체계에 필수적으로 탑재되는 핵심 소프트웨어임
- ✓ 탑재되는 무기체계의 특성에 따라 개별적으로 개발
- ✓ 공통적인 요소와 개별적으로 개발되어야 할 요소를 식별하여 전략적인 재사용 필요
- ✓ 다양한 요구사항에 대비 및 품질 향상이 필요

▶ SPL 개발 프레임워크 및 환경 구축

- ✓ 각 단계별 SPL 프로세스 정립
- ✓ 통합개발환경(SPLide) 구축

▶ 도메인 특징 분석 및 피처모델링

- ✓ 항법소프트웨어 특징 분석 및 SPL 적용 범위결정(Scoping)
- ✓ 공통성과 가변성 식별을 통한 피처모델링

국방과학연구소
Agency for Defense Development

기능 안전 표준 기반의 무기체계 소프트웨어 개발 및 관리 매뉴얼 문제점 분석 및 개선 방안 제시

김태현 O, 박다운, 백옥현
국방과학연구소
{tae_hyoun, dwpark90, ohpaek}@add.re.kr,

Problem Analysis and Improvement of Weapon System Software Development and Management Manual Based on Functional Safety Standards

Taehyoun Kim O, Daun Bak, Okhyun Paek
Agency for Defense Department

요 약

최근 기능 안전에 대한 요구가 증가함에 따라 무기체계 분야에서도 관련 활동으로 소프트웨어의 정적 및 동적 분석 활동이 요구되고 있다. 일반적으로 기능 안전 표준은 모든 활동의 선행 조건으로 위험 분석 및 평가를 통한 등급 분류를 요구한다. 하지만 현재 방위사업청의 무기체계 소프트웨어 개발 및 관리 매뉴얼은 이와 관련된 내용이 부족하다. 본 연구에서는 기능 안전 표준을 기반으로 무기체계 소프트웨어 개발 및 관리 매뉴얼의 문제점을 분석하였으며 이에 대한 개선 방안을 제시하도록 한다. 이를 통해 무기체계 소프트웨어 분야 개발자들이 기능 안전 관련 활동을 더욱 원활하게 수행할 수 있기를 기대한다.

1. 서 론

국내 무기체계 소프트웨어 분야 담당자들은 방위사업청에서 발간한 무기체계 소프트웨어 개발 및 관리 매뉴얼[1]에 따라 소프트웨어를 개발해야 한다. 이 매뉴얼에는 크게 소프트웨어 개발, 관리, 지원 측면의 3가지 프로세스를 제공하며 이는 소프트웨어 생명 주기 관련 국제 표준인 ISO/IEC/IEEE 12207 Systems and software engineering - Software life cycle processes[2]를 기반으로 작성되었다.

최근에는 소프트웨어 생명 주기 관련 표준 외에도 기능 안전에 관한 국제 표준의 적용 요구가 증가하고 있다. 기능 안전 표준이란 시스템의 오작동을 방지하기 위해 필요한 기능 안전 관련 활동들을 정의한 문서이다. 대표적인 표준으로는 전자, 전기 분야에 적용되는 IEC 61508 Functional safety of electrical/electronic/programmable electronic safety-related systems[3](이하 'IEC 61508')가 존재한다. 자동차, 철도 등 다양한 산업 분야에서는 IEC 61508을 기반으로 각 분야에 적합한 별도의 기능 안전 표준을 만들어 적용하고 있다.

국내 무기체계 분야에서는 기능 안전 관련 활동으로 개발자에게 소프트웨어의 정적 및 동적 분석 활동을 요구한다. 정적 분석 활동으로는 MISRA-C:2012[4] 등과 같은 코딩 규칙의 준수 및 CWE(Common Weakness Enumeration) 658, 659, 660[5]에 목록화 되어 있는 실행시간 오류 항목의 점검, 함수의 복잡도와 같은 소스코드 메트릭 분석 및 제한 값 준수를 요구하고 있으며 동적 분석 활동으로는 요구사항 기반의 구조적 coverage 달성을 요구하고 있다.

일반적으로 기능 안전 표준은 모든 활동의 선행 조건으로 위험 분석 및 평가를 통한 등급 분류를 요구한다[3,6,7]. 이에 따라 소프트웨어 담당자는 개발하는 소프트웨어 별로 요구되는 활동 수준을 달리할 수 있다. 하지만 방위사업청의 매뉴얼에는 이와 관련된 내용이 부족하다. 따라서 본 연구에서는 기능 안전 표준을 기반으로 등급 분류 관점에서 무기체계 소프트웨어 개발 및 관리 매뉴얼의 문제점을 분석하였으며 이에 대한 개선 방안을 제시하도록 한다.

2. 문제점

2.1 타 산업 분야 대비 다양한 시스템 개발 및 적은 양산 수량으로 인한 등급 분류 기반 데이터 수집/분석의 어려움

무기체계 분야는 다른 산업 분야와 비교했을 때 매우 다양한 시스템을 개발하며 적은 수량을 양산한다는 특징을 갖는다. 이로 인해 특정 분야에 대한 등급 분류 기반 데이터 수집 및 분석이 어렵다는 문제점이 존재한다.

아래 표 1은 방위사업청에서 제공하는 무기체계 분류체계의 대분류 항목[8]이다. 무기체계는 대분류 10종, 중분류 42종, 소분류 132종으로 분류된다. 타 산업 분야에서 적용되는 기능 안전 표준을 적용해보면 자동차 분야 기능 안전 표준 ISO 26262[6]는 기동 무기체계 분야의 일부에만 적용될 수 있으며 항공기 분야 기능 안전 표준 DO-178C[7]는 항공 무기체계의 일부에만 적용될 수 있다.

표 1 무기체계 분류체계 - 대분류

번호	대분류 항목	번호	대분류 항목
1	지휘통제·통신 무기체계	6	화력 무기체계
2	감시·정찰 무기체계	7	방호 무기체계
3	기동 무기체계	8	기타 무기체계
4	함정 무기체계	9	비무기체계
5	항공 무기체계	10	국방정보체계

2.2 시스템 수준의 활동과 소프트웨어 활동 연계 미흡

국내 무기체계 분야에서는 방위력개선사업 수행 시 시스템 수준에서 위험 관리 활동을 수행하도록 요구하고 있으며 이를 위해 방위사업청에서는 2018년에 SE 기반 위험관리 가이드북[9]을 발간하여 제공하고 있다. 하지만 해당 가이드북에서 제시하는 시스템 수준의 위험 분석 및 평가 활동이 소프트웨어 수준의 기능 안전 관련 활동과 연계되고 있지 않다는 문제점이 존재한다.

아래 표 2는 매뉴얼에서 요구하는 정적 분석 및 동적 분석 활동에 대한 등급 분류 기준 적용 현황을 나타내는 표이다. 표 2에서 보는 바와 같이 정적 분석 활동에 대해서는 등급 분류 기준을 적용하고 있지 않으며 개발되는 모든 소프트웨어에 대해 동일한 수준의 활동을 요구하고 있다. 동적 분석 활동에 대해서는 등급 분류 기준을 적용하고는 있으나 시스템 수준의 위험 관리 활동과 별개로 해당 활동을 요구하고 있어 체계 수준의 기준과 소프트웨어 수준의 활동 내용이 일치하지 않을 수 있다는 문제가 존재한다.

표 2 매뉴얼 내 등급 분류 기준 적용 현황

매뉴얼 활동	등급 분류 기준 적용 현황
정적 분석	등급 분류 기준 미적용 및 모든 소프트웨어에 대한 동일한 기준의 활동 요구
동적 분석	시스템 수준의 위험 관리 활동과 별개로 등급 분류 기준 적용 및 활동 요구

2.3 기능 안전 표준의 요구 수준 대비 높은 수준의 동적 분석 활동 요구

매뉴얼에서는 동적 분석 활동을 위해 별도의 등급 분류 기준을 제공하고 있으며 동적 분석 활동은 이 기준에 따라 statement, branch, Modified condition / Decision coverage 달성으로 구분된다. 하지만 매뉴얼에 존재하는 동적 분석 활동의 수준은 다른 산업 분야의 기능 안전 표준[3,6,7]에서 요구하는 동적 분석 활동의 수준보다 상당히 높다. 예를 들어 타 분야의 기능 안전 표준에서는 등급이 낮은 소프트웨어에 대해서는 동적 분석 활동을 요구하고 있지 않으나 방위사업청의 매뉴얼에서는 등급 분류가 낮은 소프트웨어에 대해서도 최소 statement coverage 달성을 요구하고 있다.

3. 개선방안

3.1 무기체계 분류체계 및 국방과학기술 표준분류체계 기준 등급 사전 분류 및 정보 제공

본 연구에서는 방위사업청에서 제공하는 무기체계 분류체계[8] 및 국방과학기술 표준분류체계[10]를 기반으로 시스템 및 하위 구성품에 대한 등급을 사전에 분류하고 이에 대한 정보를 제공해주는 방안을 제시한다.

표 1에서 보는 바와 같이 무기체계 분류체계는 각각의 시스템 유형을 기준으로 무기체계를 분류하고 있으며 표 3에서 보는 바와 같이 국방과학기술 표준분류체계는 시스템 수준이 아닌 하위 구성품 수준으로 기술을 분류하고 있다. 관련 전문가들의 설문 조사 등을 통해 두 분류체계에 대한 등급 분류를 사전에 수행하고 이를 시스템 및 소프트웨어 개발 담당자들에게 제공한다면 기반 데이터 부족으로 인한 담당자들의 어려움을 해소할 수 있을 뿐만 아니라 기능 안전과 관련된 시스템 수준 활동과 소프트웨어 수준 활동 간의 연계도 가능할 것으로 판단한다.

표 3 국방과학기술 표준분류체계 - 대분류

번호	대분류 항목	번호	대분류 항목
1	센서	5	추진
2	정보통신	6	화생방
3	제어전자	7	소재
4	탄약/에너지	8	플랫폼/구조

3.2 기능 안전 표준 수준의 동적 분석 활동 요구

2장에서 언급한 바와 같이 매뉴얼에 존재하는 동적 분석 활동의 수준은 다른 산업 분야의 기능 안전 표준[3,6,7]에서 요구하는 동적 분석 활동의 수준보다 상당히 높다. 개발되는 모든 소프트웨어에 대해 요구사항 기반의 동적 분석 활동을 수행한다면 상당히 많은 시간과 비용이 필요하다. 따라서 투입 노력 대비 효과를 고려하여 기능 안전 표준에서 제시하는 수준의 활동으로 수준을 조정하는 방안이 필요하다. 소프트웨어의 기능 안전을 고려한다면 모든 소프트웨어에 대한 동적 분석 활동 보다는 기능 안전 표준에서 요구하는 생명 주기 별 다른 활동들을 복합적으로 수행하는 것이 더욱 도움이 될 것으로 판단한다.

4. 결론

본 연구에서는 기능 안전 표준을 기반으로 무기체계 소프트웨어 개발 및 관리 매뉴얼의 문제점을 분석하였으며 이에 대한 개선 방안을 제시하였다. 이를 통해 무기체계 소프트웨어 분야 개발자들이 기능 안전 관련 활동을 더욱 원활하게 수행할 수 있기를 기대한다.

참고문헌

[1] 방위사업청, “무기체계 소프트웨어 개발 및 관리 매뉴얼”, 2018년 11월.
[2] ISO/IEC/IEEE, ISO/IEC/IEEE 12207:2017 Systems and software engineering - Software life cycle processes, 2017.
[3] IEC, IEC 61508:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems, 2010.
[4] MISRA, MISRA-C:2012 Guidelines for the use of the C language in critical systems, 2013.
[5] MITRE, https://cwe.mitre.org/
[6] ISO, ISO 26262 Road vehicles - Functional safety, 2012.
[7] RTCA, DO-178C Software Considerations in Airborne Systems and Equipment Certification, 2011.
[8] 방위사업청, 국방과학기술 정보관리 업무지침 별표 2, 2018.
[9] 방위사업청, SE기반 위험관리 가이드북, 2018년 3월.
[10] 방위사업청, 국방과학기술 정보관리 업무지침 별표 3, 2018.

가상화 환경 기반의 무기체계 소프트웨어 규격화 품질향상 방안

최민관 O, 국승학, 이태호
국방과학연구소
{mkchoi, kuk, lth2094}@add.re.kr

An Improvement Method of Weapon System Software Standardization Quality Based on Virtualization Environment

Minkwan Choi O, Seunghak Kook, Taeho Lee
Agency for Defense Department

요 약

최근 무기체계에서 소프트웨어가 차지하는 비중이 증가함에 따라 소프트웨어 개발환경 또한 매우 다양해지고 있다. 무기체계 소프트웨어 분야에서는 국방 규격자료로 소프트웨어 파일목록 및 개발환경을 문서화하도록 요구하고 있다. 하지만 연구개발 종료 후 해당 규격자료를 기반으로 소프트웨어를 재생성하기 위해서는 추가적인 노력이 필요하다. 따라서 본 연구에서 가상화 환경을 활용해 소프트웨어 규격자료의 품질을 향상하는 방안을 제시하고자 한다. 이를 통해 소프트웨어 개발환경 재구축에 대한 노력 절감 및 개발환경 단종으로 인한 문제를 해결 할 수 있을 것으로 기대된다.

1. 서 론

국방 연구개발은 타산업분야와 다르게 규격화 활동을 통해 국방 표준을 지정하고 있다. 국방 표준화(Standardization)는 군수품의 조달·관리 및 유지를 경제적·효율적으로 수행하기 위하여 표준을 설정하여, 이를 활용하는 조직적 행위와 기술적 요구사항을 결정하는 품목지정, 규격제정, 형상관리 등의 지정에 관한 제반활동을 의미한다. 이러한 국방 표준은 군수품의 다양성 감소로 총수명주기비용 절감 및 획득기간 단축, 군수품 상호운용성·호환성·공통성 증진을 통하여 저비용·고효율의 국방자원 운영체제 구축하는데 활용되고 있다. 규격제정을 위한 기술자료로는 국방규격서, 도면, 부품/BOM(Bill of Material), 품질보증요구서(QAR), 소프트웨어 기술문서 등으로 구성된다[1-2].

무기체계에서 소프트웨어 중요성 및 소프트웨어가 차지하는 비중이 증가하고 있는 추세이다. 연구개발하는 소프트웨어 이외에 다양한 상용/공개 소프트웨어 및 개발환경을 활용하여 개발되고 있고 이를 소프트웨어 규격자료에 기술하고 있다. 무기체계 소프트웨어 개발 및 관리 매뉴얼에서는 소프트웨어 분야에서 규격화 해야하는 내용을 다루고 있지만, 개발한 소프트웨어가 완전한 기능을 수행하기 위해서 필요한 개발환경 및 개발하는 소프트웨어 이외의 부분에 대해서는 제한적으로 다루고 있다. 따라서 본 연구에서는 무기체계 소프트웨어 규격화 현황에 대해 소개하고 규격화 품질 향상 방안에 대하여 제안하고자 한다.

2. 소프트웨어 규격화 현황 및 한계점

규격화 활동은 그림 1의 무기체계 소프트웨어 개발 프로세스에서 개발 및 시험평가가 완료된 이후에 수행한다. 무기체계 소프트웨어 개발 및 관리 매뉴얼에서는 소프트웨어 규격화 시 소프트웨어 기술문서, 각종 컴퓨터 파일(소스코드 및 체계의 실행과 관련된 모든 프로그램 파일) 등을 관련 규정·지침(방위사업관리규정, 국방규격·표준서의 서식 및 작성에 관한 지침, 표준화 업무 지침)에 따라 표준화 업무 지원 시스템인 국방표준종합정보시스템(KDSIS, Korea Defense Standard Information System)에 제출하도록 요구하고 있다[1-4].

* SDP: SW개발계획서, SRS: SW요구사항명세서, SDD: SW설계기술서, IDD: I/F설계기술서, DBDD: DB설계기술서, STP: SW통합시험계획서, STD: SW통합시험절차서, STR: SW통합시험결과보고서, FIG: 컴퓨터설치절차서, SVD: SW버전기술서, SPS: SW산출물명세서, SCS: SW목록명세서, SIP: SW설치계획서

그림 1 무기체계 SW 개발 프로세스

KDSIS에 제출되는 소프트웨어 규격자료의 구성은 그림 2와 같다. 실제 자료는 소프트웨어 형상항목(CSCI, Computer Software Configuration Item)별로 소프트웨어 기술문서(SRS, SDD, STP, STD, STR, SPS, SIP) 7종 및 소프트웨어소스(소스패키지), 실행파일을 포함하고 있다. 분류체계/속성자료는 소프트웨어 개발 및 유지보수, 재활용

업무에 사용되는 자료로 CSCI의 하위 구성요소인 CSC(Computer Software Component)별로 작성하고 있다[5].

그림 2 국방표준종합정보시스템(KDSIS) 메타데이터 구성

소프트웨어 기술문서 중 파일목록(소프트웨어소스, 실행파일) 및 생성/탑재 절차를 기술한 문서는 SPS(Software Product Specification)이다. 그림 3은 SPS, 도면 내 파일목록 및 개발환경 작성 현황을 도식화한 것이다. SPS에 기술된 소스코드, 프로젝트 파일 등 각종 파일을 이용하여 실행파일을 생성 및 타겟 하드웨어에 탑재한다. SPS에 명시한 모든 파일은 KDSIS에 제출 및 관리되고 있지만 개발환경은 개발 당시에 사용한 개발환경 정보(운영체제, 소프트웨어 개발키트, 컴파일러 등)만 간략히 명시하고 있다. 무기체계(임베디드) 소프트웨어 개발환경을 구축하는데 많은 노력이 필요하다. 호환성이 떨어지는 다양한 컴파일러와 보안 이슈로 폐쇄망에서만 운영하고 있다. 또한 타산업분야와 다르게 연구개발 기간이 길어 주기적인 컴퓨터 하드웨어 업그레이드, 개발자 퇴사 등의 문제로 기 구축해놓은 개발환경 유지에 어려움을 겪고 있기 때문에 보완이 필요한 상황이다.

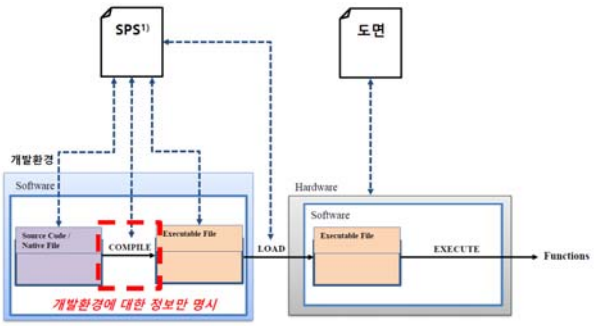


그림 3 SW 규격자료 내 '파일목록 및 개발환경' 작성현황

3. 가상화 환경 기반의 소프트웨어 규격화 적용방안

소프트웨어 개발환경을 가상화 기술을 활용하여 규격화하는 방안을 제시하고자 한다. 본 논문에서 언급한 가상화(Virtualization)라 함은 하나의 물리적 시스템을 논리적으로 분할해 여러 대의 컴퓨터처럼 보이게 하는 기술을 의미한다. 해당 기술을 활용하여 물리적인 컴퓨터/시스템에서 소프트웨어로 구성해 만들어진 가상의 컴퓨터 시스템을 가상머신(Virtual Machine)이라 한다[6].

본 논문에서 제안하는 방안은 그림 4와 같이 SPS에서 개발환경 정보만 명시한 부분을 개발환경이 구축된 가상화 환경 이미지 파일을 규격자료에 포함시키고자 한다.

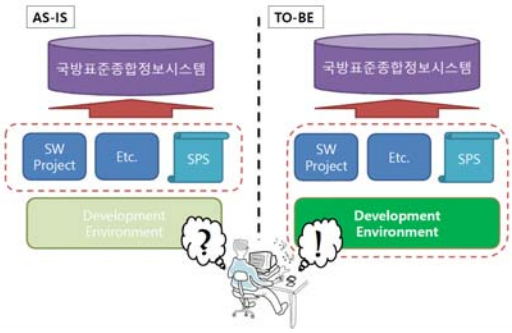


그림 4 개발환경이 구축된 가상환경 이미지 파일 추가(안)

현재는 KDSIS에 존재하는 산출물(SPS, 소프트웨어소스)을 기반으로 개발환경을 새로 구축해야 하지만 제안한 방안에 따라 KDSIS에 제출한 가상화 환경 이미지 파일을 재활용하면 된다. 또한 개발환경 단종으로 인한 문제를 해결할 수 있다. 현재는 개발환경 단종 시 기존의 개발환경 구축에 필요한 각종 파일 회득 및 재구축에 어려움이 있었지만 가상화 환경 이미지 파일을 활용하면 된다. 제안한 방안을 적용하기 위한 방법으로 개발환경이 구축된 가상화 환경 이미지 파일 추가 제출 방법은 표 1과 같다. 현재는 소프트웨어소스(실행, 소스, 헤더, 오브젝트 등 각종 파일) 및 소프트웨어 실행파일(하드웨어에 탑재되는 파일)을 구분하고 있다. 변경(안)은 빌드 시 필요한 파일과 탑재 시 필요한 파일로 구분한다. 가상화 환경 이미지 파일은 소스코드 빌드 시 필요한 개발환경이기 때문에 소프트웨어소스에 포함된다.

표 1 가상화 환경 이미지 파일 추가 방법(안)

구분	현재	변경
소프트웨어소스	실행, 소스, 헤더, 오브젝트 등 각종 파일	- 원시파일 - 빌드에 필요한 각종 파일 - 가상화 환경 이미지
소프트웨어실행파일	실행파일	- 실행파일 - 실행에 필요한 각종 파일

4. 결 론

본 논문에서는 무기체계 개발에서 수행하는 소프트웨어 규격화 품질을 향상하기 위해 가상화 환경을 활용하는 방안을 제안하였다. 연구개발 당시의 소프트웨어 개발환경을 가상화 기술을 통해 구축함으로써 사업 종료 이후 소프트웨어 수정 및 실행파일 재생성을 위한 개발환경 재구축 노력 절감 및 개발환경 단종으로 인한 문제를 해결할 수 있을 것으로 기대된다.

참고문헌

[1] 방위사업청 예규 제484호, 표준화 업무지침.
[2] 방위사업청, 국방표준업무 실무 가이드북, 2017년 10월.
[3] 방위사업청 예규 제485호, 국방규격·표준서의 서식 및 작성에 관한 지침.
[4] 방위사업청 매뉴얼 제2018-7호, 무기체계 소프트웨어 개발 및 관리 매뉴얼.
[5] 방위사업청, 국방표준종합정보시스템, 2017년 8월.
[6] 안성원, 클라우드 가상화 기술의 변화(SPRi 인사이트리포트 제2018-004호), 2018년 12월.

텍스트 마이닝과 TF-IDF 유사도 가중치를 이용한 연관 아이템 발견

김민정 [1O], 김주창 [2], 박성수 [3], 신동훈 [4], 정호일 [5], 정경용 [6]

[1,2,3,4]경기대학교 컴퓨터과학과 데이터마이닝 연구실

[5]대림대학교 컴퓨터소프트웨어과, [6]경기대학교 컴퓨터공학부

minjeog0513@kyonggi.ac.kr, kjc2232@naver.com, wolfboy@kyonggi.ac.kr,
dhshin8222@gmail.com, hijung@daelim.ac.kr, dragonhci@gmail.com

Discovery of Associative Items using Text Mining and TF-IDF Similarity Weight

Min-Jeong Kim[1O], Joo-Chang Kim[2], Sungsoo Park[3], Donghoon Shin[4],
Hoill Jung[5], Kyungyong Chung[6]

[1,2,3,4]Data Mining Lab., Department of Computer Science, Kyonggi University

[5]Department of Computer Software, Daelim University College

[6]Division of Computer Science and Engineering, Kyonggi University

요 약

스마트 디바이스는 보급률이 증가하고 다양한 용도로 사용되며 현대사회에 없어서 안될 필수품이 되었다. 이에 따라 모바일 시장에서 인공지능을 완성하는 다양한 소프트웨어 모델이 제시되고 있다. 이로 인해 사용자에게 정확한 의사결정을 위해 텍스트 마이닝을 통해 연관성을 추출하여 제공하는 것이 필요하다. 또한 상호 유의미한 아이템의 집합에서 중요도를 측정하여 추출하는 방법이 요구되고 있다. 따라서 본 논문에서는 텍스트 마이닝과 TF-IDF 유사도 가중치를 이용한 연관 아이템 발견하는 방법을 제안한다. 이는 사람들의 구매패턴에서 연관규칙을 생성하고 키워드 분석을 기반으로 유사도 가중치를 통해 아이템을 비교하여 추천한다.

1. 서론

스마트폰의 상용화로 인해 모바일 어플리케이션의 이용자가 급증하고, SNS 와 네트워크의 발달로 인해 언제 어디서든 다른 사용자들과 자신의 의견을 공유할 수 있다. 다른 사용자의 평점과 리뷰는 새로운 사용자의 아이템을 선택에 많은 영향을 미친다. 기존의 추천시스템은 명확하게 수치화 되는 정량적인 정보를 주로 이용해 유사한 사용자들에게 추천한다[1]. 정량적인 정보는 사용자의 기준에 따라 동일한 아이템이라도 평점이 달라질 수 있기 때문에 정확하게 판단하지 못하는 단점이 존재한다. 또한 아이템의 정보를 얻기 위해 사용자들이 정량적인 정보보다 리뷰와 같은 정성적인 정보에 초점을 맞추고 있기 때문에 정성적인 정보의 중요성이 증가하고 있다. 평점은 높지만 리뷰 텍스트는 부정적인 견해를 보이는 경우가 존재하며, 동일한 평점이지만 사용자의 견해의 차이가 존재한다. 이로 인해 평점과 사용자의 주관적인 의견을 모두 반영해야 한다. 따라서 본 논문은 텍스트 마이닝과 TF-IDF 유사도 가중치를 이용한 연관 아이템 발견을 제안한다. 이는 연관 마이닝을 통해 사용자의 최근 구매목록에서 구매패턴에 대한 연관규칙을 생성하고 사용자 리뷰에 따른 감성분석을 통해 변환된 TF-IDF 가중치를 구축한다. 또한 변환된 가중치 매트릭스를 사용하여 높은 유사성을 갖는 아이템을 도출한다. 이를 통해 사용자 개인의 선호도와 리뷰 내에 존재하는 빈도를 이용하여 보다 개인화되고 정확한 아이템 추천이 가능하다.

2. 관련연구

2.1 TF-IDF

텍스트 마이닝에서 TF-IDF(Term Frequency-Inverse document Frequency)는 검색하거나 특정한 키워드를 추출하기 위해 자주 이용되는 가중치이다[2,3]. 이는 단어와 문서의 빈도를 기반으로 하여 문서 내에서의 특정

단어에 대한 중요도를 측정하는 기법이다. TF(Term Frequency)는 문서내에서의 단어의 빈도를 나타내며, 값이 높을수록 문서에서 자주 등장함을 의미하지만 이는 또한 흔하게 분포한다는 것을 의미한다. 이를 보완하기 위해 IDF(Inverse Doucument Frequency)와의 곱을 이용한다. IDF 는 역문서 빈도를 의미하며 DF(Document Frequency)의 역수 값을 나타낸다. 이는 특정한 단어가 문서 내에서 나오지 않아 분모가 0 이 되는 것을 막기 위해 분모에 1을 더한다. 또한 문서 양에 따라 IDF 값이 급격하게 증가하는 것을 막기 위해 단순히 역수를 취하는 것이 아닌 로그 값을 사용한다. TF-IDF 값은 전체 문서에서 빈번하게 등장하는 단어일 수록 낮은 중요도를 가지며, 특정 문서에서 빈번하게 등장할수록 높은 중요도를 가지도록 가중치를 부여한다.

2.2 형태소 분석

자연어 처리는 텍스트 안에 존재하는 의미있는 정보를 분석하고 추출하는 과정을 말한다[4,5]. 이는 문서의 키워드 추출, 감성분석, 카테고리 분류에서 자주 사용되고 있다. 자연어 처리를 하기 위해서 형태소 분석 과정을 거친다. 형태소 분석은 원시적인 말뭉치를 최소 단위로 나누어 각 문장의 언어 구조를 파악하는 것이다. 한국어 형태소 분석기 라이브러리로 R에서 지원하는 KoNLP와 파이썬에서 지원하는 KoNLPy 등이 존재한다[6]. KoNLP는 한글 자연어를 분석 가능한 패키지로 형태소를 분석할 수 있는 27개의 함수가 내장되어 있으며 이를 통해 자연어 처리가 가능하다. KoNLPy는 C, C++, JAVA등의 언어를 통해 형태소분석 할 수 있도록 구현된 오픈 패키지이다. KoNLPy 패키지는 5개의 형태소 분석기로 구성되어 있다. 이는 Hannanum, Kkma, Komoran, Twitter, Mecab를 포함한다. 문장의 토큰화를 거친 뒤 형태소에 품사를 태깅하는 작업을 통해 정보를 파악하고, ‘나’, ‘을’, ‘를’, ‘에’, ‘게’, ‘으로’ 등과 같은 큰 의미를 지니고 있지 않은 불필요한 불용어들을 제거함으로써 형태소 분석의 정확도를 증가시킨다.

3. 텍스트 마이닝과 TF-IDF 유사도 가중치를 이용한 연관 아이템 발견

3.1 데이터 수집

아이템 추천을 위해 사용자들의 구매목록 정보와 아이템에 대한 리뷰 데이터를 이용한다. 우선적으로 SNS와 인터넷 커뮤니티에서 설문조사를 진행하여 어플리케이션 마켓을 이용하는 사용자의 구매목록 데이터를 수집한다. 사용자의 구매목록 중 가장 최근 구매한 아이템 5가지로 질문을 구성한다. 수집된 97명의 데이터 중 부정확한 이름을 적거나 결측치 값이 존재하는 데이터를 제외한 82명의 데이터를 활용한다.

각각의 아이템에 대한 리뷰를 수집하기 위해 R 프로그램에서 RSelenium 패키지와 rvest 패키지를 이용하여 어플리케이션 마켓의 리뷰 데이터를 스크래핑한다[7]. 총 47개의 아이템에서 텍스트 형태로 구성된 1529개의 작성자, 평점, 리뷰 데이터를 수집한다. 평점은 1점부터 5점까지 구성되어 있다. 스크래핑이란 웹페이지에 존재하는 데이터 중 필요한 정보를 추출하는 것을 의미한다. rvest 패키지를 이용하면 간편하게 스크래핑이 가능하지만 html 로 구성된 페이지만을 수집할 수 있다는 문제점이 존재한다. RSelenium 패키지는 사용자가 동작해야 되는 행동들을 코드로 설정하여 웹 내에서 수행할 수 있도록 한다. 따라서 RSelenium 을 이용하여 자바스크립트로 구성된 반응형 웹의 동작들을 수행한 뒤 rvest를 이용하여 평점과 리뷰 내용을 반자동적으로 스크래핑한다. 웹페이지 중 불필요한 정보를 제외한 작성자, 평점, 리뷰 데이터만을 수집하기 위해서 HTML 의 구조의 파악이 필요하다. 각각의 추출요소들이 속해 있는 상위 클래스와 하위클래스의 구조를 파악하여 해당 클래스 안에 있는 텍스트 정보들을 추출한다.

3.2 감성분석 기반 리뷰 분류와 TF-IDF 산출

수집한 리뷰 데이터 중 결측치가 존재하는 데이터를 제외시킨 뒤 형태소 분석 과정을 거친다. 본 논문에서는 R에서 지원하는 KoNLP 패키지를 활용한다. 자주 등장하지만 큰 의미를 가지고 있지 않는 불용어들을 제거하기 위해 불용어 사전을 통해 471개의 불용어를 처리한 뒤 감성분석에서 자주 사용되는 품사 중 형용사, 명사, 동사 만을 선별하여 추출한다[7,8]. 형태소 분석을 통해 추출한 키워드들을 문서 단어 행렬(Document Term Matrix)로 표현한다. 문서 단어 행렬은 단순히 키워드의 빈도 수를 기반으로 접근하기 때문에 이를 보완하기 위해 TF-IDF 가중치를 사용한다. 사용자의 어플리케이션 리뷰 데이터를 기반으로 감성분석을 통해 각 어플리케이션에 대한 긍정, 부정, 중립의 TF-IDF 가중치 행렬을 구성한다. 감성분석을 하기 위해 KOSAC(Korean Sentiment Analysis Corpus)에서 제공하는 긍정, 중립, 부정의 극성정보 수치로 이루어진 감성사전을 이용한다. 이는 16362개의 n-gram을 구성하는 형태소로 이루어져 있으며 긍정과 부정의 수치, 가장 높은 비율을 차지하는 극성 값의 이름과 수치로 구성되어 있다. 리뷰 텍스트를 형태소 분석하여 긍정적인 단어일 경우 긍정 수치 값을 더해주고, 부정적인 단어일 경우 부정 수치 값을 빼준다. 해당 리뷰의 극성 값 정도가 양수일 경우 긍정적인 리뷰, 음수일 경우 부정적인 리뷰, 0에 근접할수록 중립적인 리뷰로 분류한다.

3.3 TF-IDF 가중치 기반 아이템간 유사도 비교

극성값의 수치로 분류한 리뷰들을 기반으로 각 아이

템에 대해 긍정, 부정, 중립의 단어 가중치 행렬을 구성한다. 표1은 각 아이템에 대한 긍정, 부정, 중립의 리뷰 중 긍정적인 리뷰에 대한 단어 TF-IDF 가중치 행렬을 나타낸다. 표1의 단어들은 한글 형태소 품사(Part Of Speech 태그 집합 중 P(용언), N(체언)을 의미한다.

표1. 아이템의 긍정 리뷰에 대한 단어 가중치 행렬

	Item1	Item2	Item3	Item4	Item5	…
좋/P	0.713	0.546	0.829	0.683	0.439	…
편리/N	0.515	0.490	0.326	0.398	0.621	…
최고/N	0.316	0.637	0.297	0.401	0.127	…
추천/N	0.425	0.275	0.452	0.417	0.253	…
빠르/P	0.341	0.469	0.147	0.285	0.501	…
만족/N	0.608	0.398	0.495	0.577	0.332	…
…	…	…	…	…	…	…

TF-IDF 가중치 매트릭스를 이용하여 아이템 간의 유사도를 구하기 위해 코사인 유사도(Cosine Similarity)를 이용한다[9]. 코사인 유사도 값은 -1과 1사이의 값을 가진다. -1과 근접할수록 유사도가 낮다고 판단하며, 1과 가까운 값 일수록 유사도가 높다고 판단한다. 식(1)은 코사인 유사도식을 나타낸다.

$$\textbf{Similarity} = \frac{(A_1+A_2+\cdots+A_n)\times(B_1+B_2+\cdots+B_n)}{\sqrt{A_1{}^2+A_2{}^2+\cdots+A_n{}^2}\times\sqrt{B_1{}^2+B_2{}^2+\cdots+B_n{}^2}} \quad \text{식(1)}$$

그림1은 단어의 TF-IDF 가중치 행렬을 통해 아이템 간의 코사인 유사도 계수를 나타낸 것이다. 파란색일수록 아이템들의 유사도가 높음을 의미하고, 빨간색일수록 유사도가 낮음을 의미한다. 가로축과 세로축은 47개의 아이템들을 나타내고, 일직선으로 이루어진 파란 점들은 자기자신과의 비교이기 때문에 유사도가 1이 된다.

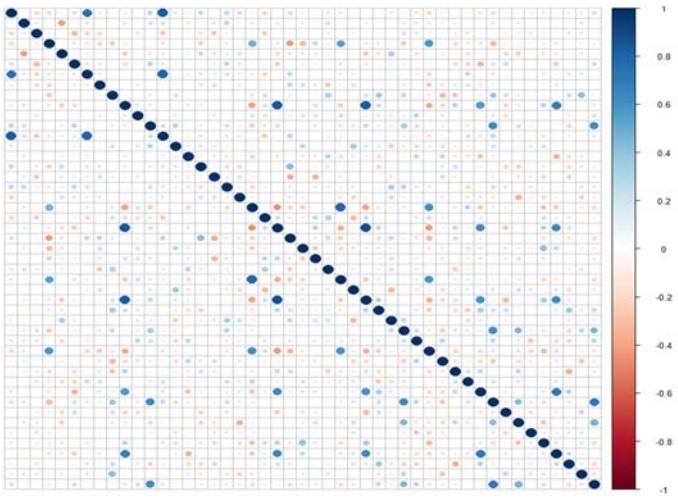


그림1. 아이템 간의 유사도 계수

3.4 아이템 구매 연관규칙 발견

수집한 사용자의 구매목록 데이터를 기반으로 Apriori 알고리즘을 사용하여 아이템 구매패턴을 분석한다[10]. 연관 규칙에서 사용될 구매 목록 데이터는 총 47개의 아이템들을 비식별화하여 Item1, Item2, …, Item47과 같이 구성한다. 표2는 사용자의 최근 구매목록 내역에 따른 아이템 구매패턴 연관규칙 생성 결과를 나타낸다. 표2에서 Conf.는 신뢰도(Confidence), Lift는 향상도(Lift), Sup.는 지지도(Support)를 나타낸다. 또한 Item3 => Item42, Item33는 아이템3을 구매했을 경우 아이템 42와 33을 구매할 가능성이 높음을 연관규칙을 나타낸다. 이와 같이 사용자의 구매목록 데이터를 기반으로 구매패턴 연관규칙을 생성하여 사용자가 선호할 것이라고 예상되는 아이템을 발견할 수 있다.

표2. 사용자의 구매목록에 따른 아이템간 연관규칙

Rule	Conf.	Lift	Sup.
Item15 => Item21, Item47	1	4.82	4
Item29, Item39 => Item21	1	4.1	3
Item9 => Item40	0.83	3.42	3
Item25 => Item13. Item30	0.67	2.73	5
Item39 => Item17, Item27, Item32	0.63	2.33	3
…	…	…	…

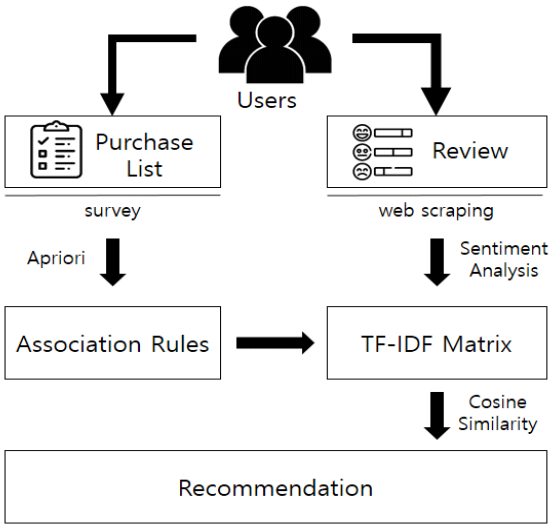


그림2. 시스템 프로세스

그림2는 제안하는 방법의 시스템 프로세스를 나타낸다. 제안하는 방법은 사용자들의 구매목록을 설문조사를 통해 수집하고, 각 아이템의 리뷰 데이터를 스크래핑한다. 구매목록 데이터를 Apriori 알고리즘을 통해 연관규칙을 생성한다. 수집한 리뷰데이터를 감성분석을 통해

TF-IDF 매트릭스를 구축한다. 연관규칙을 통해 추출된 아이템 중 코사인 유사도를 이용하여 유사도가 높은 아이템을 도출하여 최종적으로 아이템을 추천한다. 본 논문에서는 아이템을 어플리케이션으로 선정하였지만 어플리케이션뿐만 아니라 영화, 여행, 쇼핑 등 다양한 콘텐츠에 접목 가능하다.

4. 성능평가

제안하는 추천 방법의 성능을 평가하기 위해 기존에 수집한 사용자들의 구매목록을 이용한다. 최근 5개의 구매목록에서 가장 먼저 구매한 아이템을 기준으로 새로운 아이템을 추천하고, 남은 4개의 구매목록과 비교한다. 제안하는 방법은 TF-IDF 가중치에 따른 아이템의 유사도와 사용자 구매목록에 따른 연관규칙을 사용하기 때문에 성능평가의 비교대상으로 유사도 기반 추천방법과 규칙기반 추천방법을 이용한다. 따라서 제안하는 TF-IDF와 연관 마이닝 기반 추천방법(TAbR)과 유사도 기반 추천방법(SbR), 규칙 기반 추천방법(RbR)을 비교한다. 각각의 방법으로 아이템을 추천하고, 추천된 어플이 구매 목록에 있는지 비교하여 평가한다. 표3은 성능평가 결과를 나타낸다. 제안하는 TAbR방법의 정확도는 0.768, F-Measure는 0.855로 다른 추천방법에 비해 더 우수하게 추천이 가능함이 나타난다.

표3. 성능평가 결과

Measure	TAbR	SbR	RbR
Recall	0.8280	0.6925	0.8112
Precision	0.8837	0.8496	0.5416
Accuracy	0.7680	0.6560	0.6401
F-Measure	0.8550	0.7630	0.6495

5. 결론

빅데이터 컴퓨팅 기술의 발전과 다양한 스마트 기기 보급률 증가로 인해 소프트웨어 분야의 어플리케이션이 고도화되고 있다. 다양한 어플리케이션에서 선택의 폭이 넓어짐에 따라 사용자의 선호도에 맞는 아이템을 추론하기 어렵다. 따라서 본 연구에서는 아이템과 사용자의 선호도를 모두 고려한 텍스트 마이닝과 TF-IDF 유사도 가중치를 이용한 연관 아이템을 발견하는 방법을 개발하였다. 자연어로 이루어진 비정형 또는 정형의 집합으로부터 패턴이나 특징을 발견하기 위해 텍스트 마이닝을 사용한다. 사용자의 구매목록 정보를 기반으로 연관규칙을 생성하여 구매패턴을 추론한다. 또한 리뷰 데이터를 수집하여 형태소분석을 통해 긍정, 부정, 중립의 리뷰로 분류한다. 이와 같이 분류한 리뷰를 통해 단어의 TF-IDF 가중치 행렬을 구성하고, 유사도를 비교하여 아이템을 추천한다.

지원문구

This work was supported by the GRRC program of Gyeonggi province. [GRRC KGU 2018-B05, Smart Manufacturing Application Technology Research]

참고문헌

[1] B. K. Jeon, H. C. Ahn, “A Collaborative Filtering System Combined with Users Review Mining: Application to the Recommendation of Smartphone Apps”, Journal of Intelligence and Information Systems, Vol.21, No.2, pp.1-18, 2015.
[2] A. Aizawa, “An Information-theoretic Perspective of tf-idf Measures”, Information Processing & Management, Vol. 39, No.1, pp.45-65, 2003.
[3] D. S. Park, H. J. Kim, "Proposal of Join Vector for Semantic Factor Reflection in TF-IDF Based Keyword Extraction", Vol.16, No.2, pp.1-16, 2018.
[4] K. M. Park, K. B. Hwang, “Bio-Text Mining System Based on Natural Language Processing”, Journal of KIISE: Computing Practices and Letters, Vol.17, No.4, pp.205-213, 2011.
[5] J. W. Baek, J. S. Kang, K. Chung, "Multimodal Media Content Classification using Keyword Weighting for Recommendation", The Society of Convergence for Small and Medium Business, Vol.9, No.5, pp.1-6, 2019.
[6] D. W. Ko, J. J. Yang, "Korean Natural Language Processing and Analysis Using KoNLPy and Word2Vec", The Korean Institute of Information Scientists and Engineers, pp.2140-2142, 2018.
[7] J. C. Kim, K. Chung, "Associative Feature Information Extraction using Text Mining from Health Big Data", Wireless Personal Communications, Vol.105, No.2, pp.691-707, 2019.
[8] J. Y. Hyun, S. Y. Ryu, S. Y. Lee, "How to improve the accuracy of recommendation systems: Combining ratings and review texts sentiment scores", Korea Intelligent Information Systems Society, Vol.25, No.1, pp.219-239, 2019.
[9] J. C. Kim, K. Chung, "Mining Health-Risk Factors using PHR Similarity in a Hybrid P2P Network", Peer-to-Peer Networking and Applications, Vol.11, No.6, pp.1278-1287, 2018.
[10] Y. Kim, "A Study on Design and Implementation of Personalized Information Recommendation System based on Apriori Algorithm", The Korean Biblia Society For Library And Information Science, Vol.23, No.4, pp.283-308, 2012.

도커시스템에서 사용 가능한 모니터링 도구 비교 분석

정채은 ○ 박용범
단국대학교 소프트웨어학과
chaeeunjeong@dankook.ac.kr, ybpark@dankook.ac.kr

Comparative analysis of the monitoring tools available on the Docker system

Chae-eun Jeong ○ Young B. Park
Dankook University, Dept of Software Engineering

요 약

클라우드 컴퓨팅의 가장 큰 특징인 가상화 기술을 바탕으로 클러스터링 환경에서 서비스를 컨테이너로 배포하고 할당시킬 수 있다. 이러한 클러스터링의 규모가 커질수록 서비스의 감시와 관찰이 더욱 필요하다. 본 논문에서는 도커를 활용하여 데이터의 수집, 분석, 저장, 전달 그리고 시각화를 통해 모니터링을 하는 도구들에 대해 알아보고 특성을 분석하여 모니터링을 하는 도구를 선택하는 데에 도움을 주고자 한다. 실험을 통해 시간의 흐름에 따라 각 도구들의 하드웨어 자원의 변화를 비교 분석하였고 각 도구들의 CPU와 메모리의 사용량을 통해 하드웨어 자원의 사용량이 많은 도구를 알아볼 수 있었다. 실험 결과 12시간 후 CPU 최대 사용량은 cAdvisor, elasticSearch, kibana가 각각 0.97%, 0.8%, 0.44%로 다른 도구들에 비해 비교적 많은 CPU를 사용한다는 것을 알 수 있었다. 메모리의 사용량은 시간이 지나도 급격한 변화는 없었지만 초반 메모리 사용량은 elasticSearch가 61.36%으로 다른 도구들에 비해 압도적으로 많은 메모리를 사용하는 것을 알 수 있었고, 다른 도구들은 적은 메모리 사용량을 비슷하게 유지하는 것을 확인할 수 있었다.

1. 서 론

클라우드 컴퓨팅은 대규모의 분산된 네트워크 환경에서 컴퓨터 자원이나 서비스에 온라인으로 접근할 수 있는 기술이다. 네트워크를 통해 클라이언트가 필요한 리소스를 요청하여 제공받을 수 있다. 구조적으로 클라우드 컴퓨팅 환경은 프론트엔드와 백엔드로 나뉘어 있다. 프론트엔드를 통해서 클라이언트가 접근할 수 있고, 백엔드에는 컴퓨터 서비스가 클라우드 구조를 이루고 있다. 분산된 환경에서 리소스의 활용도를 높이기 위한 클라우드 컴퓨팅은 가상화, 확장성 그리고 상호운용성 등의 특징을 가지고 있다[1]. 특히, 가상화 기술은 클라우드 컴퓨팅을 서비스로 만들어 배포할 수 있는 핵심 기술이다. 가상머신을 이용한 RedHat KVM, VmWare ESX 그리고 컨테이너 기술을 이용한 오픈 소스 프로젝트인 도커 등을 이루는 가상화 기술들은 클라우드 컴퓨팅에서 주요하게 사용되어 왔다[2].

가상머신은 물리적 서버 하나를 통해 여러 개의 가상화된 운영체제를 제공하는 가상화 기술 중 하나이다. 물리적 서버 위에서 다른 운영체제를 실행하는 가상머신은 하드웨어를 할당하고 관리하여 하드웨어를 가상화 시키는 역할을 하는 것에 중점을 두고 있다[3]. 반면에, 각 가상머신은 하나의 서버를 운영하므로 관리해야 할 서버가 많다. 또한 가상머신에서 제공된 서비스는 가상머신이 아닌 서버에서 제공되는 서비스에 비해 성능이 떨어진다는 단점도 있다.

도커는 가상머신보다 가벼운 개념의 컨테이너 기술을 사용한다. 컨테이너들은 하나의 운영체제를 공유하여 하드웨어 자원의 사용이 가상머신에 비해 적다. 도커는 이러한 컨테이너 기술에 기능을 추가하여 서비스를 컨테이너 형식으로 사용한다. 도커는 성능 손실 면에서도 가상머신보다 우수하며, 실행할 수 있는 환경에 제약이 거의 없기 때문에 이식성이 높다[4]. 이처럼 도커가 가상머신보다 더 월등하다고 판단하여 본 논문에서는 도커와 관련된 컨테이너 기술에 더 초점을 맞췄다.

컨테이너 기술이 가진 유용한 기술 중 하나는 서비스를 클러스터링 환경에서 제공할 때 관리가 가능하다는 것이다. 컨테이너 기술은 개별적으로 실행하는 것뿐만 아니라 분산 환경에서 서비스를

제공할 때 가장 유용하다[4]. 이러한 특징은 사용자에게 서비스를 배포할 때 컨테이너를 추상적인 형태로 제공하기 때문에 따로 네트워크가 필요하지 않다. 도커에서 제공하는 대표적인 클러스터링 도구로 도커 스웜이 있다. 하나의 호스트 노드에서 여러 노드로 작업을 할당하고 자원을 최적화시켜 시스템의 성능을 유지시킨다. 이로 인해 한 노드에서 문제가 발생했을 때 다른 노드로 작업을 할당시켜 시스템적으로 문제가 나타나지 않도록 협력하게 하여 문제를 해결한다. 도커 스웜과 같은 협력하는 시스템 내에서 사전에 관리하여 문제가 발생하기 전에 해결할 수 있도록 모니터링을 할 필요가 있다[5].

클라우드 컴퓨팅 환경은 인프라 구조와 관련 없이 서비스를 유연하게 확장할 수 있고 자원 관리 측면에서 비용도 낮출 수 있고 서비스에 집중할 수 있게 하여 품질을 향상시켜준다[6]. 하지만 집단화가 되고 규모가 커지면 인프라를 관리하기 어려워진다. 이러한 어려움들을 해결하기 위해 서비스를 관리할 필요가 있고 서비스에 문제가 생겼을 때를 대비할 필요가 있다.

클라우드 환경에서 분산된 자원들을 감시하고 관찰하는 것은 중요하다. 데이터의 사용량 측정과 관리, 성능에 대한 관리, 환경을 안전하게 보호할 수 있도록 하는 안정성 유지 등의 지속적인 모니터링이 필수적이다[1]. 모니터링을 통해 자원들의 상태를 분석하면 사용자의 환경에서 문제가 발생했을 때 빠르게 감지할 수 있고, 문제의 해결도 위험한 지경에 이르지 않았을 때, 안전하게 이룰 수 있다. 또한, 클라우드 환경에서 사용되는 사용자의 서버의 CPU, 메모리, 네트워크 정보와 같은 메트릭 데이터들을 모니터링 하여 관리를 하면 어플리케이션의 성능을 향상시킬 수도 있다[7].

모니터링을 하는 과정은 메트릭 데이터들을 수집하고, 수집한 데이터들을 분석한 뒤 저장하거나 외부 모니터링 시스템으로 전달한다. 더하여, 데이터들의 관리 및 관찰을 위해 사용자의 목적에 맞게 다양한 대시보드로 시각화를 하고 알림 기능을 해주는 과정을 포함한다. 과정마다 제공하는 서비스는 다양하다. 본 논문에서는 도커 서비스를 생성하여 컨테이너로 배포할 수 있는 오픈소스로 제공된 모니터링 도구들에 대해서 알아보고, 성능과 특징을 조사한다. 따라서 모니터링을 하기 위해 도구를 선택하는 사용자들에게 도움을 주고자 한다.

2. 본 론

2.1. 모니터링 서비스 도구

2.1.1. cAdvisor(v0.24.1)

cAdvisor는 container Advisor의 약자로, 컨테이너 형식만 지원하는 구글에서 제공하는 오픈 소스 기반의 도구이다. 실행중인 컨테이너에 대한 자원 사용량이나 성능을 측정하기 위한 데이터들을 수집하고 분석하여 처리하는 역할을 하며 외부 시스템으로 전달하는 관리 도구이다. 지원하는 스토리지 드라이버에는 Prometheus, influxDB, Elasticsearch 등이 있다[8].

2.1.2. Elasticsearch(v6.8.6)

Elasticsearch는 검색과 분석을 하는 분산형 오픈 소스 엔진으로, 데이터를 저장, 분석 및 검색하는 역할을 한다. 분산형이기 때문에 확장성이 좋다는 장점이 있고 속도도 빠르다. 검색하는 쿼리에 해당하는 모든 항목이 색인되어 빠르게 정보에 접근할 수 있다. 더불어 다양한 요인과 기능을 고려하여 사용자에게 적합하고 정확한 정보를 제공하는 도구이다[9]. 이 도구는 데이터를 수집하고 원하는 형식으로 필터링 하여 데이터를 전달하는 오픈소스 서버인 Logstash와 데이터를 관리할 수 있게 시각화해주는 오픈소스 시스템인 kibana(v6.8.6)와 같이 사용된다[10].

2.1.3. influxDB(v1.7.9)

influxDB는 시계열 데이터베이스(TSDB: Time-Series Database)의 한 종류로 클라우드 환경 혹은 어플리케이션의 데이터들을 저장하는 역할을 하여 실시간 모니터링을 가능하게 한다. 특히 나노 초(ns)까지의 정확성을 제공한다. influxDB는 모니터링을 하는 사용자 혹은 조직에 자동화된 기능을 통해 slack이나 paperduty 등을 통해 알림을 하는 서비스도 제공한다[11].

2.1.4. node-exporter(v0.18.1)

node-exporter는 Prometheus가 제공하는 서비스로 호스트 서버의 자원 사용량에 대해 수집하고 데이터를 전달하는 역할을 한다. node-exporter는 네트워크에 백엔드로 연결되어 있어서 수집된 정보를 Prometheus를 통해 나타낼 수 있다. 또한 이 도구는 호스트 서버에 접근해 관련 데이터를 수집하므로 도커 컨테이너로 배포를 하기엔 적절하지 않은 도구이다[12].

2.1.5. Prometheus(v2.15.1)

Prometheus는 CNCF(Cloud Native Computing Foundation)의 두번째 프로젝트로 모니터링 기능과 알림 기능을 제공하는 오픈 소스 도구이다. PromQL을 제공해서 사용자들이 실시간으로 데이터들을 선별하고 수집하는 역할을 한다. 수집된 데이터들을 Prometheus의 웹 브라우저에 그래프로 나타낼 수 있고 시각화를 도와주는 다른 외부 시스템과 연결하여 모니터링에 이용할 수도 있다. 다른 저장소에 의존하지 않아도 되어 자율적

이고 문제 발생시 빠르게 감지할 수 있는 신뢰성 기반의 도구이다. 하지만 수집된 데이터들이 100% 정확하다고 할 수는 없어 100%의 정확도가 필요한 모니터링에는 사용하지 않는 것이 좋다[13].

2.1.6. Grafana(v6.5.2)

Grafana는 데이터들을 다양한 그래프로 시각화하여 모니터링하고 관찰하는데 사용되는 오픈 소스 플랫폼이다. 모니터링 과정에서 시각화를 해주는 역할을 하므로 데이터의 저장과 수집에 관해서는 별도의 도구가 필요하다. 모니터링을 하고자 하는 사용자가 위험하다고 판단하는 지점을 설정하면 Grafana가 감시하여 그 지점을 넘어가면 slack이나 paperduty 등을 통해 알리는 기능을 제공한다. 수십 개의 데이터베이스를 지원하기 때문에 같은 대시보드에 여러 저장소의 데이터들을 나타내서 한눈에 모니터링할 수 있다. 또한, 모니터링 하고자 하는 데이터들을 선택해서 볼 수 있게 대시보드를 직접 생성할 수도 있고 다른 사용자가 만든 대시보드를 가져와서 사용할 수도 있다[14].

2.2. 모니터링 서비스 도구 비교 실험

2.2.1. 실험 환경

실험 환경은 Intel Core i5 프로세서와 4GB 1600MHz DDR3 RAM으로 구성된 mac OS에서 진행되었다.

2.2.2. 모니터링 서비스 도구의 성능 측정

도커를 사용해서 하드웨어 자원의 의존도를 낮추어서 성능을 높아지는데, 모니터링 도구가 하드웨어 자원을 많이 사용하는 것은 도커를 사용한다는 장점을 잃을 수 있다. 따라서, 컴퓨터 하드웨어 자원인 CPU와 메모리의 사용량을 비교하였다. 또한, 성능의 비교는 도커에서 제공하는 docker statistics 도구를 이용해서 시간은 1분, 5분, 15분, 60분, 3시간, 6시간 그리고 12시간이 지났을 때 각각 측정하였고 도구들이 자원을 얼마나 사용하는지를 확인할 수 있다. 이 실험에서는 CPU와 메모리의 사용량 만을 비교하였다.

2.2.3. 실험 결과

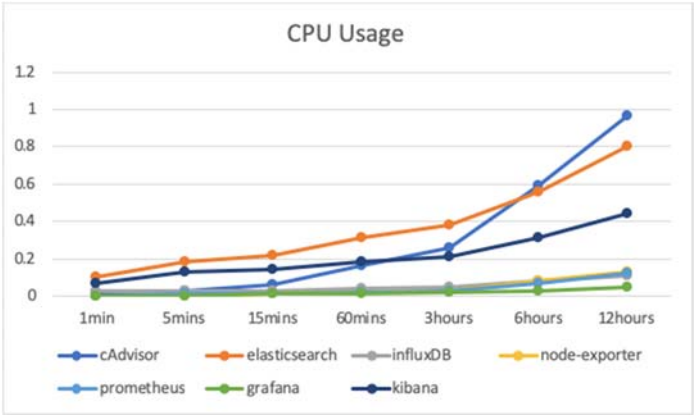


그림 1 시간의 흐름에 따른 CPU 사용량

그림 1 에서 node-exporter 와 Prometheus, influxDB 는 초반 한시간 동안 CPU 의 사용량이 0%에서 평균 0.03%로 미미한 정도가 상승되었다. 그 후 3 시간 마다 측정을 해도 큰 변화가 없었고 12 시간이 지난 후 세 개의 도구들은 평균 0.12%의 CPU 를 차지하는 정도밖에 되지 않았다. elasticSearch 는 처음 실행시켰을 때 0.1%로 제일 많은 CPU 를 차지하고 있었고, 12 시간 동안 0.8%까지 상승하여 다른 도구들에 비해 확연하게 CPU 를 사용하고 있는 것을 확인할 수 있었다. 반면에 cAdvisor 는 실행한 지 얼마되지 않았을 때는 0.06%정도의 CPU 를 차지했지만 30 분이 지나는 시점부터 한시간 마다 0.1%씩 상승해서 CPU 를 차지하여 그래프에서 눈에 띄게 급격한 상승을 보였다. 6 시간 후부터는 cAdvisor 가 모든 도구들 중에서 가장 많은 CPU 를 차지하고 있는 모습이 보였다. 시각화를 하는 역할의 grafana 와 kibana 를 비교해 보면 kibana 도 전체적인 그래프를 보았을 때 0.07%에서 0.44%까지 상승하여 비교적 많은 CPU 를 차지하고 있음을 알 수 있었다. 반면에 grafana 는 초반 한시간 동안 0.01% 만큼을 차지하다가 12 시간 후에는 0.05%를 차지하여 CPU 의 사용량이 상당히 적다는 것을 알 수 있었다.

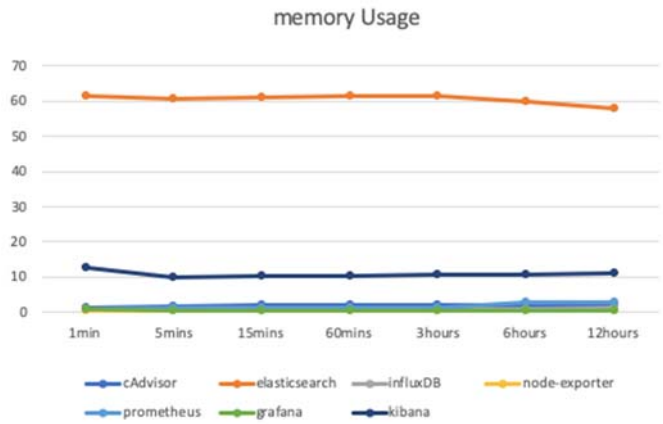


그림 2 시간의 흐름에 따른 메모리 사용량

그림 2 에서는 elasticSearch 가 처음 실행시켰을 때부터 약 1.19GB 의 메모리를 차지하여 전체 메모리의 61.36%로 가장 많은 메모리를 사용하고 있다가 3 시간이 지난 후부터 조금씩 감소하여 12 시간 후가 되었을 때는 58.06%의 메모리를 차지하고 있었다. Prometheus 와 cAdvisor 는 1 분 후부터 12 시간이 흐른 시점까지 각각 1.05%~2.92%와 1.15%~2.14%로 메모리를 차지했고 influxDB 는 0.91%을 계속 차지하다가 6 시간이 흐른 뒤 1.06%로 상승하고 12 시간 후에 1.24%만큼의 메모리를 차지했다. node-exporter 는 실행 초반에 0.33%의 메모리를 차지하다가 12 시간 후에도 0.46%의 메모리를 차지하여 모든 도구들 중에서 제일 적은 메모리 사용량을 보였다. 시각화 도구인 grafana 와 kibana 를 비교하면 grafana 는 컨테이너로 실행시켰을 때 0.79%의 메모리 사용량에서 시작하여 점차 감소하다가 12 시간 후에는 0.48%의 메모리 사용량을 보인 반면에 kibana 는 grafana 보다는 높은 12.78%의 메모리 사용량을 차지하는 것을 시작으로 점차 감소하여 12 시간 후에는 10.99%의 메모리 사용량을 보여 grafana 에 비해 컨테이너로 실행 시 많은 메모리를 차지한다는 것을 알 수 있었다.

2.2.4. 실험 결과 분석

	최대 CPU 사용량	초반 메모리 사용량	기능
cAdvisor	0.97%	1.15%	수집, 분석, 전달
elasticsearch	0.8%	61.36%	검색, 분석
influxDB	0.11%	0.91%	저장
node-exporter	0.13%	0.33%	수집, 전달
prometheus	0.12%	1.05%	수집, 시각화, 알림
grafana	0.05%	0.79%	시각화, 알림
kibana	0.44%	12.78%	시각화

표 1 모니터링 도구의 성능과 기능 비교

표 1 을 통해 도커를 활용한 모니터링 도구들을 비교 분석하기 위해 가장 확연한 차이를 보인 최대 CPU 사용량과 초반 메모리 사용량 그리고 각 도구들의 기능을 정리하였다.

CPU 사용량은 초반에는 도구들간 확연한 차이가 없다가 시간이 흐를수록 눈에 띄게 변화하였다. 최대 CPU 사용량은 cAdvisor 가 0.97%로 가장 높았고 elasticSearch 와 kibana 가 각각 0.8%와 0.44%로 그 뒤를 이었다. 나머지 도구들은 그래프 상에서 눈에 띌 정도로 차이가 나지 않았고, 그 수치는 node-exporter 가 0.13%, Prometheus 가 0.12%, influxDB 가 0.11%, 그리고 grafana 가 0.05%의 CPU 를 차지하는 것을 알 수 있었다.

메모리 사용량은 시간이 흘러도 급격한 변화가 없이 초반의 메모리 사용량과 큰 차이가 없었다. 따라서 표를 통해 초반 메모리 사용량을 비교하여 도구들을 비교하였다. 그림 2 를 통해 알 수 있듯이 다른 도구들에 비해 확연하게 메모리 사용량이 컸던 도구는 elasticSearch 로, 그것의 초반 메모리 사용량은 61.35%이다. 그 다음으로는 kibana 가 12.78%의 메모리 사용량을 보여주었다. 나머지 도구들은 그래프 상에서 구분이 어려울 정도로 비슷한 수치를 보여주었다. cAdvisor 가 1.15%, Prometheus 가 1.05%, influxDB 가 0.91%, grafana 가 0.79%, 그리고 node-exporter 는 0.33%로 모두 메모리 사용량이 비교적 낮았다.

3. 결 론

실험을 통하여 elasticSearch 와 kibana 는 모니터링 할 때 주로 같이 쓰이는데 컨테이너로 실행했을 때에는 CPU 와 메모리의 사용량이 다른 도구들 보다 확연히 많다는 것을 알 수 있었다. influxDB 와 node-exporter 그리고 Prometheus 는 CPU 와 메모리의 사용량이 전체적으로 낮아서 성능을 고려하여 모니터링 도구를 선택할 때 고려할 수 있을 것이다. 모니터링을 할 때 쓰이는 도구는 앞에 언급한 도구들 외에도 다양하고 많은데 실시간으로 모니터링 하는 도구가 필요할 때는 Prometheus 혹은 elasticSearch 를 선택하고, 정확한 정밀도가 제일 중요할 때는 InfluxDB 를 선택하는 등 자신이 필요한 도구 혹은 상황에 적합한 도구를 고려하여 모니터링 도구를 선택할 수 있을 것이다.

도커 스웜과 쿠버네티스와 같은 클라우드 기반의 클러스터링 환경에서도 모니터링을 하는 것이 당연시되었다. 본문에서 다룬 모니터링 툴을 오케스트레이션 도구인 도커 스웜이나 쿠버네티스에서 서비스를 배포하고 감시하기 위해 컨테이너를 배포하여 모니터링할 수 있다. 클러스터링 환경에서 이러한 모니터링 환경을 구성하면, 다른 호스트에서 문제를 일으키려는 움직임이 발생되면 감지가 되어 발생할 문제를 사전에 차단할 수 있다는 기대를 할 수 있을 것이다. 또는 이미지를 배포 받은 다른 컨테이너에서 문제가 발생했을 때 호스트에게 미리 알려서 대응할 방법을 마련해 놓을 수 있을 것이다. 이러한 방식으로 모니터링을 시켜 자동화가 되면 클러스터에 서비스를 배포하고 사람이 직접 관리하지 않아도 운영되는 자율적 시스템의 발전도 활발해질 것이다.

4. 참고문헌

[1] Jadeja, Yashpalsinh, and Kirit Modi. "Cloud computing-concepts, architecture and challenges." *2012 International Conference on*

Computing, Electronics and Electrical Technologies (ICCEET). IEEE, 2012.
[2] Liu, Di, and Libin Zhao. "The research and implementation of cloud computing platform based on docker." *2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing (ICCWAMTIP)*. IEEE, 2014.
[3] Pahl, Claus. "Containerization and the paas cloud." *IEEE Cloud Computing* 2.3 (2015): 24-31.
[4] Bernstein, David. "Containers and cloud: From lxc to docker to kubernetes." *IEEE Cloud Computing* 1.3 (2014): 81-84.
[5] Naik, Nitin. "Building a virtual system of systems using Docker Swarm in multiple clouds." *2016 IEEE International Symposium on Systems Engineering (ISSE)*. IEEE, 2016.
[6] Aceto, Giuseppe, et al. "Cloud monitoring: A survey." *Computer Networks* 57.9 (2013): 2093-2115.
[7] Observing the clouds: a survey and taxonomy of cloud monitoring
[8] cadvisor[online] https://github.com/google/cadvisor
[9] elasitsearch[online] https://www.elastic.co/kr/what-is/elasticsearch
[10] https://www.elastic.co/kr/products/kibana
[11] influxdb[online] https://www.influxdata.com/customers/infrastructure-and-application-monitoring/
[12] node exporter[online] https://github.com/prometheus/node_exporter
[13] Prometheus [online] https://prometheus.io/docs/introduction/overview/
[14] grafana [online] https://grafana.com/grafana/

미들웨어 기반 빅데이터 시각화 시스템 아키텍처 설계

백재형 ○, 송진범, 서상원, 남재창
한동대학교 전산전자공학부
{21500850, 21500362, 21300362, jcnam}@handong.edu

A Big Data System Architecture for Visualization using Middleware

Jaehyung Baek○, Jinbeom Song, Sangwon Seo, Jaechang Nam
School Of Computer Science And Electrical Engineering, Handong Global University

요 약

최근 여러 방면에서 빅데이터 분석이 활용되고 있다. 그리고 빅데이터는 데이터의 양이 많고 종류가 다양하며 새로운 데이터가 빠르게 발생한다. 그래서 새로운 종류의 데이터가 발생하면 데이터의 특성에 맞는 데이터 분석 알고리즘과 시각화 그래프가 적용되어야 한다. 따라서 본 논문에서는 다양한 데이터를 분석하기 위한 분석 알고리즘과 시각화 그래프를 구현하는데 있어, 확장이 용이한 데이터 시각화 기능을 지원하는 빅데이터 분석 아키텍처를 제안하고자 한다. 따라서 사용자와 상호작용을 가지는 프론트엔드와 데이터를 저장하는 백엔드 사이에 빅데이터 분석을 담당하는 미들웨어를 추가하여 아키텍처를 설계 및 구현하였다.

1. 서 론

근래 들어 다양한 분야에서 빅데이터 분석이 많이 활용되고 있다. 우리가 생활하면서 접하는 모든 것들이 데이터로 저장된다. 이러한 데이터들은 규모를 가늠할 수 없을 정도로 수없이 다양한 형태로 존재하며, 데이터의 생성 주기는 매우 짧다. 이렇게 모이는 거대한 데이터를 우리는 빅데이터라고 한다. 이러한 빅데이터 분야는 다양한 분야의 대규모 데이터를 생성, 수집, 분석 그리고 표현 과정을 통해 실시간으로 빠르게 변화하는 현대 사회를 더욱 정확하게 예측하여 효율적으로 움직이게 할 수 있다[1]. 한 일화로, 미국의 타깃(Target)사는 한 여성의 구매 형태나 검색 기록을 분석하여, 가족들보다 먼저 여성의 임신을 추측했고, 유아용품 할인쿠폰을 보내어 화제가 되었다[2]. 아마존은 CEF(Collaborative Filtering Engine) 이라는 기술을 통해 소비자의 선택을 분석한다. 소비자가 구매한 내역, 위시리스트, 제품에 대한 평점 등의 데이터를 수집하여 저장하고, 데이터 분석을 통해 물건을 검색할 때 최적화된 제품을 도출하여 보여주거나 제품을 미리 보관해 두어 소비자가 구입할 때 빠르게 배송 한다[3]. 그리고 넷플릭스는 지역별로 사용자들의 시청 기록 데이터를 이용하여 인기 있는 영상 콘텐츠를 분석하고, 지역의 캐시 서버에 미리 그 영상을 복사해 두어 지연 없이 빠른 스트리밍 서비스가 가능하도록 빅데이터 분석을 활용한다[4]. 이렇게 매우 방대한 자료의 데이터들을 분석함으로써 기존에는 발견하지 못했던 새로운 관점과 가치를 창출해 낼 수도 있다.

하지만 빅데이터 분석 결과를 이용하여 서비스를 개선시키는 것 못지 않게 분석 결과의 시각화를 통한 새로운 정보의 생성과 전달 또한 매우 중요하다[5].

빅데이터 분석은 데이터의 특성에 맞게 효율적이고 빠르게 처리가 되어야 한다. 빅데이터는 데이터의 양(Volume)이 많고, 종류(Variety)가 다양하며, 수집되고 변화하는 속도(Velocity)가 빠르다[6]. 따라서 새로운 종류의 데이터가 발생하면 데이터의 성질에 맞게 분석이 되어야 한다. 다시 말해, 빅데이터 분석 시스템에서 새로 추가되는 데이터의 특성에 맞는 분석 알고리즘과 시각화 그래프의 추가 및 관리가 용이한 시스템 아키텍처가 요구된다.

따라서 본 논문에서는 확장이 용이한 데이터 시각화 지원 빅데이터 분석 아키텍처를 제안하고자 한다. 그리고 본 논문에서 제안하는 확장이 용이한 빅데이터 분석 아키텍처를 통일 관련 연구 수행을 위한 통일 빅데이터 시스템 설계 및 개발에 적용하였다.

확장이 용이한 빅데이터 분석 아키텍처를 만들기 위해 사용자와 상호작용을 가지는 프론트엔드와 데이터를 저장하는 백엔드 사이에 빅데이터 분석을 담당하는 미들웨어를 추가하여 아키텍처를 설계하였다. 다양한 종류의 데이터에 사용하는 새로운 분석 알고리즘이나 시각화 그래프를 추가하는데 있어서 다른 모듈이나 컴포넌트와의 의존성을 낮추고 관리 효율을 증가시키는 성과를 보일 수 있었다.

2. 배경 및 선행 연구

빅데이터 분석 도구는 많은 분야에 사용되고 있고, 사용하고자 하는 분야와 아키텍처의 목적에 맞는 프레임워크와 아키텍처를 사용한다. 정부에서는 공공기관에서 기관의 목적을 위해서 생성하고 취득한 데이터들을 민간에 개방하여 데이터 분석에 활용하고자 하고 있다. 공공개방 데이터를 활용하는 빅데이터 분석 포털을 구현하기 위한 아키텍처 제안에 따르면, 빅데이터

분석 플랫폼은 다음과 같은 구성요소를 포함해야 한다. 여러 공공기관에 분산되어 있는 데이터를 수집하는 기능을 가진 데이터 수집 영역, 수집된 데이터에서 보이지 않던 의미를 찾아내는 분석 계층 영역, 수집된 데이터를 안정적으로 저장하고 관리하는 데이터 저장 영역, 분석 결과를 사용자에게 그림 또는 그래프 형태로 보여주는 데이터 시각화 영역, 시스템의 장애 해결과 관리를 담당하는 관리 계층 영역으로 아키텍처를 구성할 수 있다[7].

그리고 빅데이터 아키텍처에서 데이터를 다룰 때, 적용되는 분야와 데이터를 생산하는 기관과 서비스에 따라 다양한 형식의 데이터 타입을 사용하게 된다. Excel, csv 뿐만 아니라 json 형식의 데이터를 사용하고 있다. 다양한 형식의 데이터를 분석을 위해 하나의 형식으로 통합해줘야 하는 과정이 요구되었고, 변환과정에서 많은 자원이 소요되는 문제가 발생했다. 이러한 문제점을 해결하기 위해서 XML 형식으로 모든 데이터를 변환하여 빅데이터 아키텍처를 구성하는 방법이 제안되었다. 해당 연구에서는 데이터의 수집 및 관리, 데이터베이스에 저장, 데이터 재가공, 데이터셋의 융합과 시각화 모듈로 구성된 아키텍처를 제안하였다. 데이터 수집 단계에서는 공공데이터를 제공하는 사이트의 URL 과 Key 을 이용하여 데이터를 수집한다. 그리고 수집된 데이터 셋을 XML 형식을 사용하여 종류에 맞게 재분류해 DB 로 저장한다. 이후 여러 데이터 셋을 결합하여 숨겨져 있는 새로운 정보를 만들어내고 시각화로 표현한다[8].

본 연구에서 설계한 아키텍처는 통일 연구자료 검색 엔진에 사용될 색인 데이터를 수집 후 분석에 적용하였다. 그리고 검색 엔진이라는 도메인에 맞게 일반적인 빅데이터 시스템 아키텍처에서 사용하는 수집, 저장, 분석, 시각화의 과정을 백엔드, 미들웨어, 프론트엔드의 구조로 설계하였다. 그리고 유연한 데이터 분석을 위해 XML 등의 데이터 모델보다 NoSQL 과 json 형식으로 데이터를 저장 및 활용한다. 또한, json 형태의 데이터를 다루는 과정에서 모듈화를 통해 데이터 분석 알고리즘과 시각화 구현의 의존성을 최소화하는 방식으로 확장이 용이한 아키텍처를 설계하였다.

3. 확장이 용이한 빅데이터 분석 아키텍처 설계

3.1 아키텍처 요구사항 분석

본 연구의 최종 목적은 상이한 데이터에 적용되는 알고리즘 구현과 시각화 구현의 의존성을 최소화하고 관리 효율을 올린 아키텍처 제안이다. 아키텍처를 처음 설계하고 구현할 때, 시각화 라이브러리 D3.js 를 사용하여 데이터 분석 알고리즘과 시각화 다이어그램을 함께 구현하였다. 이는 시각화를 담당하는 프론트엔드 Angular 와 d3.js 가 Javascript 를 통해 호환되는 점을 이점으로 사용하고자 했기 때문이다. 그러나 시각화 그림과 데이터 분석 알고리즘에서 의존성 문제가 발생하였다. 하나의 데이터 분석 알고리즘이 여러가지 시각화 다이어그램에 동일하게 적용되는 경우나 다른 종류의 데이터가 동일한 알고리즘에 적용될 경우에서 코드의 반복이 지속되어 유지 보수에 문제가 되었다. 그리고 간단한 데이터 분석 알고리즘이 다른 복잡한 분석 알고리즘의 부분으로 활용될 때에도 의존성 문제가 발생하였다. 따라서 알고리즘 구현과 시각화 구현의 의존성 최소화하고 관리 효율을 증가시키려는 목적을 수행하기 위해서 기존의 백엔드-프론트엔드 구조의 아키텍처에서 미들웨어를 추가한 아키텍처를 구성하였다. 그래서 데이터 수집과 저장은 백엔드, 데이터 분석은 미들웨어, 그리고 데이터 시각화는 프론트엔드로 할당하였다. 그림 1 은 아키텍처를 보여준다.

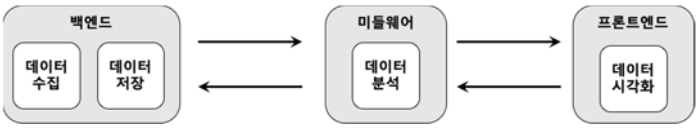


그림 1. 분석 알고리즘과 시각화 구현의 의존성을 최소화하기 위한 시스템 아키텍처

3.2 백엔드

백엔드(Backend)는 데이터를 수집하는 크롤러(crawler)의 역할과 크롤링된 데이터를 저장하는 데이터베이스의 역할을 담당한다.

3.2.1 데이터 수집

웹 어플리케이션 기반 빅데이터 아키텍처의 백엔드의 데이터 수집은 데이터를 생성하는 다른 장치나 서비스, 웹으로 부터 데이터를 수집해오고 수집한 데이터들을 누적시켜 대용량 데이터들을 생성하고 증강시키는 단계이다. 본 시스템에서는 크롤러로 python 기반의 scrapy 를 사용하였다. scrapy 는 프레임워크의 형태로 제공되기 때문에 불필요한 코드를 최소화할 수 있다. 그리고 비동기적 요청으로 작동하여 하나의 요청에 대한 에러가 다른 요청의 스케줄에 영향을 끼치지 않기 때문에 다른 에러로 인한 처리 지연이 일어나지 않아 속도가 빠르다는 특징이 있다[9]. 본 시스템에서는 검색엔진에 사용될 통일 연구 자료들의 색인(indexing) 데이터를 수집하였다. 만약 데이터를 추가하고 싶다면 크롤링할 사이트의 목록에서 새로운 사이트의 주소만 추가하면 된다.

3.2.2 데이터 저장

데이터들을 수집하여 대용량 데이터가 생성되면 데이터 저장 영역에 저장된다. 백엔드는 사용자의 요청을 받고 전달해주는 웹 어플리케이션 요소인 프론트엔드에서 사용자가 요구한 정보의 쿼리가 백엔드로 전달되면, 해당 정보를 탐색하여 그 결과를 다시 프론트엔드로 반환해주는 역할을 담당한다. 본 시스템의 백엔드 프레임워크로는 오픈소스 검색 엔진인 Elasticsearch 를 사용하였다.

Elasticsearch 는 Apache Lucene 을 기반으로 비정형 데이터 기반 검색이 가능하고, 동시에 자체적인 비정형 데이터베이스를 포함하고 있다. Elasticsearch 를 사용하면 클러스터 분산을 통해 데이터를 저장하기 때문에 데이터의 형태를 변경하거나 indexing 을 변경하는 경우에 유용하다. 또한 다양한 검색 옵션을 가지고 있어서 날짜별 또는 저자별 등 특수한 경우에 대한 검색을 빠르게 수행할

수 있다. 따라서 방대한 양의 새로운 데이터를 추가로 저장하고 처리하는 빅데이터 시스템에 유용하다. 또한 Elasticsearch 는 json 형식의 결과를 HTTP 통신을 통해 다른 프레임워크에 전달한다. 따라서 아키텍처 내부에서 동일한 형식의 데이터를 사용하여 유연한 적용이 가능하다[10].

3.3 미들웨어

미들웨어(Middleware)는 프론트엔드와 백엔드 사이에서 저장된 데이터를 호출하여 데이터 분석하는 역할을 담당한다. 이러한 분석을 통해 기존에 알려져 있지 않은 새로운 의미들을 파악할 수 있다. 빅데이터의 특성으로 인해 매우 다양한 종류의 데이터가 존재하기 때문에 상이한 분석 알고리즘을 적용해야 하며, 비슷한 종류의 데이터 분석은 동일한 알고리즘을 재사용해야한다. 따라서 미들웨어는 데이터 분석 제공하는 알고리즘에서 확장성을 가져야 한다. 특히 사용하려는 알고리즘을 다른 알고리즘을 변경시키는 경우와 알고리즘의 결과를 다른 알고리즘에 사용하는 경우, 알고리즘의 재사용성이 요구된다.

본 시스템에서는 미들웨어의 프레임워크로 Python 기반의 웹 프레임워크 Flask 를 사용했다[11]. Flask 는 개발자에게 높은 자유도를 제공하기 때문에, 수행하고 있는 프로젝트에 걸맞은 유연한 최적화가 가능하다. 또 Python 을 기반으로 하고 있기 때문에 간단하고 직관적인 코딩이 가능하고, 파이썬의 다양한 데이터 분석 라이브러리도 자유롭게 사용할 수 있다는 장점이 있다[11].

Flask 를 적용하여 데이터 분석 알고리즘을 미들웨어로 분리하고, 데이터 분석 알고리즘마다 고유한 indexing 과 함께 모듈화를 적용하였다. 그래서 Angular 에서 새로운 시각화를 추가하거나 기존의 시각화 모듈을 변경하는 경우, 필요한 알고리즘의 index 만을 호출하는 인터페이스를 통해 분석 알고리즘과 시각화의 의존성을 최소화하였다. 이처럼 모듈화를 완료한 미들웨어에서 연산의 확장 및 수정을 거치게 되면 프론트엔드와 백엔드의 확장 및 수정을 최소화할 수 있는 확장성을 얻을 수 있다.

3.4 프론트엔드

프론트엔드(Frontend)는 웹 어플리케이션에서 사용자와 직접적으로 상호작용을 하는 요소로서, 사용자에게 보여지고 사용자로부터 입력을 받는 모든 기능을 담당한다. 그리고 데이터 분석 결과를 시각화 그림이나 그래프 등을 통해 직관적으로 표현하여 사용자에게 보여줄 수 있고 새로운 의미를 도출할 수 있다. 본 논문에서 설계한 시스템에서 사용자는 빅데이터 분석을 원하는 키워드들을 프론트엔드에 입력하고, 그 키워드에 대한 빅데이터 분석 결과를 프론트엔드를 통해서 확인할 수 있다. 사용자가 입력한 모든 정보들의 연산과 수행은 프론트엔드에서 진행되지 않고 백엔드 혹은 미들웨어로 전송되어 수행된다. 프론트엔드 프레임워크로는 구글에서 개발한 Typescript 기반의 오픈소스 웹 프레임워크 Angular 를 사용하였다. Angular 는 SPA(Single Page Application)의 형식을 사용하여 다른 페이지로의 전환 속도가 빠르고 각 페이지를 Component 기반으로 구현하기 때문에 각각의 시각화 다이어그램을 모듈화 하기 용이하다[12]. 그리고 데이터 분석 결과를 시각화 그래프로 사용자에게 나타내기 위해 Javascript 기반의 라이브러리 D3.js 을 Angular 에 적용하였다. D3.js 는 웹 브라우저 상에서 동적이고 인터렉티브한 정보의 시각화를 위한 라이브러리이고, SVG 와 HTML5, CSS 등 웹 표준을 기반으로 구현되어 수십 개의 다양한 템플릿이 존재하며, 범용성이 높다는 장점이 있다[13]. 따라서 프론트엔드는 Angular 와 d3.js 을 사용하여 GUI 기반에서 시각화를 구현한다. 그리고 시각화에 사용될 데이터에 의존성을 가지지 않고 확장성 향상을 도모할 수 있다.

3.5 시스템 메커니즘

본 시스템의 메커니즘은 그림 2 에 나와있다.

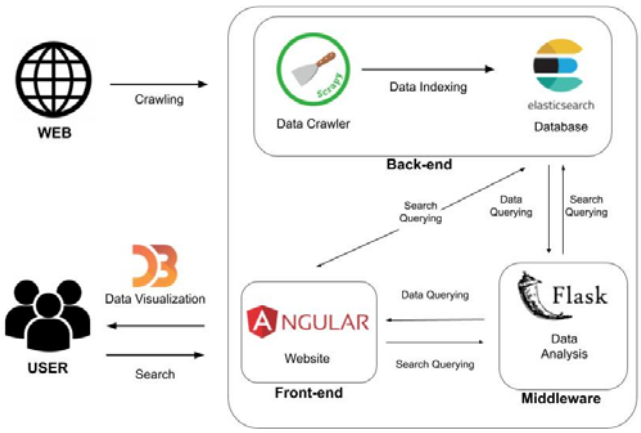


그림 2. 자료 검색 엔진에 사용되는 미들웨어 기반 빅데이터 분석 아키텍처

3.5.1 단순 데이터 호출

사용자가 단순하게 백엔드에 저장되어있는 데이터를 호출하는 경우, 빠르게 검색결과를 보여주는 것이 중요하기 때문에 이 경우에는 미들웨어를 거치지 않는다. 프론트엔드에서 사용자가 키워드를 입력하면, 해당 키워드의 쿼리를 바로 백엔드로 전송한다. 백엔드에서는 쿼리를 받으면 해당하는 데이터들을 프론트엔드로 다시 전송해준다. 데이터를 받은 프론트엔드는 사용자에게 데이터 호출 결과를 보여준다.

3.5.2 사용자의 요청에 따른 데이터 분석

사용자가 특정 데이터의 분석을 호출하였을 경우, 각 키워드와 해당 분석에 대한 형식이 정해지고 그것을 미들웨어로 전송한다. 키워드와 형식을 받은 미들웨어는 키워드로 쿼리를 만들어 백엔드로 전송한다. 쿼리를 받은 백엔드는 해당 키워드를 가진 데이터들을 미들웨어로 전송한다. 데이터를 받은 미들웨어는 해당 형식에 맞는 분석 알고리즘을 실행한다. 분석 과정이 끝나면 해당 결과를 프론트엔드로 전송하고, 결과를 받은 프론트엔드는 시각화 그래프를 통해 사용자에게 데이터 분석 결과를 보여준다.

3.5.3 정기적인 데이터 분석 함수 호출

사용자의 요청이 있을 때만 분석을 하는 것이 아니라 일정 주기(일/주/월 등)에 맞춰서 주기적으로 데이터를 분석하여 제공하는 경우, 프론트엔드 서버에서 자동적으로 미들웨어에 분석을 요청하고 미들웨어는 해당 양식에 필요한 데이터를 백엔드에 요청한다. 데이터를 받은 미들웨어는 분석 결과를 프론트엔드로 전송한다. 그리고 결과를 받은 프론트엔드는 결과를 파일로 저장하여 사용자의 요청이 없더라도 다음 주기의 업데이트 전까지 지속적으로 데이터의 분석 결과가 보이도록 한다.

3.6 데이터 분석/시각화 확장성 구현

본 논문에서 제안한 미들웨어를 통한 확장이 용이한 아키텍처를 활용하여 빅데이터 기반 통일 연구자료 검색 엔진이라는 도메인에서 구현하였다. 데이터 분석에는 키워드 분석과 카테고리 분석을 사용하였다. 키워드 분석에는 개별 문서에서 중요 키워드를 추출하는 WordRank, 전체 문서 집단에서 개별 문서의 특징적인 주요 키워드를 추출하는 TF-IDF 알고리즘을 사용하였다. 키워드 분석을 시각화하기 위해서 워드클라우드 그래프를 사용하였다. 그리고 전체 문서 집단에서 개별 문서들을 카테고리 별로 Clustering 하는 기능에는 LDA(Latent Dirichlet Allocation) 알고리즘을 사용하였다. 카테고리 분석 결과에 사용한 시각화 그래프는 노드들을 분류해주는 Circle Packing 과 Sun burst 다이어그램을 사용하였다.

미들웨어가 없는 상태에서 본 시스템을 구현할 때, 프론트엔드에서 시각화와 데이터 분석 알고리즘의 의존성이 높았다. 그래서 새로운 그래프를 추가할 때는 동일한 알고리즘 코드의 반복이나 개별 시각화에 적용할 때마다 마다 일일이 수정을 해줘야 하는 불필요한 시간 자원이 소모되었다. 하지만 미들웨어를 추가하게 되면서 알고리즘과 시각화 그래프를 모듈화하여 인터페이스를 적용하였다. 결과적으로 그래프나 알고리즘의 추가 과정에 각각의 의존성이 줄어들어 빠르게 개발이 가능하여 시간 자원의 소모를 줄일 수 있었다. 이후 새로운 시각화 그래프와 알고리즘의 추가에 있어서도 확장성을 가지고 적용하는 것을 기대할 수 있다.

4. 결론 및 향후 연구 방향

본 논문에서는 확장성을 용이한 시각화 기반 빅데이터 분석 시스템 아키텍처를 설계하였다. 기존의 데이터 수집, 저장, 분석, 시각화의 과정을 프론트엔드, 미들웨어, 그리고 백엔드의 구조로 구현하였다.

결론적으로 제안된 아키텍처를 통해 새로운 데이터가 축적되는 빅데이터 분석 환경에서 미들웨어를 사용하여 모듈화함으로써 알고리즘과 시각화의 확장성을 증가시켰다.

본 연구에서 설계하고 구현한 아키텍처는 검색 엔진 사이트의 형태로 사용자가 원하는 키워드로 나온 데이터에 대해 분석과 시각화 서비스를 제공한다.

향후 연구에서는 사용자가 관심 있는 데이터와 적용하고 싶은 차트를 직접 선택하면, 그에 따라 데이터를 분석하여 시각화를 보여주는 대시보드의 형태로 사용자 맞춤형 시각화 기능을 구현하여 더욱 확장성이 있는 아키텍처를 설계할 수 있을 것이다.

※ 이 논문은 과학기술정보통신부의 소프트웨어중심대학 지원사업(2017-0-00130)과 2019 년도 통일부 통일교육원 재원으로 한동대학교 통일교육선도대학사업의 지원을 받아 수행된 과제임.

6. 참고문헌

[1] Manyika, J.,Chui, M., McKinsey Global Institute, Brown, B, Bughin, J., Dobbs, R., Roxburgh, C., Byers. A.H., “Big Data: The Next Frontier for Innovation, Competition, and Productivity,” NY:McKinsey.

[2] Kashmir, H. (Feb 16, 2012). “How Target Figured Out A Teen Girl Was Pregnant Before Her Father Did.” Retrieved Dec 25, 2019 from https://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/#571037916668

[3] Linden, G., Smith, B., York, J. “Amazon.com recommendations : item-to-item collaborative filtering,” IEEE Internet Computing, vol 7, no. 1 , pp. 76-80 Jan.-Feb. 2003.

[4] Jon Markman (Feb 25, 2019). “Netflix Harnesses Big Data To Profit From Your Tastes” Retrieved Dec 26 2019 from https://www.forbes.com/sites/jonmarkman/2019/02/25/netflix-harnesses-big-data-to-profit-from-your-tastes/#15bc84f566fd

[5] O, J. H., Kim, D. J., Kim, J. D., “빅데이터 시각화 방법 및 시각화 프로세스”, 한국멀티미디어학회지. vol. 18, no. 1, pp.24-31, 2014.

[6] Laney, D. “Controlling Data Volume, Velocity, and Variety.”, Application Delivery Strategies, Meta group, Feb 6. 2001.

[7] Bang, S. Y., Ha, H. D., Kim, C. J., “A Study on BigData-based Software Architecture Design for Utilizing Public Open Data,” The Journal of Korean Institute of Information Technology, vol. 13, no. 10, pp. 99-107, 2015.

[8] Back, B. H., Ha, I. K., “A Method for Selective Storing and Visualization of Public Big Data Using XML Structure,” Journal of the Korea Institute of Information and Communication Engineering, vol. 21, no. 12, pp. 2305- 2311, 2017.

[9] Scrapy, https://scrapy.org/

[10] Elasticsearch, https://www.elastic.co/kr

[11] Flask, https://flask.palletsprojects.com/en/1.1.x/

[12] Angular, https://angular.io/guide/architecture

[13] D3.js, https://github.com/d3/d3/wiki

AI 와 SE 를 활용한 수화 번역 프로젝트

이유렬, 김은진 O, 이혁진, 이선아
경상대학교 항공우주및소프트웨어공학전공
sl0773@naver.com, asd0755@naver.com, bay03117@naver.com, saleese@gnu.ac.kr

Sign Language Translation Project Based on AI and SE

Yuryeol Lee, Eunjin Kim O, Hyeokjin Lee, Seonah Lee
Department of Aerospace and Software Engineering, Gyeongsang National University

요 약

AI 기술은 현대인의 편리성을 위한 많은 분야에 적용되고 있고, 사회적 약자를 위한 AI 기술도 발전하고 있다. 대부분의 일반인은 수화를 잘 이해하지 못한다. 우리는 농아 장애인과 수화를 모르는 일반인 간의 의사소통 격차를 줄이기 위해 AI 기술을 이용해 수화번역 시스템을 개발하였다. 또한 수화 번역 프로젝트에서 AI 기술을 실제 활용하기 위해서 소프트웨어 공학 기법을 적용하였다. 그 결과, 데이터 수집, 전처리, 피쳐 추출과 같은 AI 측면의 고려 요소뿐만 아니라, 소프트웨어 모듈화와 IoT 에서의 성능 이슈 등을 추가로 찾아낼 수 있었다. 우리는 수화번역 AI 를 기술을 통해 농아 장애인과 일반인 간의 원활한 의사소통 연구 및 AI 프로젝트를 위한 소프트웨어 공학 기법 적용 연구에 공헌하기를 기대한다.

1. 서 론

최근 4 차 산업혁명 이후 소프트웨어 기술이 크게 발달하고 있다. 그 중 가장 눈에 띄는 기술은 AI 기술이다. 이 AI 기술은 현대인의 편리성을 증진시키기 위해 발달해오고 있으며, 대표적으로 알파고나 차량 자율주행 등 많은 분야에 적용되고 있다. AI 기술 중 일반적으로 잘 알려진 기술은 이미지 분류이다. 이미지 분류 기술을 적용하면 사람의 손 글씨를 인식하고 분류하는 기술이 있다. 나아가 이러한 분류 기술에서 이미지를 손 글씨가 아닌 사람의 수화로 적용하여 사람의 수화를 인식하여 변환시켜 주는 기술을 개발할 수 있다.

본 논문에서는 사회적 약자들을 위해 이러한 AI 기술을 적용시키는 방향에서, 수화 번역 프로젝트를 진행한다. 해당 프로젝트에서는 이미지 분류 기술을 이용하여 문자에 해당하는 수화를 번역하는 프로젝트를 수행한다. 본 논문의 프로젝트는 농아장애인과의 원활한 의사소통을 위해 수화를 번역하여 시각적 및 청각적 매개체로 전달하는 것이다. 농아장애인들에게 수화번역을 제공함으로써 수화를 알지 못하는 일반인과의 원활한 의사소통을 제공할 수 있으며, 높은 정확도를 만족할 시 수화 교육 자료로 이용 가능하다.

또한 실질적인 개발을 위하여 소프트웨어 공학 기법들을 활용한다. 시스템 모델링에 있어서는 소프트웨어 공학에서의 UML 에서의 유스케이스 다이어그램, 클래스 다이어그램, 시퀀스 다이어그램을 활용하였다[7]. 또한 프로젝트 진행에 있어서는 Agile 개발론의 반복 개발로 진행하였다[7].

본 논문은 2 절에서는 관련 연구, 3 절에서는 설계된 AI 모델을 4 절에서는 소프트웨어공학적 모델을 서술한다. 5 절에서는 시스템을 개발한 반복 개발과 테스트 결과를 설명하며 6 절에서는 논의, 7 절에서는 결론 및 향후 과제를 설명한다.

2. 관련 연구

수화는 단일 프레임의 이미지가 아닌 손의 동작으로 이루어진 언어이다. 즉 단순한 이미지가 아닌 이미지 여러 프레임이 합쳐진 영상이다. 허나 모든 수화가 이러한 동작으로 이루어 진 것이 아니다. 숫자 하나, 둘, 셋 등을 표현할 땐 단순히 손가락을 펼치는 정도만 수행한다. 이와 마찬가지로 어떠한 문자에 대해 일치하는 손 모양이 존재한다.

수화 번역의 관련 연구로 연구[1]에서 제시한 영어권 수화 이미지 분류 기법을 통해 수화 번역의 실현 가능성을 확인했다. 손동작 인식을 위해 연구[2]에서 손과 손 형상 패턴을 추출하는 기법을 적용하고, 손 추출을 위해 피부색과 boundary energy 정보를 이용한다. 또한 손 동작의 구분은 이미지 분류 Neural Network 로 이미지 분류가 가능한데, 이때 이미지 분류에 적합한 Network 는 Convolution Neural Network 이다.

본 논문에서는 CNN 설계 기법 등을 이용하여 AI 모델을 설계한다[3], [4]. 또한 카메라의 자세 정보와 2 차원의 손가락 위치 정보를 이용하여 선형 동작 패턴으로 복원해 일반 카메라로도 효율적으로 손동작을 인식 가능한 시스템을 이용한다[6].

3. AI 모델

본 논문에서는 사람의 수화를 인식하여 평가 후 사용자에게 해당되는 문자를 시각적, 청각적으로 출력하는 것이 요구된다. 사용자는 수화번역을 위해 임의의 장소에서 시스템을 이용할 것이다. 임의의 장소에서 수행될 시 사용자의 손 이외의 배경들은 시스템의 동작을 방해하는 노이즈 요소로 작용한다. 정확한 평가를 위해서는 배경을 추출하여 노이즈를 제거하는 기능이 요구된다.

배경을 추출하기 위한 방법에는 색공간을 활용하는 방법이 있다. 색공간 내에서 사람의 손과 유사한 색상 만을 추출하고 나머지 색상은 제외한다. 이 알고리즘을 수행하기 위해 RGB 색상 모델은 적합하지 않다. 사용자는 다양한 피부색을 가질 수 있으며, 임의의 공간 내의 조도가 일정하지 않기 때문이다. RGB 모델은 각각의 색상에 채도와 명도가 포함되어 있기 때문에 부적합하다. 따라서 본 논문에서는 YCrCb 색상모델을 활용하여 배경추출을 수행한다. 적합한 배경을 추출하기 위하여 색상모델의 색상범위를 Y 값은 20~240, Cr 값은 135~190, Cb 값은 73~158 사이 값으로 정의하였다[2].

학습에는 다수의 학습 데이터가 요구된다. 본 논문에서 학습데이터로 각 알파벳당 7500 장의 이미지를 사용하였다. 데이터를 수집하기 위해서 영상에서 프레임을 추출하는 방법을 이용하였다. 또한 사용자가 명암이 일정하지 않는 임의의 장소에서 시스템을 사용할 것을 고려하여 학습데이터 추출 시 카메라의 명암 조절 기능을 이용하여 명암을 조절하였다. 추출된 각 프레임은 배경추출 기법을 이용하여 학습데이터에 존재하는 배경을 추출한다.

사용자의 수화를 평가하기 위해 학습모델과 평가모델이 요구된다. 학습모델은 이미지 분류에 뛰어난 CNN(Convolution Neural Networks)를 기반으로 구축하였다. Tensorflow 에서는 다양한 Neural Network 의 학습의 정확도를 위한 다수의 Optimizer 를 제공하고 있다. 우리는 그 중 Mnist Database 를 통하여 Loss function 의 함수 값을 최소화 시키는데 최적화된 Adam optimizer 를 이용한다[5].

본 논문의 개발 프로세스에 따라 Prototype 을 우선하여 학습모델을 구축하며, 초기 설계된 Prototype 을 점차 개선하는 방향으로 학습모델을 구축하였다. 초기 설계된 Prototype 의 Layer 는 다음과 같이 구축되었다. 각 Layer 에는 데이터의 수를 늘리기 위하여 Convolution 연산이 연속적으로 2 번 수행된다. 허나 Convolution 연산을 이용하여 데이터의 수가 늘어나면 학습에 요구되는 시간 또한 늘어난다. Epoch 당 학습시간이 30 분을 넘지 않기 위하여 Maxpooling 을 이용하여 데이터의 사이즈를 줄인다[3].

그림 1 는 초기의 구현 가능성 확인 등을 위한 초기 모델이다. 본 모델을 이용하여 초기 10 개의 평가 Case 에 대한 학습을 진행하였다. 학습의 결과는 그림 2 와 같다. 이 결과에 따라 본 논문에서는 iteration 을 진행함에 따라 요구되는 수정사항을 이전 설계된 Prototype 에 적용하여 개발을 진행해 나간다.

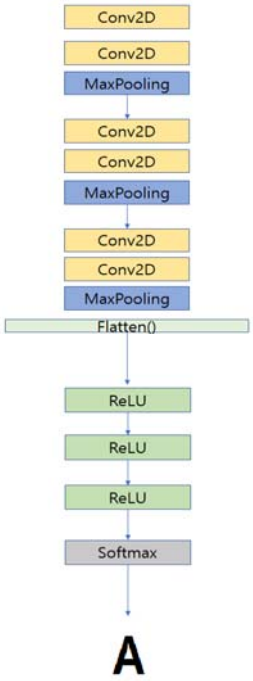


그림 1. Prototype 모델 Layer

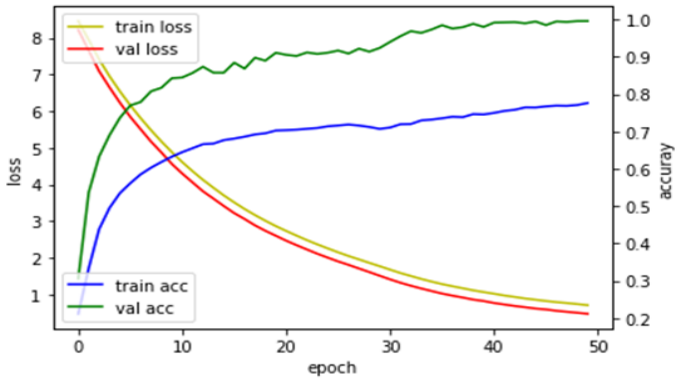


그림 2. 초기 학습 결과

평가 Case 가 늘어남에 따라 정확도가 감소해 학습 모델의 수정이 요구되고, 학습에 필요한 Data 가 늘어난다. 이런 대용량의 학습 Data 를 메모리에 전부 적재하는 것은 어렵다. 따라서 본 논문에서는 HDF5 파일을 이용해 학습을 위한 Pipeline 을 구축한다. HDF5 파일로 Pipeline 을 구축하면 다수의 학습 데이터를 하나의 파일로 관리가 가능해 편리하다. Pipeline 이용 시 GPU 가 학습하는 동안 CPU 가 batch size 만큼의 데이터를 load 하므로 학습에 소요되는 시간이 증가하지만, 대용량의 학습 data 를 메모리 크기에 관계없이 사용할 수 있으므로 overhead 가 납득 가능하다.

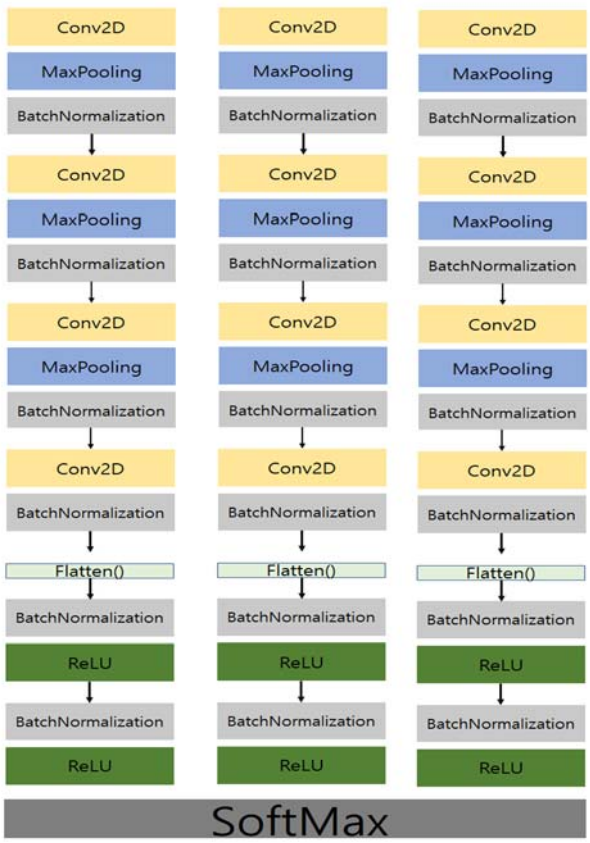


그림 3. 최종 모델 Layer

평가 Case 가 늘어나면서 기존의 학습모델로는 요구되는 정확성을 만족하지 못했다. 평가 Case 가 늘어나더라도 정확한 평가를 위해서 새로운 모델로 수정하는 것이 필요하다. 본 논문에서는 모델을 더욱 Wide 하게 설계하는 방향으로 진행

하였다. 그림 3 의 최종 모델은 초기 모델과 다르게 Convolution 연산을 Layer 마다 1 번씩만 수행하였다. 또한 Activation 에 input 을 대입하기 전 Batchnormalization 을 수행하여 internal covariance shift 문제를 해결하였고, 학습의 속도가 증가하였다. Activation 은 초기 모델과는 다르게 2 개로 줄였다. 또한 빠른 초기값을 위하여 Xavier Initializer 를 활용하였다. Loss function 에는 categorical hinge 를 이용하였다. Optimizer 간 성능 비교 분석[6] 을 통하여 본 논문에서는 optimizer 중 Adam optimizer 를 이용하여 학습을 진행하였다. 위와 같은 새로운 모델을 구축하여 26 가지의 평가 Case 에 대응 가능하다.

4. 소프트웨어공학적 모델

본 논문에서는 최초 반복 개발에서 사용자를 농아장애인으로 설정하여 표 1 과 같이 기능적, 품질적 요구사항을 추출했다. 이후 사용사례를 기반으로 시스템과 상호작용하는 외부 객체와 시스템의 기능을 표현하여 그림 4 의 유스케이스 다이어그램을 작성하였다. 사용자는 파이 카메라를 통한 수화 입력, 오탈자 수정, 입력된 수화의 평가 기능을 사용하여 시스템과 상호작용한다. 시스템이 수화 평가를 마치면 외부 객체인 파이 스피커와 파이 스크린이 평가 결과를 출력한다. 개발자는 모델을 개선하고 학습하는 작업을 수행한다.

표 1. 기능적, 품질적 요구사항

기능적 요구사항	품질적 요구사항
영상 인식	정확도 70%
수화 학습 및 평가	응답시간 1.5 초 이하
하드웨어를 통한 출력	모델의 파일화

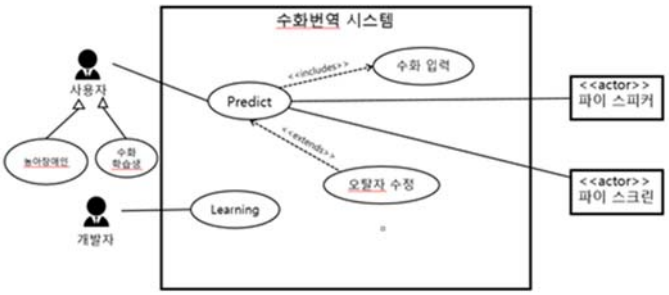


그림 4. 유스케이스 다이어그램

본 논문에서 구현을 시도한 프로토타이핑 모델의 H/W 구성을 위해 Jetson Nano 보드에 DeveloperSD Kit SD Card Image JP4.2.3, openCV 3.4, 오픈소스 배포가 용이하도록 Keras 2.0 를 설치했다. 초기 모델에서 Jetson nano GPU 메모리의 성능을 고려해 메인 컴퓨터에서 학습을 진행하고 저장된 모델을 Jetson nano 보드로 옮겨 평가를 진행했다. 학습과 평가 케이스가 늘어나면서 평가 응답시간이 길어져 응답시간 1.5 초 이내의 품질적 요구사항을 충족하지 못했다. 더 나은 시스템 성능이 요구되나 품질적 요구사항 중 응답시간의 향상과 휴대성 저하 방지를 모두 고려하여 메모리를 추가로 설치하지 않고 Socket 통신 방안을 채택하였다.

본 논문에서 Socket 통신 클래스를 적용한 시퀀스 다이어그램은 그림 5 와 같다. 시퀀스 다이어그램의 중앙 회색선을 기준으로 좌측은 개발자 영역, 우측은 사용자 영역이다. 개발자 영역의 경우, Making database 클래스에서 학습의 Pipeline 구축을 위해 수집한 학습 데이터를 HDF5 파일로 생성한다. Image load 후 HDF5 파일에 Write 하고 HDF5 파일을 저장한다. 동시에 Learning Model 클래스에서 모델을 구축하고 HDF5 파일을 load 한 후 학습을 진행한다. 시퀀스 다이어그램에서는 모델을

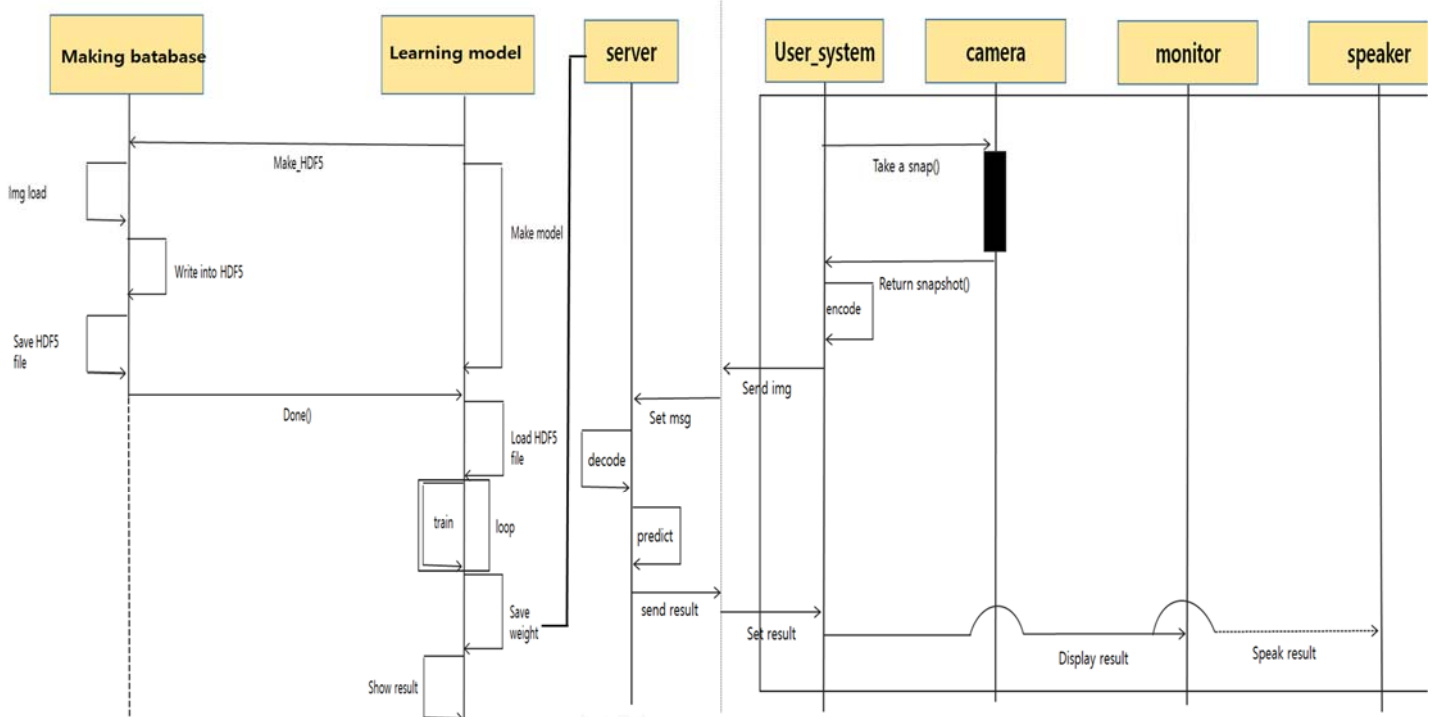


그림 5. 시퀀스 다이어그램

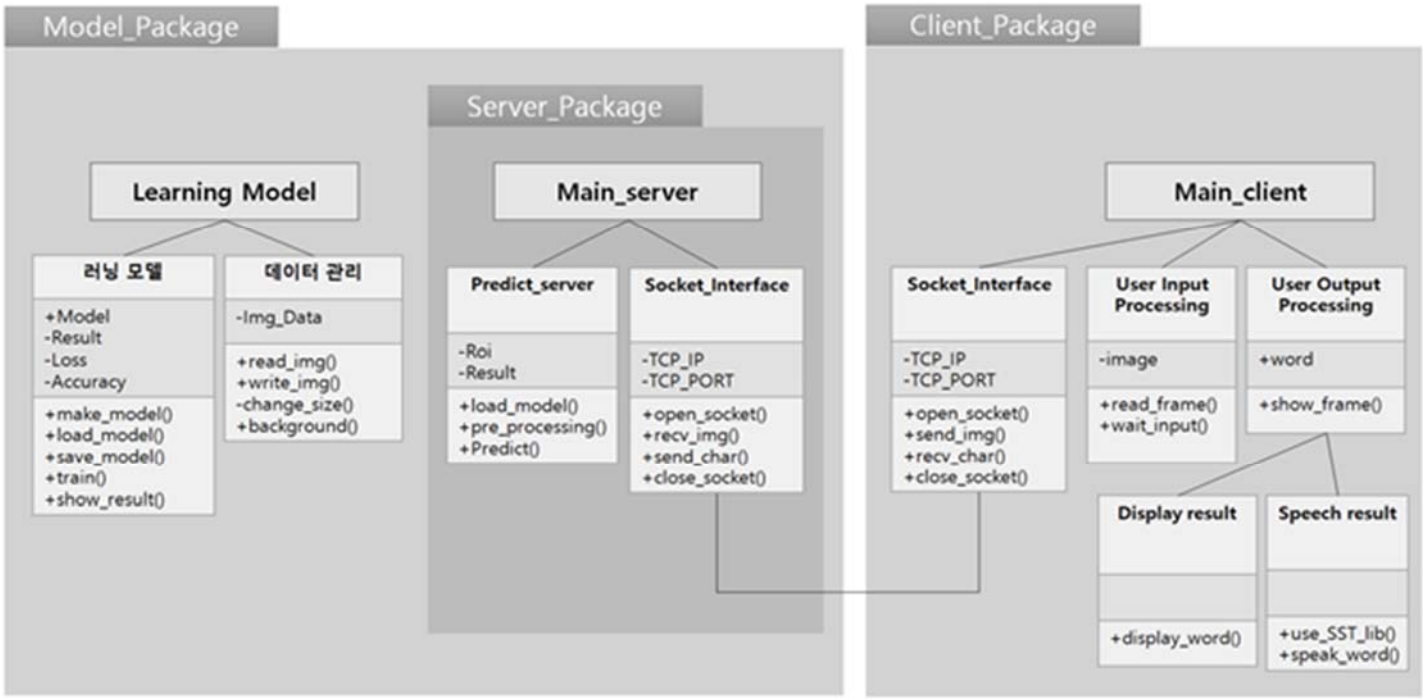


그림 6. 클래스 다이어그램

구축한 즉시 학습을 시도하는 것으로 표현되었다. 모델과 Weight 를 H5 파일로 저장하고, epoch 당 accuracy 와 loss Value 를 그래프로 나타낸 후 최종 평가 결과를 개발자에게 보여준다. 서버가 학습된 모델과 Weight 를 불러오고 통신이 연결될 때까지 기다린다.

사용자 영역의 경우, 사용자가 H/W 에 전원을 추가하면 User_system 클래스가 동작한다. User_system 클래스는 사용자가 카메라로 이미지를 보내는 즉시 서버에 이미지를 전송한다. 서버는 해당 이미지의 전처리와 평가를 수행하고, 평가 결과를 client 에 전송한다. 사용자는 monitor 를 통해 결과를 확인하며, 필요 시 speaker 로 결과를 음성으로 출력 가능하다.

본 논문에서 그림 4 의 유스케이스 다이어그램과 그림 5 의 시퀀스 다이어그램을 참조하여 그림 6 의 클래스 다이어그램을 작성했다. Model_Package 내부에는 학습 모델 구축과 학습을 위한 러닝 모델 클래스, 학습 데이터 전처리과 관리를 위한 데이터 관리 클래스, Server_Package 가 있다. Server_Package 에는 모델을 불러와 이미지를 평가하는 Predict_server 클래스, 통신을 위한 Socket_Interface 클래스가 있다. Client_Package 에는 통신을 위한 Socket_Interface 클래스, 이미지 입력을 위한 User Input Processing 클래스, 평가 결과 출력을 위한 User Output Processing 클래스가 있다.

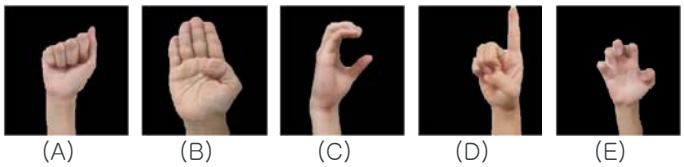
5. 반복 개발, 데모 및 테스트 수행 결과

5.1 반복 개발

본 논문은 Agile Process 를 기반으로 프로젝트를 수행하였다. 각 iteration 은 4 주를 기준으로 하여 7 월부터 11 월까지 총 5 회의 iteration 을 수행하였다. 최초 iteration 에서 요구사항을 추출 및 명세화 하고, 추출된 요구사항을 기반으로 하여 Product Backlog 를 추출하였다. 반영할 요구사항으로 구성된 기능을 찾아내고 기능들 간의 관계를 나타내기 위해 UML 모델링을 수행하였다. 사용자 관점에서 사용 사례와 기능적 요구사항을 정의하기 위하여 그림 1 의 Use-Case Diagram 을 작성하였다. 각 use-case 를 대상으로 작성된 Use-case 명세서를 이용하여 Sequence Diagram 과 Class Diagram 을 작성하였다. 다음 iteration 에서는 프로토타입 제작을 통한 기능적 구현을 수행하였으며 3,4 차 iteration 에서는 제작된 Prototype 과 Diagram 을 기반으로 AI 모델과 IoT 를 위한 H/W 모델을 구축하고 단위시험을 수행하였다. 마지막 5 차 iteration 에는 테스트 V 모델 절차에 따라 시스템, 인수 시험을 수행하였다[8].

5.2 데모

그림 7 는 알파벳 A 부터 E 까지 해당되는 수화이고, 그림 8 은 그림 7 의 수화 입력 시, A 부터 E 까지 정상 출력됨을 보여주는 데모 결과이다. 데모 영상은 https://youtu.be/RfailNpa5b4 에서 확인할 수 있다.



(A) (B) (C) (D) (E)

그림 7. 알파벳 A~E 에 해당하는 수화

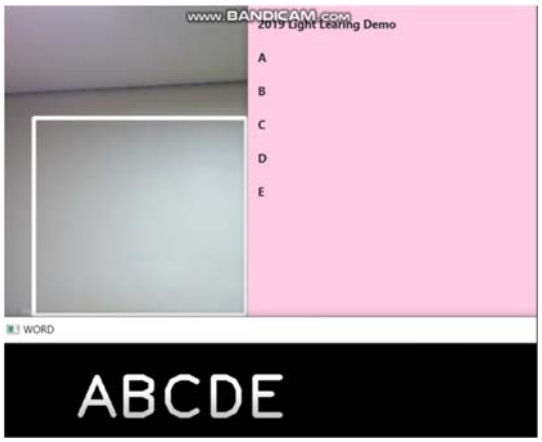


그림 8. 데모 결과

5.3 테스트 결과

본 논문에서 설계한 모델의 정확도와 응답시간을 평가하기 위해 테스트를 수행한다. 본 테스트는 원활한 통신을 위해 LAN 에서 수행되었으며, 원활한 배경 추출을 위하여 배경색이 피부색과 다른 실내공간에서 수행하였다. 또한 응답시간 측정을

위하여 python 의 time 모듈을 이용하였다. 이미지를 인식하고 평가한 후 출력까지 소모되는 시간을 측정하기 위해 CPU 시간을 측정한다.

정확도 테스트는 알파벳 A 부터 Z 까지 각 알파벳 마다 20 번을 수행한다. 카메라의 공간을 9 개로 분할하여 각 공간에 손을 위치시키며 테스트하였고, 동시에 조도와 각도를 바꾸어 가며 수행하였다. 정확도 테스트 결과는 표 2 와 같다. 예를 들어 표 2 의 첫번째 데이터 줄에 있는 결과를 보면, 알파벳 A 에 대하여 20 번의 시도 횟수에 있어서 16 번의 일치와 4 번의 불일치를 확인하였다. 이를 모든 알파벳에 대하여 수행한 결과 전체 평균 72.5%의 정확도를 가짐을 확인하였다.

응답시간 테스트는 총 15 회 실시하였다. 테스트 결과는 표 3 과 같다. 상기 표 2 의 모든 시도에서 1 초 미만의 응답시간을 가졌다. 평균적으로 약 0.6 초의 응답시간이 걸렸다. 최소 0.15 초가 걸렸으며, 최대 0.91 초가 소요되었다.

표 2. 정확도 테스트 결과

	시도 횟수	불일치 횟수	정확도		시도 횟수	불일치 횟수	정확도
A	20	4	0.8	N	20	5	0.75
B	20	4	0.8	O	20	8	0.6
C	20	0	1	P	20	8	0.6
D	20	7	0.65	Q	20	0	1
E	20	2	0.9	R	20	7	0.65
F	20	4	0.8	S	20	2	0.9
G	20	5	0.75	T	20	3	0.85
H	20	3	0.85	U	20	8	0.6
I	20	7	0.65	V	20	10	0.5
J	20	3	0.85	W	20	3	0.85
K	20	8	0.6	X	20	10	0.5
L	20	7	0.65	Y	20	10	0.5
M	20	6	0.7	Z	20	9	0.55
				Avg			0.725

표 3. 응답시간 테스트 결과

평균	0.636 sec
최솟값	0.15 sec
최댓값	0.91 sec
제 1 사분위수	0.545 sec
제 3 사분위수	0.78 sec
표준편차	0.2008 sec

6. 한계점과 향후 연구 논의

본 논문은 Agile Process 를 사용해 하나의 iteration 을 수행하고 회고하면서, 프로젝트를 수행할 때 발생한 문제를 해결하고 대안책을 고안하였다. 이러한 회고를 통하여 규명한 본 연구의 한계점과 향후 연구는 다음과 같다.

첫째, 배경 추출 시 색공간만을 활용하여 피부색과 유사한 색상을 지닌 배경에서는 한계점이 드러난다. 본 프로젝트에서의 전처리는 배경을 추출한 후, 이를 학습하여 러닝 모델과 평가모델을 구축하였다. 영상에서 프레임 추출을 통한 데이터 수집을 하고, 데이터를 학습시키기 위해 배경 추출을 위한 색공간 기술을 사용하였다. 색공간 기술의 경우 HSV 를 이용해서 많이 사용하지만, 본 연구에는 취지에 맞게 YCrCb 색상 모델을 사용하였다. 이 색공간 기술의 경우, 색상에 따른 배경을 추출한다는 점에서 급격한 변화가 있는 배경이 있더라도 추출이 잘 된다는 장점이 있지만 추출하고자 하는 배경과 물체가 같거나 비슷한 색인 경우 추출이 힘든 한계점이 있다. 추후 연구 개발시엔 경계선추출 기법을 추가적으로 적용해야 환경에 영향을 받지 않는 배경추출이 가능할 것으로 예상한다.

둘째, 본 논문에서는 하나의 알파벳 각각의 입력을 요구하였으나 실제 수화는 단어마다의 손 동작으로 이루어져 있다. 본 프로젝트에서는 이미지 분류를 이용하여 알파벳당 일치되는 수화를 번역하여 시각적, 청각적 매개체로 전달하는 시스템을 구현했다. 하지만 본 시스템으로 수화를 번역할 시 하나의 단어를 출력하기 위해서 알파벳과 일치하는 수화를 하나씩 입력해야 하는 어려움이 존재한다. 수화를 번역하는 관련 연구 개발 시에는 영상입력을 통한 단어 입력이 만족된다면 자연스러운 수화 번역이 수행 가능하다. 또한 일반적인 수화는 단순한 이미지가 아닌 손의 동작과 사람의 표정으로써 이루어진 여러 프레임이 합쳐진 영상이기 때문에, 이미지 분류 기법보다 더 추가적인 기법 활용이나 피쳐 추출이 필요하다.

셋째, 통합과정에서 개발자의 수월한 유지보수를 위해 모듈화가 필요하다. 본 프로젝트에서는 AI 모델, S/W 모델과 하드웨어 모델과의 결합을 위해 요구사항을 반영한 큰 그림을 볼 수 있는 유스케이스 다이어그램, 시간에 따른 모듈 간의 데이터 전송이나 함수 호출을 볼 수 있는 시퀀스 다이어그램, 모듈의 구성을 보여주는 클래스 다이어그램을 통해서, H/W 와 S/W 의 통합, 통신 및 구현을 하였다. 그러나 유지보수성을 높이는 부분이 충분하지 않았다고 회고한다. 추가적인 모듈화를 수행한다면 요구사항 또는 모델의 변경에도 유연한 대처가 가능하다.

넷째, 실제 활용을 위한 머신러닝과 IoT 의 결합을 위해선 H/W 의 성능개선이 필수적이다. 본 프로젝트에서 사용된 H/W 인 Jetson nano 의 메모리 자원은 4GB 를 가지지만 기본적으로 O.S 를 위해 사용되는 메모리 자원을 제외한다면 사용 가능한 메모리는 3GB 이다. 모델 적재 시에 최소 3GB 이상이 필요하므로 필연적으로 Swap memory 를 활용하여야 한다는 한계점이 존재한다. 추후 모델을 최적화하여 요구되는 메모리를 줄이거나, H/W 자체 성능을 개선하는 방안을 고려하여야 한다.

마지막으로, 추후 AI 모델 설계 시 Prototype 은 최종 출력값을 모두 고려하여 설계할 필요가 있으며, 학습 데이터는 모델의 품질에 영향을 끼치기 때문에 높은 품질의 학습데이터를 빠르게 추출하는 방안이 필요하다. 그 방법에는 손의 각도와 조도를 달리하여 영상을 촬영한 후 프레임을 추출하는 기법이 있다.

7. 결 론

본 논문은 농아장애인들을 위한 AI 기술을 이용하여 수화 번역 프로젝트를 수행하였다. 프로젝트를 위한 소프트웨어 공학 기법으로는 UML 모델링과 Agile 개발론을 참조하여 수행하였다. 학습모델은 추후 오픈소스로 배포 시 머신 러닝 학습자 또한 쉽게 이용이 가능하도록 keras API 를 활용하였다. S/W 모델은 이미지 분류에 뛰어난 CNN 을 기반으로 구축하였으며 26 가지의 평가 Case 에 높은 정확도를 가지기 위하여 Wide 하게 구축하였다. H/W 모델은 빠른 응답시간을 위해 Jetson Nano 보드를 활용하였으나, 3 초 이상의 응답시간을 가지는 문제가 발생하였다. 해당 문제를 해결하기 위해 서버를 구축하여 평가 모델은 서버에서 동작하며 Jetson Nano 보드는 이미지를 추출하여 서버로 전송, 결과를 출력하는 역할을 수행하도록 변경하였다. 이로 인해 평가 시 1 초 미만의 응답시간을 가지게 되었다.

본 논문에서 수행한 내용은 손동작 인식 S/W 가 필요한 다양한 분야에 적용 가능하다. 농아 장애인을 위한 프로젝트는 AI 가 수화 제스처를 문자로 번역해줌으로, 수화를 모르는

일반인을 대상으로도 원활한 의사소통이 가능함을 기대할 수 있다. 또한 IoT 와 결합하여 장소나 환경에 크게 제약 받지 않아서 사용 용이성 또한 기대할 수 있다. 또한, 본 논문은 소프트웨어 공학적 측면에서 고려해야 할 AI 프로젝트에서의 여러 요소를 보여준다. 따라서 AI 프로젝트를 수행하는 연구자 및 개발자들은 본 논문을 통해 소프트웨어공학적으로 고려할 요소를 참조할 수 있다.

7. 참고 문헌

[1] Bheda, Vivek and Radpour, Dianna. “Using Deep Convolutional Networks for Gesture Recognition in American Sign Language.”, arXiv preprint arXiv:1710.06836. 2017
[2] 박상윤, 이응주. “복잡한 영상에 강인한 손동작 인식 방법.” 멀티미디어학회논문지, 13(7), 1000-1015. 2010
[3] Knutsson, Adam. "Hand Detection and Pose Estimation using Convolutional Neural Networks." 2015.
[4] Eirini Mathe, Alexandros Mitsou, Evaggelos Spyrou, and Phivos Mylonas. “Hand Gesture Recognition using a Convolutional Neural Network.” In 2018 13th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP) (pp. 37-42). IEEE. 2018.
[5] 임현교, 김주봉, 권도형, 한연희, “MNIST 의 CNN 모델 기반 TensorFlow Optimizer 성능 비교 분석”, Advanced Technology Research Center, 2017
[6] 정 승 대, 장 경 호, 정 순 기, “실시간 손동작 인식을 위한 동작 평면 추정”, 정보처리학회논문지 B, VOL 16-B NO. 05 PP. 0347 ~ 0358, 2009
[7] 정인상, ”오픈 소스 소프트웨어로 실습하는 소프트웨어 공학”, 생능출판사, 2017

문서 유형에 따른 분류를 이용한 마이닝 기반 토픽 추출

구병국 [1O], 최소영 [2], 강지수 [3], 백지원 [4], 박찬홍 [5], 정경용 [6]

[1,2,3,4]경기대학교 컴퓨터과학과 데이터마이닝 연구실

[5]상지대학교 정보통신소프트웨어공학과, [6]경기대학교 컴퓨터공학부

gukman97@gmail.com, qpalqp3396@gmail.com, kangjs920@gmail.com,
hy1233hh@naver.com, roypark1984@gmail.com, dragonhci@gmail.com

Mining based Topic Extraction using Classification of Document Types

Byung-Kook Koo [1O], Soyoung Choi[2], Ji-Soo Kang[3], Jiwon Baek[4], Roy C. Park [5],
Kyungyong Chung[6]

[1,2,3,4]Data Mining Lab., Department of Computer Science, Kyonggi University

[5]Department of Computer Information Software Engineering, Sangji University

[6]Division of Computer Science and Engineering, Kyonggi University

요 약

정보기술의 발달로 인해 정보의 습득뿐만 아니라 생산과 공유가 용이하다. 하지만 대량의 데이터 생성으로 인한 정보의 범람화로 인해 문서, 블로그, SNS 등에서 원하는 정보의 수집이 어렵다. 이에 따라 직장, 학교 등 일상에서 가장 많이 사용하는 문서를 대표하는 단어들을 추출하는 토픽 추출 방법이 필요하다. 따라서 본 논문에서는 문서 유형에 따른 분류를 이용한 마이닝 기반 토픽 추출을 제안한다. 이는 크롤링을 통해 웹 페이지, 블로그 및 SNS의 리뷰 문서들을 수집한다. 수집한 데이터는 Topci과 Label 섹션으로 나누어 전처리 과정을 거친다. 문서 유형 정보를 활용하기 위해 관련 섹션의 데이터들을 k-평균 알고리즘을 사용해 클러스터링 한다. 클러스터를 사용하여 새로운 문서를 분류하기 위해 k-최근접 이웃 알고리즘을 이용한 분류기를 모델링 한다. 새로운 문서가 속한 클러스터의 트랜잭션에서 Apriori 알고리즘을 통해 연관 규칙을 생성한다. 이에 따라 문서 유형 별 토픽 추출이 가능하다. 성능평가 결과 실제 문서의 토픽에 대해 정밀도 0.69과 재현률 0.74을 가진다. 따라서 제안하는 토픽 추출 방법은 문서 유형을 고려한 적합한 토픽을 추출하는 방법이다.

1. 서 론

정보통신 소프트웨어의 발달로 정보를 습득할 수 있는 수단과 방법이 증가하고 있다. 특히 스마트 디바이스의 상용화로 실시간으로 SNS를 통해 정보가 확산되고 있다. 이에 따라 사용자들이 손쉽게 다양한 분야, 주관적인 생각 등에 따른 정보를 습득, 생산, 공유할 수 있다. 하지만 범람하는 문서들 사이에서 자신이 원하는 특정한 정보를 수집하는 것에 대한 어려움을 겪는다. 예를 들어 특정 키워드를 통해 문서를 찾으려고 할 때 키워드가 여러 분야에서 범용적으로 쓰이거나 다의어일 경우 원하는 문서를 찾기 어렵다. 또한 같은 키워드를 포함하고 있더라도 문서 유형에 따라 문서가 내포하고 있는 정보가 상이할 수 있다. 이와 같은 한계점을 해결 하기 위해 문서를 유형 별로 분류하여 수집할 필요가 있다[1].

토픽 추출은 문서를 대표하는 핵심 키워드들을 추출하는 기술이다[2]. 토픽 추출 과정을 통해 특정 문서에 대해 도출한 핵심 키워드들은 문서에 담긴 정보들을 파악하기 용이하게 한다. 이에 따라 토픽 추출을 위한 연구들이 진행되고 있으며 여러 모델들이 개발 중이다[3, 4].

인터넷 웹 상에 존재하는 정보들을 수집하여 필요한 데이터들을 추출하는 웹 크롤링 기술이 발전하고 있다. 웹 크롤링을 통해 수집한 텍스트 데이터는 텍스트 마이닝을 통해 유의미한 결과를 도출하는 것이 가능하다[5]. 따라서 본 논문에서는 웹 크롤링을 통해 문서들을 수집한다. 수집한 문서들을 바탕으로 문서 유형에 따른 분류를 이용한 마이닝 기반 토픽 추출 방법을 제안한다.

2. 관련연구

2.1 연관규칙을 이용한 탐색

연관 규칙 분석은 여러개의 데이터 집합에 대하여

데이터를 구성하고 있는 항목들 간 유의미한 규칙, 즉 연관규칙을 찾는 분석이다[6]. 연관규칙을 생성하기 위해서는 항목들이 얼마나 빈발하게 발생하였는지를 계산하여 최소지지도 이상의 빈발항목집합을 찾는다. 이를 이용해 조합 가능한 연관규칙들을 나열하여 일정 수준 이상의 지지도 및 신뢰도를 가지는 연관규칙을 최종 선별한다. 식(1)은 연관 규칙 A→B에 대한 지지도(Supp)와 신뢰도(Confidence)의 수식을 나타낸다. 지지도는 주어진 트랜잭션에서 A와 B가 포함되는 트랜잭션의 빈도를 의미한다. 신뢰도는 A가 발생하는 경우에 대한 B의 조건부 확률이다.

$$Supp(A \rightarrow B) = P(A \cap B)$$
$$Confidence(A \rightarrow B) = P(A|B) \qquad \text{(식 1)}$$

<그림 1>은 연관 규칙을 탐색하는 항목 집합 트리를 나타낸다. 항목 집합 트리는 부분집합의 특성을 이용한다. 만약 특정 항목 집합이 최소 지지도를 만족하지 않는다면 특정 항목을 부분 집합으로 가지는 모든 항목 집합들을 탐색 범위에서 제외하며 탐색한다. 빈발항목집합을 찾는 과정에서 만약 전체 항목이 k개일 때, 항목들에 대해 2^k-1 번의 탐색을 해야 하므로 항목 수가 많아질수록 탐색 횟수가 지수 배가 되는 어려움이 있다.

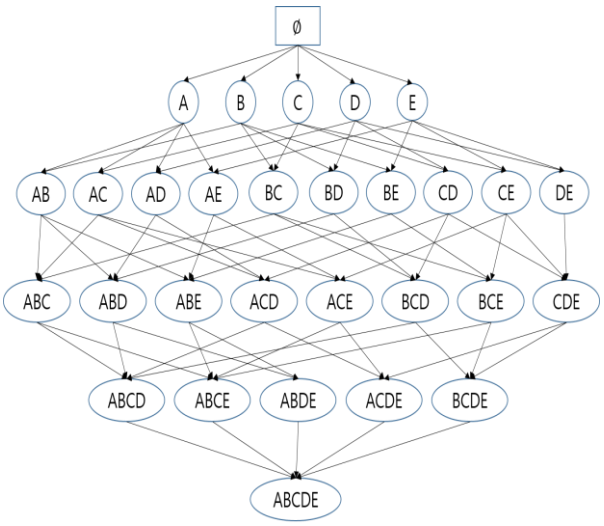


그림 1. 항목 집합 트리

반면에 연관 규칙을 찾는 알고리즘 중 Apriori 알고리즘은 후보항목 집합의 수를 절감하여 효율적으로 연관 규칙을 찾는 알고리즘이다. 이는 항목들에 대해 넓이 우선적이며 접두 탐색방법을 사용하여 빈발항목집합을 찾는다[7].

2.2 웹 크롤링을 통한 데이터 수집

웹 크롤링은 웹 페이지로부터 스크랩 방식을 통해 데이터를 추출하는 방법이다[8]. 월드 와이드 웹, 인터넷의 등장에 따라 방대한 양의 정보들에 접근 가능하게 되었다. 이에 웹에서 원하는 데이터를 수집하는 기술이 필요하게 되었다. 웹 크롤링은 인력에 의한 단순한 복사-붙여넣기 방법부터 HTTP 프로그래밍, HTML 및 DOM 파싱, 컴퓨터 비전 웹 페이지 분석 기술을 사용한 제한적으로 자동화 된 방법까지 다양한 기술들이 존재한다.

3. 문서 유형에 따른 분류를 이용한 마이닝 기반 토픽 추출

3.1 데이터 수집 및 전처리

문서 유형에 따른 분류를 이용한 마이닝 기반 토픽 추출을 위해 문서를 수집한다. 수집하는 문서는 국내기업 3곳과 블로그 및 SNS 등에서 웹 크롤링을 이용하여 국내외 여행지에 대한 사용자들의 리뷰와 관광명소, 여행 유형에 따라 저장한다[9-12]. 따라서 본 논문에서는 여행지에 따라 문서 유형을 결정한다. 문서 유형에 따라 토픽이 달라지기 때문에 문서 유형에 대한 메타 데이터를 수집한다. 메타데이터는 여행에 대한 총 경비, 인원 수, 형태, 일정 및 여행 시기이다. 웹 크롤링을 통해 수집한 문서들을 장소를 기준으로 묶는다. 수집된 문서들에 대해 장소들은 트랜잭션 생성을 위하여 Topic 섹션으로, 나머지 문서들은 트랜잭션들의 분류를 위한 Label 섹션으로 분류한다. Topic 섹션에 들어가는 장소들은 명칭이 들어간다. Label 섹션에 들어가는 경비 속성은 총 경비를 100,000으로 나눈 금액이다. 인원 수와 일정은 전처리 과정에 포함되지 않는다. 형태는 가족인 경우 0, 솔로인 경우 1, 커플인 경우 2, 단체인 경우에는 3으로 처리한다. 또한 데이터 분석 시스템의 성능 향상을 위해 Item 섹션의 항목 수가 3개 미만이거나 Label 섹션에 결측치 등 불필요한 데이터는 분석에서 제외한다. <표 1>은 데이터의 전처리 결과를 나타낸다. 두 개의 데이터에 대해 Topic 섹션의 대문자 알파벳이 들어간 공간에는 실제로 장소 명칭이 대응되며, Label 섹션에는 앞서 서술한 경비, 인원 수, 형태, 일정, 시기(월)이 전처리 되어 저장된다.

표 1. 데이터의 전처리 결과

Topic					Label				
					경비	인원수	형태	일정	월
1	A	B	C	D	110	2	2	4	3
	E	F	G	H					
2	I	J	K	L	60	1	3	3	8
	M	N	O	P					

3.2 유형에 따른 토픽 분석 및 추출

유형에 따른 토픽 분석 및 추출은 데이터 군집화, 새로운 데이터에 대한 분류, 연관규칙 생성의 단계로 진행한다. 첫 번째 단계는 Label 섹션의 경비, 인원 수, 형태, 일정, 시기를 parameter로 가지는 k-평균 알고리즘을 통해 토픽 섹션의 트랜잭션들을 군집화한다.

k값은 군집화하는 클러스터의 개수를 나타내며, Label 섹션 내 속성들의 조합으로 여러 경우가 나타남에 따라 본 논문에서는 k의 값을 16으로 한다. <그림 2>는 k-평균 알고리즘 결과를 나타낸다.

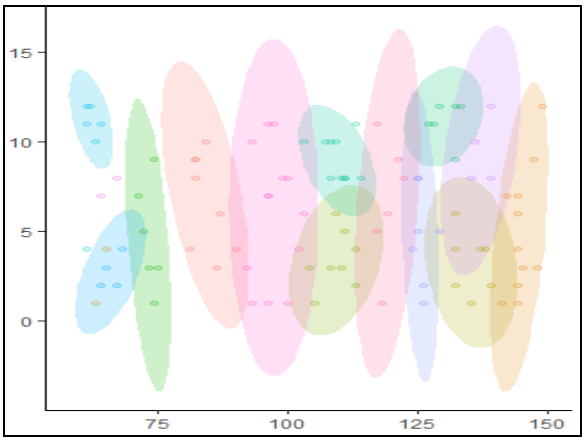


그림 2. k-평균 알고리즘 결과

두 번째 단계는 라벨의 연속형 변수를 범주형 변수로 변환하기 위해 데이터들의 분포를 기준으로 binning 한다. binning 된 데이터에 대해 k-최근접 이웃 알고리즘을 사용한 분류기를 모델링 한다. k-최근접 이웃 알고리즘은 새로운 데이터를 입력 받았을 경우 군집의 중심점과 가장 가까운 곳으로 분류한다[13]. 현 단계를 통해 새로운 여행 일정에 관련된 문서에 대해 분류기를 사용하여 문서의 여행 유형에 해당하는 클러스터의 트랜잭션들을 선발한다.

마지막 단계로 선발된 트랜잭션들에 대해 Apriori 알고리즘을 적용하여 유형에 따른 토픽들의 연관 규칙을 생성한다[14]. 연관 분석 알고리즘은 적용 시 제약 조건을 포함시키지 않으면 관련성이 떨어지는 연관 규칙을 생성하는 문제점이 있다. 따라서 본 논문에서 생성하는 연관 규칙은 지지도 0.2 이상 신뢰도 0.7이상으로 제한한다. <표 2>은 문서 유형에 따른 토픽들의 연관 규칙 결과이다. Supp.는 지지도를 의미한다. Conf.는 신뢰도, Lift는 향상도를 의미한다. 예를 들어 연관규칙 'Topic 3=> Topic 26'에 대한 지지도는 0.68, 신뢰도는 0.89, 향상도는 1.41이다. 연관 규칙의 향상도가 1보다 크기 때문에 두 항목간의 관련성이 있으며, 음의 상관관계를 가지지 않는다.

표 2. 문서 유형에 따른 토픽들의 연관 규칙

Rule	Supp.	Conf.	Lift
Topic 3 => Topic 26	0.68	0.89	1.41
Topic 5=> Topic 26	0.63	0.89	1.41
Topic 1 => Topic 5	0.42	0.80	1.27
Topic 5, Topic 7 => Topic 26	0.42	0.87	1.27
Topic 1, Topic 26 => Topic 5	0.38	0.70	1.67
…	…	…	…

<그림 3>은 <표 2>의 연관 규칙 결과를 네트워크 그래프로 시각화 하여 나타낸 것이다. 그림 3에서 원의 크기는 지지도에 비례하며, 색상은 향상도에 비례한다.

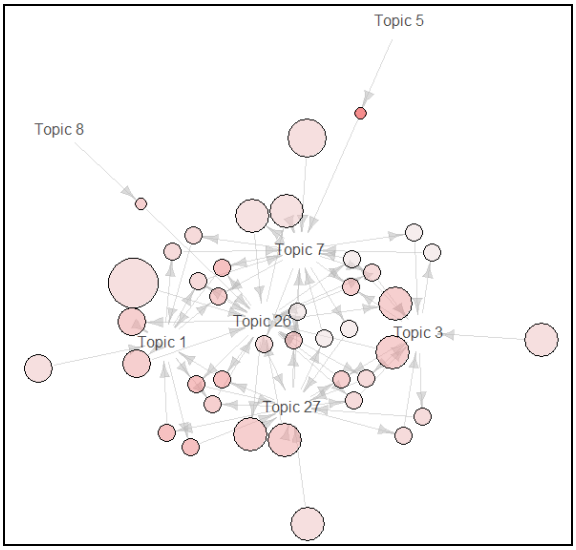


그림 3. 연관 규칙들의 네트워크 그래프

이와 같이 생성된 연관 규칙들을 이용하여 새로운 일정에 관련된 문서에 대해 문서 유형 별 토픽을 추출 가능하다.

3.3 문서 별 토픽 추출 과정

새로운 일정에 관련된 문서를 웹 크롤링을 하여 장소와 경비, 인원 수, 형태, 일정 및 시기를 입력한다. 희망 장소에 대한 데이터들은 k-평균 알고리즘을 통해 binning되어 군집화되어 있다. 이를 기반으로 k-최근접 이웃 알고리즘을 적용하여 모델링 한 분류기는 새로운 문서 유형이 어느 클러스터에 속하는지 분류한다. 새로운 문서 유형에 대해 같은 클러스터에 속하는 트랜잭션들을 선발한다. 선발한 트랜잭션들에 대해 Apriori 연관 분석 알고리즘을 적용하여 토픽들 간 연관규칙을 생성한다. <그림 4>은 유형에 따른 토픽 추출 과정의 구성도를 나타낸다.

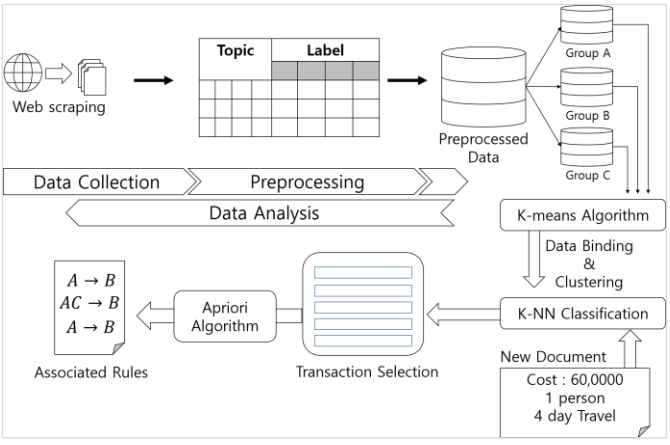


그림 4. 유형 별 토픽 추출 과정의 구성도

4. 성능평가

성능평가는 제안하는 방법의 분류와 연관규칙을 이용한 토픽추출 방법(KAbR)과 연관규칙만을 이용한 토픽 추출 방법(RbR)[15]의 정밀도(Precision)와 재현율(Recall)을 비교한다. 정밀도는 모델이 참이라고 분류한 것 중에서 실제 참인 경우의 비율을 의미한다. 또한 재현율은 실제로 참인 것 중에서 모델이 참이라고 분류한 비율을 의한다[16]. 성능 평가 방법은 분석 과정에서 사용하지 않은 문서에 대해 두 가지 방법을 이용하여 토픽을 추출한다. 추출한 토픽과 실제 문서의 토픽과 비교한다. *식 (2)*는 정밀도와 재현율을 나타낸다. 본 논문에서 True Positive(TP)는 추출한 토픽에 대해 중복되는 항목 수를 의미한다. False Positive(FP)는 중복되지 않는 항목 수를 의미한다. False Negative(FN)는 실제 문서의 토픽에 대해 추출한 토픽과 중복되지 않는 항목 수를 의미한다. True Negative(TN)는 문서가 속한 유형의 모든 토픽들에 대해 추출한 토픽과 실제 문서 모두 중복되지 않는 항목 수를 의미한다. <표 3>은 KAbR과 RbR의 정밀도와 재현률 비교 결과를 나타낸다[18].

$$Precision = \frac{TP}{TP \times FP}, \quad Recall = \frac{TP}{TP+FN} \qquad (식\ 2)$$

표 3. KAbR과 RbR의 정밀도와 재현율 비교 결과

시스템	정밀도	재현률
KAbR	0.69	0.74
RbR	0.53	0.57

<표 3>에서는 KAbR의 정밀도 0.69, 재현율 0.74로 평가되었고, RbR의 정밀도는 0.53, 재현율 0.57이 평가되었다. 이에 따라 본 논문이 제안하는 문서 유형에 따른 토픽 추출 방법이 문서 유형을 고려하지 않는 방법보다 정밀도 및 재현률의 성능이 우수하게 평가되었다.

5. 결 론

본 논문에서는 문서 유형에 따른 분류를 이용한 마이닝 기반 토픽 추출 방법을 제안하였다. 제안한 방법은 기업 웹 사이트, 블로그 및 SNS에서의 웹 크롤링을 통해 데이터를 수집하여 시스템에 입력되는 형식으로 전처리한다. 전처리한 데이터들을 장소 별로 그룹화한다. 사용자가 원하는 문서의 유형에 적합한 클러스터를 찾기 위해 그룹 별로 k-평균 알고리즘을 이용하여 군집화한다. 이를 기반으로 k-최근접 이웃 알고리즘을 이용한 분류 모델을 모델링한다. 새로운 문서에 대해 같은 클러스터에 속한 데이터들의 트랜잭션들을 선발하여 Apriori 알고리즘을 적용한다. 이에 대해 생성되는 연관규칙들을 사용하여 문서 유형 별 토픽 추출이 가능하다. 제안하는 텍스트 마이닝과 분류를 이용한 문서 유형에 따른 토픽 추출 방법은 실제 문서의 토픽에 대해 정밀도 0.6과 재현율 0.67을 보였다. 향후, 기존에 수집하는 유형들을 포함한 다양한 변수들을 섹션에 포함하여 문서 분류기의 정확도를 높이는 것으로 토픽 추출 방법의 정확도를 향상시킬 수 있게 한다.

Acknowledgement

This work was supported by the GRRC program of Gyeonggi province. [GRRC KGU 2018-B05, Smart Manufacturing Application Technology Research]

참고 문헌

[1] Y. Sun, E. Lank, M. Terry, "Label-and-Learn: Visualizing the Likelihood of Machine Learning Classifier's Success during Data Labeling", In Proceedings of the International Conference on Intelligent User Interfaces, 2017.
[2] D. M. Blei, Y. N. Andrew, M. I. Jordan, "Latent Dirichlet Allocation", Journal of machine Learning research, Vol. 3, pp. 993-1022, 2003.
[3] J. Noh, S. Lee, "Extracting and Evaluating Topics by Region", Multimedia Tools and Application, Vol. 75 No. 20, 2016
[4] C. H. Papadimitriou, "Latent Semantic Indexing: A Probabilistic Analysis", Journal of Computer and System Sciences, Vol. 61 No. 2, pp. 217-235, 2000.
[5] R. Feldman, J. Sanger, "The Text Mining Handbook" Cambridge University Press, 2007.
[6] P. S. Gregory, "Discovery, Analysis, and Presentation of Strong Rules", Knowledge discovery in databases, pp. 229-238, 1991.
[7] R, Agrawal, S. Ramakrishnan, "Fast Algorithms for Mining Association Rules", In Proc. of 20th International Conference Very Large Databases, Vol. 1215, 1994.
[8] B. Geoff, W. Paul, "New insights into Rental Housing Markets across the United States: web scraping and analyzing craigslist rental listings", Journal of Planning Education and Research, Vol. 37, No. 4, pp. 457-476, 2017.
[9] Mode Tour, "www.modetour.com".
[10] Hana Tour, "www.hanatour.com".
[11] Interpark Tour, "tour.interpark.com".
[12] J. C. Kim, K. Chung, "Associative Feature Information Extraction using Text Mining from Health Big Data", Wireless Personal Communications, Vol. 105, No. 2, pp. 691-707, 2019.
[13] Altman, Naomi S, "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression" The American Statistician, Vol. 46 No. 3, pp. 175-185,

1992.
[14] Y. Kim, "A Study on Design and Implementation of Personalized Information Recommendation System based on Apriori Algorithm", The Korean Biblia Society for Library and Information Science, Vol. 23 No. 4, pp. 283-308, 2012.
[15] K. H. Lee, C. Kim, "Constructing Knowledge Structure based on Association Rules", The Korean Institute of Information Scientists and Engineers, pp. 235-237, 2018.
[16] Al-Shalabi, Riyad, R. Obeidat, "Improving KNN Arabic text classification with n-grams based document indexing", Proceedings of the Sixth International Conference on Informatics and Systems, 2008.
[17] D. M. W. Powers, "Evaluation: From Precision, Recall and F-measure to ROC, informedness, markedness and correlation", Journal of Machine Learning Technologies, Vol. 2, No.1, pp. 37-63, 2011.

CNN 모델 평가를 위한 이미지 데이터 증강 도구 개발[1]

최영원O, 이영우, 채흥석
부산대학교 전기컴퓨터공학부
fenrir94@pusan.ac.kr, amorepooh@pusan.ac.kr, hschae@pusan.ac.kr

Development of a Data Augmentation Apparatus to Evaluate CNN Model

Youngwon ChoiO, Youngwoo Lee, Heung-Seok Chae
Dept. of Electrical and Computer Engineering, Pusan National University

요 약

CNN 모델이 이미지 인식과 객체 탐지 등 여러 이미지 분야에 활용됨에 따라, CNN모델의 다양한 환경에 대한 학습 정도를 평가하고자 한다. 평가 방법으로는 기존 테스트 이미지의 객체 이미지와 배경 이미지를 합성하여 새로운 테스트 이미지를 생성하고, 생성한 새로운 테스트 이미지로 CNN 모델을 평가한다. 사례 연구로 Pascal VOC2007을 학습한 YOLOv3 모델을 새로운 테스트 이미지로 평가를 하였으며, 기존 테스트 이미지의 mAP보다 새로운 테스트 이미지의 mAP가 약 0.11 더 낮아지는 것을 확인하였다.

1. 서 론

Convolutional Neural Network(CNN)는 이미지 내 객체 인식과 객체 탐지에 널리 사용되는 심층 신경망 모델이다. 객체 탐지는 이미지나 영상 속 객체를 탐지하는 것으로 자동차의 자율주행 기술에 활용되고 있다. 그러나 자율주행 자동차를 활용한 사업은 제한된 지역에서만 서비스되고 있다. 자동차는 안전 필수 시스템이므로, 서비스하려는 도시 내에서 CNN 모델의 성능을 신뢰할 수 있어야 한다. 따라서 여러 환경에 대해서 CNN 모델의 학습 정도를 평가하기 위한 방법이 필요하다.

본 연구에서는 CNN 모델을 평가하기 위해 CNN 모델의 테스트 이미지로부터 객체 이미지를 추출하고, 추출한 객체 이미지와 배경 이미지를 합성하여 새로운 테스트 이미지로써 합성 이미지를 생성한다. 사례 연구로 생성된 합성 이미지로 CNN 모델을 평가한다.

2. 배경 지식

2.1. Metamorphic Testing

Metamorphic Testing은 입력에 따른 프로그램 출력들을 통해 추론되는 관계로 테스트 케이스를 생성하는 방법이다[1]. Metamorphic Testing에서 프로그램 출력을 통해 예상되는 관계를 Metamorphic Relation이라 한다[1].

본 연구에서는 테스트 이미지 수를 증가시키 위해서 합성 이미지 생성과 Data Augmentation 기법들을 적용한다. 해당 이미지 변형 기법들을 적용한 변형된 이미지에 기존 이미지 내 객체가 존재하므로, CNN 모델은 변형된 이미지의 Label을 테스트 이미지의 Label과 동일하게 판별할 수 있어야 한다. 이에 Metamorphic Relation으로 테스트 이미지와 변형된 이미지 각각의 Label은 같음을 적용한다.

2.2. Instance Segmentation

Instance Segmentation은 이미지 내 객체의 형태에 따른 Mask 정보를 계산하는 방법이다. Instance Segmentation을 하는 심층 신경망 모델로는 Mask R-CNN이 있다[2].

본 연구에서는 객체 이미지 추출 방법으로 Instance Segmentation을 적용한다. MS COCO 데이터 셋[3]을 학습한 Mask R-CNN 모델을 사용하여 입력한 이미지의 Mask와 Bounding Box 정보를 이용하여 이미지에서 배경을 제거한 객체 이미지를 추출할 수 있다.

3. 합성 이미지 생성 도구

Composed Image Generator는 Object Extraction과 Image Composition 2가지 과정으로 이루어져 있다. 그림 1은 합성 이미지 생성 도구 Composed Image Generator의 전체 동작 과정을 나타낸다.

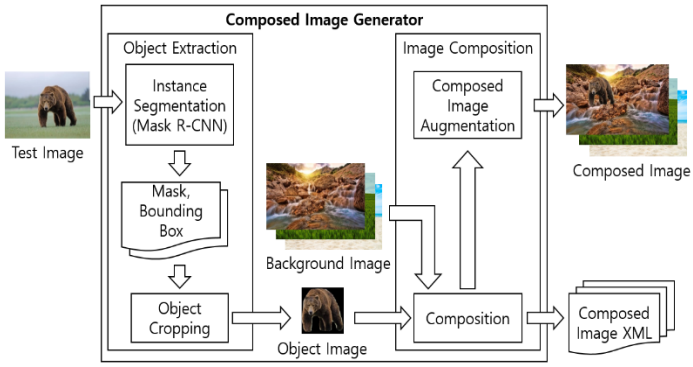


그림 1 Composed Image Generator 동작 과정

Object Extraction에서는 이미지를 입력하면 Mask R-CNN을 통해 객체 이미지를 추출한다. Image Composition에서는 객체 이미지와 배경 이미지를 합성하여 합성

[1] 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2019R1F1A1063336)

이미지와 합성 이미지 내 객체의 Label과 Bounding Box 정보를 저장한 XML 파일을 생성한다. 이후 합성 이미지에 Data Augmentation 기법을 적용한다.

4. 사례 연구

4.1. 평가 환경

테스트 이미지는 Pascal VOC2007 Data Set[4]의 테스트 데이터 중 Label 10개를 사용한다. 평가 모델은 Pascal VOC2007과 Pascal VOC2012를 학습한 YOLOv3[5] 모델을 사용한다. 배경 이미지는 이미지 사이트 Pixabay [2] 에서 Crawling한 후 이미지 분류 Tag에 따라서 24장을 선택한다.

합성 이미지 생성을 위해 테스트 이미지는 테스트 이미지의 10개 Label마다 이미지 20장을 선택하여, 총 200장을 선택한다. 선택한 테스트 이미지 200장과 배경 이미지 24장을 이미지 합성 도구에 입력하여 합성 이미지 4,800장을 생성한다. 생성한 합성 이미지에 Data Augmentation 기법으로 Brightness와 Noise, Weather를 적용한다. 각 기법을 적용한 합성 이미지 수는 아래와 같다.

- Composed Image Brightness:
  4,800 images * 4 brightness(−100, −50, +50, +100) = 19,600 images
- Composed Image Noise:
  4,800 images * 4 noise(Gaussian, Salt & Pepper, Poisson, Speckle) = 19,600 images
- Composed Image Weather:
  4,800 images * 3 weather(rain, fog, snow)
  * 2 effects/weather = 28,800 images

4.2. 평가 결과

모델 평가에는 mean Average Precision(mAP)과 Average Precision(AP)을 사용한다. Bounding Box 성능 평가에 대한 IoU Threshold는 0.5로 설정한다. 각 테스트 표 1은 각 테스트 데이터들을 사용하여 평가 모델을 평가한 결과 mAP와 각 Label의 AP를 정리한 표이다.

테스트 이미지에 대한 mAP보다 합성 이미지에 대한 mAP가 더 낮게 측정되었으며, 합성 이미지 Noise에 대한 mAP가 가장 낮게 측정되었다.

테스트 이미지와 합성 이미지의 AP를 비교하면, Train과 Bicycle, Cat의 AP가 상대적으로 많이 감소하였다. 반면에 Bus와 Cow, Person는 다른 Label보다 상대적으로 작게 감소하였다. Bus와 Cow, Person의 AP가 작게 감소한 이유는 다른 Label에 비해 객체 이미지가 일관된 형태여서 배경 변경에 대해서도 일반화가 더 잘 되었다고 판단한다.

합성 이미지의 AP를 기준으로 합성 이미지 Noise의 AP는 Cat 이외 Label에서 Noise와 Weather를 적용한 두 데이터 셋의 AP보다 더 큰 폭으로 감소하였다.

5. 결론 및 향후 연구

CNN 모델의 성능을 평가하기 위해 새로운 테스트 이미지를 생성하는 도구를 개발하였다. 테스트 이미지에서 추출한 객체 이미지와 배경 이미지를 합성한 새로운 테스트 이미지를 생성하여 CNN 모델을 평가하였다. 그 결과 기존 테스트 이미지로 모델을 평가한 결과보다 새로운 테스트 이미지로 모델의 평가한 결과의 mAP와 AP가 전체적으로 더 낮게 나오는 것을 확인하였다.

본 연구에서는 이미지 합성 시 객체 이미지와 배경 이미지 간의 연관성을 고려하지 않고 배경 이미지 내 임의의 좌표에 합성하였다. 향후 연구에서는 테스트 이미지의 Label과 배경 이미지의 Tag 간 연관관계를 정의하고, 이를 통해 합성한 이미지로 모델을 평가하는 연구를 수행할 계획이다.

참고 문헌

[1] T. Y. Chen et al., “Metamorphic Testing: A New Approach for Generating Next Test Cases,” Computer Science, Hong Kong Univ. of Science and Technology, Tech. HKUST-CS98-01, 1998.

[2] K. He et al., “Mask R-CNN,” in *ICCV*, pp. 2961-2969, 2017.

[3] T. Y. Lin et al, “Microsoft COCO: Common Objects in Context,” in *ECCV*, pp. 740-755, 2014.

[4] M. Everingham, et al., “The Pascal Visual Object Classes (VOC) Challenge,” *IJCV*, vol. 88, pp. 303-338, 2010.

[5] J. Redmon, and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv preprint arXiv: 1804.02767*, 2018.

표 1 테스트 데이터 별 mAP와 각 Label의 AP

테스트 데이터	mAP	Label(AP)									
		Bicycle	Bus	Car	Cat	Cow	Dog	Horse	Motorbike	Person	Train
테스트 이미지	0.930	0.930	0.953	0.943	0.940	0.889	0.928	0.931	0.939	0.914	0.935
합성 이미지	0.816	0.735	0.906	0.861	0.759	0.877	0.760	0.783	0.883	0.901	0.697
합성 이미지 Brightness	0.735	0.686	0.871	0.826	0.673	0.738	0.667	0.685	0.802	0.819	0.585
합성 이미지 Noise	0.645	0.588	0.716	0.746	0.588	0.659	0.581	0.603	0.690	0.754	0.522
합성 이미지 Weather	0.698	0.671	0.850	0.811	0.542	0.676	0.634	0.654	0.793	0.773	0.578

[2] Pixabay GmbH, Pixabay, https://pixabay.com

YOLO 모델을 활용한 영상 내의 유해 정보 검출 및 필터링 시스템 개발

조성윤, 권용준, 김채록, 최지원, 김석호, 최나람, 김민석, 정순기

경북대학교 컴퓨터학부

whtj78@naver.com, iyjkwon@naver.com, rhffpa23@gmail.com, acantos1@naver.com
shkim@vr.knu.ac.kr, nrchoi@vr.knu.ac.kr, minseokkim@vr.knu.ac.kr, skjung@knu.ac.kr

A system for detecting and filtering harmful information in video using YOLO

Seong-Yun Cho, Yong-Jun Kwon, Chae-Rok Kim, Ji-Won Choi

Seock-Ho Kim, Na-Ram Choi, Min-Seok Kim, Soon Ki Jung

Computer Science & Engineering, Kyungpook National University

요 약

현행 방송심의규정에 따르면 흡연 및 음주, 흉기를 사용하는 장면 등은 일부 연령층에게 부정적인 영향을 줄 수 있으므로 해당 장면을 다룰 경우에는 그 표현에 신중을 기할 것을 명시하고 있다. 각 방송사들은 이러한 장면들을 유해 정보로 규정하고, 이를 동영상 편집 프로그램을 이용해 필터링하는 작업을 수행하고 있다. 이러한 필터링 방식은 사람에 의해 수동으로 이루어지고 있으며 이는 시간과 비용이 많이 든다. 따라서 본 논문은 기존 필터링 방식의 번거로움을 해소하기 위해 딥러닝 모델을 통해 관련 유해 정보를 자동으로 검출하고 필터링하는 시스템을 제안한다. 최종 시스템은 수집된 데이터 셋을 YOLO 프레임워크를 통해 학습시키고 학습된 모델을 이용하여 유해 정보를 검출 및 블러링 처리를 한 뒤 비디오 포맷으로 저장된다.

1. 서 론

2019년 4월 23일 한국건강증진개발원의 보도자료[1]에 따르면 드라마나 영화와 같은 미디어 매체에 담배나 흡연 장면이 빈번하게 등장하는 것으로 파악되었다.[2] 현행 방송심의에 관한 규정 28조(건전성)의 내용에 의하면 방송에서 음주, 흡연 등의 내용을 다룰 때에는 이를 미화하거나 조장하지 않도록 신중하게 표현하라고 명시하고 있다.[3] 또한 제45조(출연) 제4항을 보면 방송은 어린이와 청소년이 흡연이나 음주하는 장면을 묘사할 수 없도록 하고 있으며, 잘못된 흡연, 음주 문화를 일반적인 상황으로 인식하지 않도록 신중하게 표현하라고 명시하고 있다.[4] 그렇기 때문에 방송사들은 방송심의규정을 준수하기 위해 영화를 제외하고 드라마나 예능 프로그램 등에 흡연 장면이 나올 경우 현행 방송심의 규정에 의거하여 그 장면들에 대해 자율적으로 필터링 처리를 하고 있다.[5] 각 방송사들은 영상물 컨텐츠 제작 및 편집을 위해서 Non-Linear Editing(NLE, 비선형 편집) 시스템을 이용한다. 비선형 편집 시스템은 컴퓨터를 통해 동영상 편집 소프트웨어를 사용하여 영상을 편집하는 시스템을 일컫는다.[6] 관련 소프트웨어로는 Canopus(현재 Grass Valley사에 인수됨)의 Edius Pro, Apple사의 Final Cut Pro, Adobe사의 Premiere Pro 등이 있다. 일례로 한국의 방송사 중 하나인 KBS는 영상 편집을 위해 Final Cut Pro와 Edius Pro를 사용하고 있다.[7] 프로그램을 사용해 필터링 처리를 하기 위해선, 본 영상 위에 필터링 효과를 적용하고 대상의 움직임에 따라 효과가 이동하도록 수작업으로 일일이 적용시켜야 한다. 미디어 매체에 담배나 흡연 장면이 자주 등장하고 있는 상황에 이러한 방식은 비효율적이며 시간과 비용이 많이 든다.

따라서 본 시스템은 유해 정보를 담배나 흡연 장면으로 한정하고 딥러닝 모델을 통해 흡연 장면의

담배 영역을 검출 및 필터링하여 기존 필터링 방식의 번거로움을 해결하고자 한다. 시스템의 작동 단계를 간단하게 나타내면 그림 1과 같다.

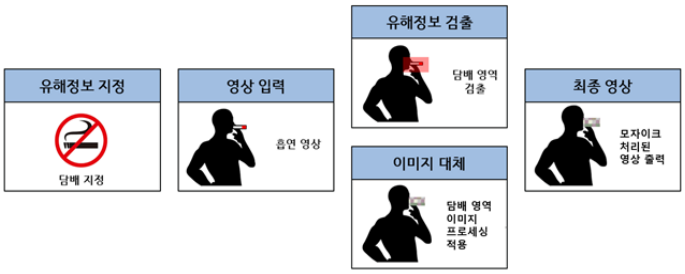


그림 1. 시스템 작동 단계

2. 본 론

2.1 기존의 필터링 방식

시스템을 개발하기에 앞서서 수작업으로 필터링하는 과정을 체험해보기 위해 비선형 편집 프로그램들 중에서 필터링을 위한 편집 프로그램으로서 Premiere Pro 2019와 Edius 9 Pro Trial을 임의로 선정하여 동영상 필터링 작업을 했다. Premiere에서 사용할 수 있는 효과에 가우시안 블러가 포함되어 있고[8], Edius도 마찬가지로 7 버전부터 가우시안 블러 효과 기능을 지원하기 시작했다.[9] Premiere로 필터링을 하기 위해선, 프로그램을 실행시키고 필터링 처리할 동영상을 읽어들인 다음 타임트랙에 동영상을 넣어주고 필터링 대상이 나오는 부분을 다른 영상 부분과 구분되도록 자른다. 그 후 비디오 효과 목록에서 가우시안 블러 효과를 찾아 필터링할 영역의 타임라인에 드래그해서 넣어주고 임의의 수치를 입력하여 효과를 적용한다. 이 때 타원형이나 사각형 상자 툴, 또는 직접 영역을 그릴 수 있는 펜 툴로 마스크를 그려 필터링 적용 범위를 한정시킬 수 있다. 이 때 필터링 대상이 움직일 경우, 마스크 패스 기능을 통해 필터링이 적용된 마스크가 자동으로 대상의 움직임을 반영하여 추적하는 기능이 있기는 하다. 하지만 담배 같이 비교적 작고 구분하기 힘든 영역은 의도한 대로 인식되지 않는 경우가 많다. 그럴 경우엔 키프레임을 따라서 필터링 마스크 영역을 필터링 대상에 맞도록 하나의 프레임마다 직접 이동시켜야 한다는 단점이 있다. Edius의 필터링 방식과 단점도 Premiere와 크게 다르지 않다. 다만 Edius를 사용할 경우 마스크 효과를 먼저 생성한 뒤에 마스크 내부에 가우시안 블러 효과를 적용시켜야 한다는 차이점이 있다.

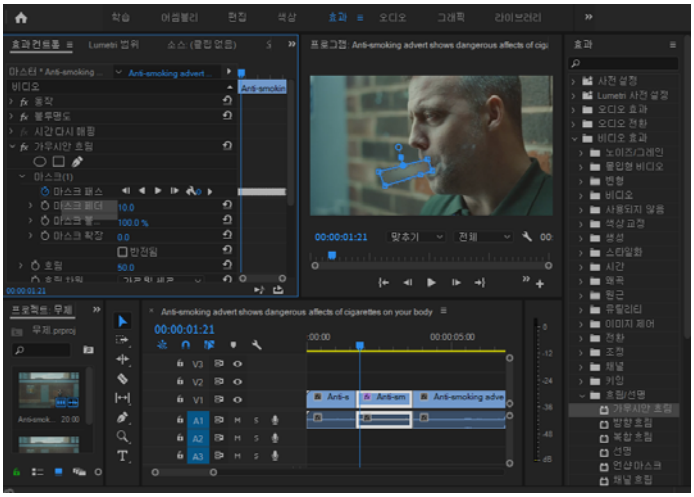


그림 2. Premiere Pro를 사용한 예시

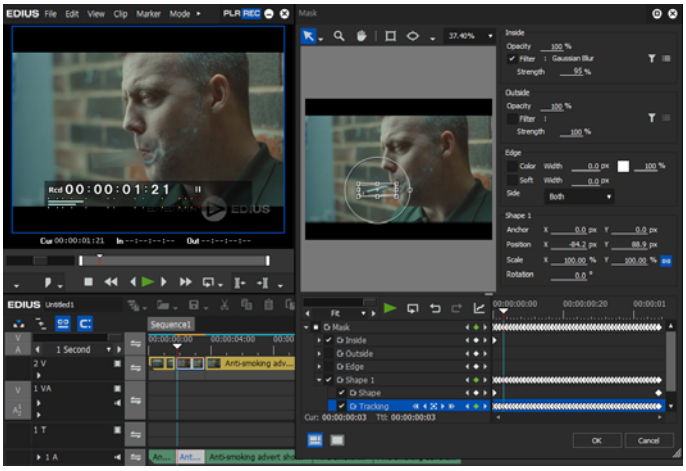


그림 3. Edius 9 Pro Trial을 사용한 예시

2.2 딥러닝 모델의 개요

딥러닝을 설명하기에 앞서 머신러닝을 설명할 필요가 있다. 머신러닝은 학습할 데이터를 주면 이것을 스스로 학습하는 기계학습을 의미한다. 딥러닝은 머신러닝의 한 종류이며 대형 인공 신경망을 형성하는 일종의 기계학습을 의미한다. 대규모 인공 신경망에 학습 알고리즘과 데이터를 공급하여 처리할 데이터를 학습하는 능력을 지속적으로 개선한다. 따라서 어떠한 문제를 데이터를 통해 컴퓨터가 스스로 처리하고 해결할 수 있도록 하는 기계학습 기술이다.[10] 딥러닝 기반 객체 탐색 기법에는 DPM, Faster RCNN, R-FCN, YOLO 등이 있다. 그 중 YOLO는 오픈소스로 제공되어 있고 라이센스의 제약도 없어 사용하기 간편하다는 장점이 있다.[11] 그래서 본 시스템은 YOLO를 사용하였고, YOLO의 모델 중 Darknet-53 모델[14]을 선택하였다.

2.3 YOLO 모델의 개요

YOLO는 You Only Look Once의 약자로, 이미지 내의 경계 박스와 클래스 확률을 단일 회귀 문제로 간주하여, 이미지를 한 번 보는 것으로 객체의 종류와 위치를 추측하는 딥러닝 모델이다. 그림 4와 같이 단일 신경망 네트워크를 통해 다중 경계 박스에 대한 클래스의 확률을 계산하는 단일 단계 방식의 객체 탐지 딥러닝 모델이다. YOLO 모델이 가진 장점은 크게 세가지가 있다. 첫째, 처리과정이 간단하여 실시간으로 객체 탐지가 가능하다. 따라서 속도가 빠르고 기존의 다른 실시간 객체 탐지 모델들과 비교하였을 때 평균 정확도가 2배 정도 높다는 특성을 보인다. 둘째, 이미지 전체를 한 번에 바라보는 방식을 차용하여 클래스에 대한 맥락적인 이해도가 높다. 그로 인해 배경으로 인한 클래스 예측 오류가 낮다는 특징이 있다. 셋째, 객체에 대한 좀 더 일반화된 특징을 학습한다. 만약 자연의 이미지를 학습하거나 테스트 했을 때, 다른 객체 탐지 시스템들에 비해 훨씬 높은 성능을 보여줄 수 있다. [12, 13]

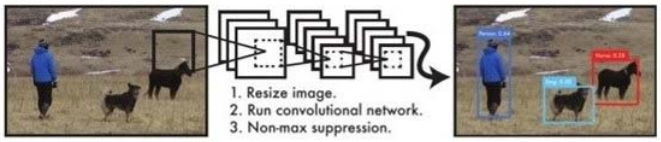


그림 4. YOLO 객체 탐지 시스템[13]

2.4 YOLO 모델의 객체 검출 과정

이미지를 동일한 크기의 S X S 그리드 영역으로 나눈다. 이 때 Background, Cigarrete 클래스로 학습한 YOLO의 Darknet-53 모델[14]을 사용하여 각각의 그리드 셀은 B개의 경계 박스와 각 경계 박스에 대한 신뢰 점수[1]를 예측한다. 만약 셀에 객체가 존재하지 않을 경우 신뢰 점수는 0이 된다. 각각의 경계 박스는 x, y, w, h, confidence 다섯 가지 요소로 구성된다. 여기서 x와 y는 경계 박스의 중심점, 즉 그리드 셀의 경계를 기준으로 한 상자의 중심 좌표를 의미하며, 그리드 셀의 범위에 대한 상대적 값이 입력된다. w와 h에는 예측된 객체와 전체 이미지의 너비, 높이에 대한 상대적 값이 입력된다. 마지막으로 confidence는 예측된 경계 박스와 ground truth 경계 박스 사이의 IOU 값을 의미한다. IOU는 Intersection Over Union의 약자로, 예측된 경계 박스와 ground truth 경계 박스 간의 차이를 평가하는 척도다. 즉, 예측된 경계 박스와 ground truth 경계 박스가 얼마나 겹쳐지는지를 판단하는 값이다. 또한 각각의 그리드 셀은 C개의 조건부 클래스 확률[2]을 갖는다. 이러한 확률은 객체를 포함하는 그리드 셀에 따라 달라진다. 하나의 그리드 셀은 예측된 경계 박스 수 B와는 상관없이 한 가지 종류의 클래스에 대한 확률만 계산한다. 테스트 할 때는 조건부 클래스 확률과 경계 박스의 신뢰 점수를 곱하여 특정 클래스의 신뢰 점수[3]를 얻는다. 이러한 점수는 해당 경계 박스가 클래스를 잘 나타내는지, 경계 박스가 얼마나 객체와 잘 맞춰져 있는지에 대한 정보를 나타낸다. 예측된 경계 박스의 신뢰도를 판단하는 기준은 IOU 값 0.5 이상이며 신뢰도가 높을수록 경계 박스의 테두리가 굵어진다. [13, 15]

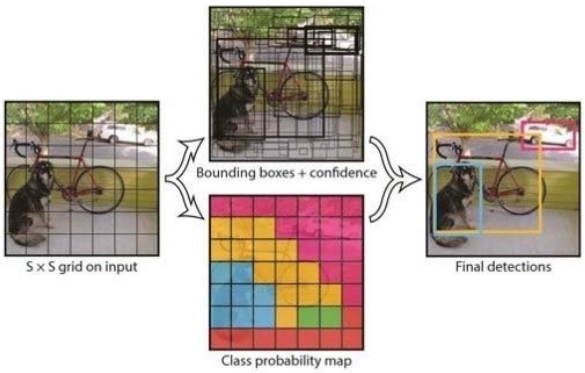


그림 5. YOLO 객체 검출 과정[13]

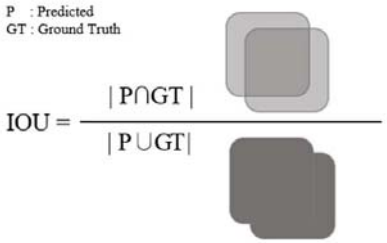


그림 6. IOU 평가[15]

2.5 블러링의 개요

블러링(blurring)은 이미지나 동영상의 세세한 부분을 제거하여 흐리게 하거나 부드럽게 하는 기술로, 스무딩(smoothing)이라고도 한다. 블러링 기법에는 평균값 필터를 사용하는 기법과 가우시안 필터를 사용하는 기법 등이 있다. 평균값 필터 기법은 입력 영상에서 특정 픽셀과 주변 픽셀들의 산술 평균을 결과 영상 픽셀 값으로 설정하여 블러링하는 방법이다.

1 $Pr(Object) * IOU_{pred}^{truth}$

2 $Pr(Class_i|Object)$

3 $Pr(Classi|Object) * Pr(Object) * IOU_{pred}^{truth} = Pr(Classi) * IOU_{pred}^{truth}$

가우시안 필터 기법은 가우시안 분포 함수[4]를 사용하여 생성한 필터 마스크를 사용해 블러링하는 방법이다. 가우시안 블러는 시그마 값을 조절하여 이미지를 얼마나 흐리게 할 지 간편하게 조절할 수 있으며 평균값 필터에 비해 좀 더 자연스러운 블러링 처리가 가능하다는 장점이 있다. 그래서 본 시스템은 필터링 방식으로 가우시안 블러를 사용하였다.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

그림 7. Darknet-53 모델 구조[14]

2.6 데이터 수집 및 학습

본 시스템에서 담배와 흡연 장면을 검출하기 위한 데이터 셋을 수집하기 위해 구글 크롬 확장프로그램(Fatkun 일괄 다운로드 이미지)[16]을 사용하여 관련 이미지를 일괄 다운로드 하는 방법과 흡연 장면이 나오는 영상의 프레임을 추출하는 방법을 사용하였다. 그리고 다운로드한 이미지들을 중에서 이름이 서로 같을 경우 이미지의 내용을 비교하여 중복된 데이터라 판단되면 제거하였다. 그 후 이미지를 크기 별로 정렬하여 크기가 서로 같은 이미지일 경우 확인 작업을 거쳐 같은 이미지일 경우 제거하는 작업을 거쳤다. 그리고 수집한 데이터 셋에서 객체를 식별하기 어렵거나 검출할 객체가 없는 경우를 제거하여 총 6010장의 이미지를 수집했다. 라벨링을 위해 이미지의 포맷은 JPEG로 통일하였다. 그런 다음 각각의 이미지에 대해 YOLO Mark[17] 프로그램을 사용하여 이미지 라벨링 작업을 진행하였다. 담배를 인식하기 위해 담배 영역은 'Cigarrete' 이라는 클래스로 라벨링 하였다. 그러나 영상에서 담배 영역의 일부가 신체 부위에 가려지거나 영상에서의 담배 각도가 변하여 흡연 장면에서 담배를 인식하지 못하는 문제가 발생할 수 있고, 이로 인해 필터링 정확도가 떨어질 수 있다는 문제점이 생길수 있다. 이 문제를 보완하기 위해 그림 8과 같이 담배를 쥐거나 물고 있는 손과 입 부분을 'Background'라는 클래스로 라벨링하고, 'Background' 클래스 영역은 'Cigarrete' 클래스 영역을 포함시켰다.

흡연을 할 땐 담배를 손가락에 쥐고 있거나 입에 물고 있다는 점을 고려하여 'Background' 클래스 영역을 같이 검출하면 담배를 인식하지 못하고 흡연 장면만 인식하는 경우를 필터링하여 보다 정확도를 높일 수 있다. 손에 담배를 들고 있는 이미지, 입에 담배를 물고 있는 이미지, 손과 입이 모두 포함된 이미지들에 대해 'Background' 클래스 라벨링 규약을 정하여 작업하였다. 상세한 클래스 라벨링 규약은 표 1과 같다. 라벨링 작업이 끝난 이미지는 그에 대응하는 텍스트 파일이 생성되는데, 텍스트 파일에는 클래스의 번호와 그에 대응하는 바운딩 박스의 좌표 값이 적혀있다. 이러한 이미지와 텍스트 파일로 구성된 데이터 셋은 CPU i7-8000, RAM 16GB, GPU RTX 2080 super, CUDA 10.1, cudnn 7.6 하드웨어 환경에서 YOLOv3 딥러닝 모델을 실행시키고 8000번에서 14000번까지 1000번 단위로 학습을 진행하여 7개의 가중치 파일을 만들어냈다.

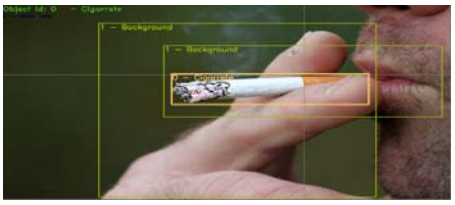


그림 8. YOLO Mark[17]를 사용한 클래스 라벨링 작업

표 1 클래스 라벨링 규약

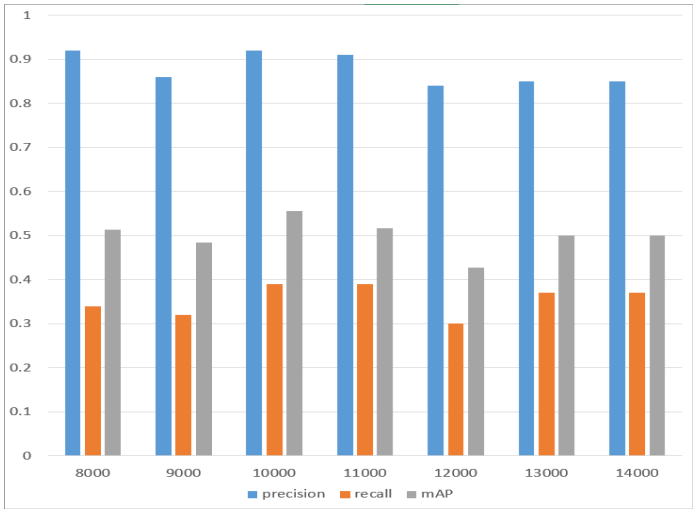
'Cigarrete' 클래스는 담배 부분에 딱 맞게 라벨링할 것
'Background' 클래스 안에 'Cigarrete' 클래스를 포함시킬 것
담배가 손에 들려있으면 손과 담배 부분을 'Background' 클래스로 라벨링할 것
담배가 입에 물려있으면 입과 담배를 'Background' 클래스로 라벨링할 것
담배를 손으로 잡고 있으며 입에 물고 있으면 손과 담배 부분을 'Background' 클래스로 라벨링하고 입과 담배 부분을 다시 'Background' 클래스로 라벨링할 것
이미지에서 잘 안 보이는 부위를 임의로 추정하여 라벨링하지 말 것

4 $G(x,y) = \frac{1}{2\pi\sigma^2} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)}$

(2차원 공간에서 평균이 (0,0), 표준편차가 1인 경우)

2.7 학습 결과 검증

각 7개의 가중치 파일들에 대해 기존 데이터 셋에 없던 100장의 이미지 데이터 셋으로 교차 검증을 수행하여 각 가중치 파일의 정확도(precision), 재현율(recall), 평균 정확도(mAP) 값을 계산하였고, 그림 9와 같이 10000번 학습시킨 파일의 정확도, 재현율, mAP값이 제일 높게 측정된 것을 확인하였다. 이 결과를 토대로 하여 시스템에 사용할 가중치 파일을 10000번 학습시킨 파일로 정하고 최종 시스템을 실행시켰다. 시스템을 실행시킨 화면은 그림 10과 같다.



Epoch	8000	9000	10000	11000	12000	13000	14000
precision	0.92	0.86	0.92	0.91	0.84	0.85	0.85
recall	0.34	0.32	0.39	0.39	0.3	0.37	0.37
mAP	0.51274	0.48468	0.55538	0.51606	0.42701	0.4999	0.4999

그림 9. 기존 데이터 셋과 중복되지 않는 100장의 이미지로 계산한 precision, recall, mAP 값 결과 표

그림 10. 시스템 실행 화면 (좌측부터 원본 영상, 영역 검출 후 필터링한 영상, 최종 영상)

3. 결 론

본 논문에선 YOLO 모델을 통한 유해 정보의 학습과 필터링 자동화 시스템에 대하여 기술하였다. 기존의 유해 정보 필터링 방식은 번거롭고 시간이 오래 걸리는 작업이다. 특히 영상이 길면 길수록, 유해 정보들이 많이 나올수록 영상 필터링은 고된 작업이 될 수밖에 없다. 따라서 본 시스템은 담배와 같은 유해 정보를 필터링 하는데 드는 시간과 노력를 줄이고 그것을 다른 편집 작업에 투자할 수 있도록 하여 동영상 편집의 효율을 높일 수 있을 것이다. 또한 이 연구의 원리를 토대로 방송에서 검열 대상이 되는 여러 객체들에 대한 데이터 셋을 모아 학습시키면 특정 검열 대상만을 필터링하는 시스템을 개발할 수 있다. 이러한 시스템을 여러 비선형 편집 프로그램의 추가 플러그인으로 배포한다면 동영상 편집의 편의성을 크게 높일 수 있다는 기대 효과를 생각해볼 수도 있을 것을 것이다. 그리고 특정한 객체를 검출해낸다는 점에 창안하여 이미지나 영상에서 실시간으로 특정 정보를 인식해야 하는 연구에도 응용이 가능할 것이다. 향후엔 기존의 데이터 셋을 보완하고 새로운 데이터를 추가하거나 'Background' 클래스 내에 있는 담배 영역을 따로 검출해낼 방법을 모색하여 현 시스템의 필터링 정확도를 높일 것이다. 또한 칼이나 흉기 등의 살상 도구 같이 방송에서 검열 대상이 될 수 있는 객체들의 데이터 셋을 모아 다양한 유해 정보를 필터링 하는 시스템을 개발할 계획이다.

Acknowlegement

"본 연구는 대한민국 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학사업의 연구결과로 수행되었음"(2015-0-00912)

"이 연구는 한국연구재단(NRF-2019R1A2C1010786) 사업의 지원을 받아 수행된 연구임"

4. 참 고 문 헌

[1] "드라마, 영화, 웹툰 작품의 절반 이상에서 담배 및 흡연장면 등장." *한국건강증진개발원*. 2019년 4월 23일 수정, 2019년 11월 12일 접속, https://www.khealth.or.kr/board/view?linkId=999240&menuId=MENU00907.

[2] "담배 권하는 미디어..."드라마·영화·웹툰 절반 이상 흡연장면 등장"," *중앙일보*, 2019년 4월 22일 수정, 2019년 11월 12일 접속, https://news.joins.com/article/23447530.

[3] 방송심의에 관한 규정 제28조(건전성), 방송은 음주,

흡연, 사행행위, 사치 및 낭비 등의 내용을 다룰 때에는 이를 미화하거나 조장하지 않도록 그 표현에 신중을 기하여야 한다. (2015.10.8 전문개정)

[4] 방송심의에 관한 규정 제45조(출연) 제4항, 방송은 어린이와 청소년이 흡연·음주하는 장면을 묘사하여서는 아니되며, 잘못된 흡연·음주 문화를 일반적인 상황으로 인식하지 않도록 그 표현에 신중을 기하여야 한다. <개정 2019. 9. 23.>

[5] "안방극장 흡연장면, '모자이크' 사라져," *The PR*, 2017년 10월 11일 수정, 2019년 11월 12일 접속, http://www.the-pr.co.kr/news/articleView.html?idxno=24894.

[6] 박성대. "영상 콘텐츠 제작을 통한 미디어 교육 - 비선형 편집을 중심으로."
한국정보통신학회논문지. 23(9). pp. 1098, (2019).

[7] 정민욱. "VDSLR을 이용한 디지털영상제작(NLE)에 관한 연구." (국내석사, 세종대학교 문화예술콘텐츠대학원). pp. 42-63, (2013).

[8] "Premiere Pro의 효과 유형", *Adobe*, 2018년 8월 6일 수정, 2019년 11월 24일 접속, https://helpx.adobe.com/kr/premiere-pro/using/effects.html#fixed_effects

[9] "EDIUS PRO 7 새로운 기능", *EDIUS.kr*, 2014년 4월 25일 수정, 2019년 11월 24일 접속, http://edius.kr/main/shop/list.php?ca_id=a090g020

[10] 최희식, 조양현. "딥러닝 기술이 가지는 보안 문제점에 대한 분석." 한국융합학회논문지. Vol 10, 2019(5), pp. 9-16. (2019)

[11] "darknet/LICENCE at master . AlexeyAB/darknet . GitHub", *GitHub*, 2016년 7월 29일 수정, 2019년 12월 16일 접속, https://github.com/AlexeyAB/darknet/blob/master/LICENSE

[12] 이아현, 박수민, 홍정수, "YOLO를 이용한 야생동물 접근 및 피해 예방 시스템 개발."
한국정보과학회 학술발표논문집, 2018(12), pp. 1897-1899. (2018).

[13] Joseph Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR, pp.1-10. (2016).

[14] Joseph Redmon, Ali Farhadi, "YOLOv3: An Incremental Improvement"

[15] 백석영, 박정인, 윤준용, 성우석, "YOLO를 활용한 실시간 교통표지 검출." 한국자동차공학회 부문종합 학술대회, 2019(5), pp. 833-837. (2019).

[16] "Fatkun 일괄 다운로드 이미지", *chrome 웹스토어*, 2019년 11월 16일 수정, 2019년 12월 16일 접속, https://chrome.google.com/webstore/detail/fatkun-batch-download-ima/nnjjahlikiabnchcpehcpkdeckfgnohf?hl=ko

[17] "GitHub - AlexeyAB/Yolo_mark", *GitHub*, 2019년 4월 24일 수정, 2019년 12월 23일 접속, https://github.com/AlexeyAB/Yolo_mark

소프트웨어 소스 코드의 다중 카테고리 분류를 위한 딥러닝 기반 기법

김민하, 심규진, 이찬근
중앙대학교 소프트웨어학부
mina11759@cau.ac.kr, kyshim@gmail.com, cglee@cau.ac.kr

Deep Learning based Technique for Multi-Category Classification of Software Source Code

Min-Ha Kim, Kyoo-Jin Shim, Chan-Gun Lee
Dept of Computer Science and Engineering, Chung-Ang University

요 약

최근 다중 레이블 분류 모델을 이용하여 이미지 및 텍스트를 자동으로 분류해주는 연구가 활발히 진행되고 있다. 우리는 본 논문에서 AST와 API를 이용하여 다중 레이블 방식의 합성곱 신경망 모델을 이용하여 소프트웨어 응용 프로그램의 카테고리를 분류하는 방법을 제안하며, 실험을 통해 본 논문에서 제안하는 모델의 성능을 측정하고 타당성에 대해 논의한다.

1. 서론

소프트웨어 공개저장소에서는 다양한 소프트웨어 프로젝트의 소스 코드, 버그 보고서, 문서 등을 보관하고 있다. 이런 소프트웨어 공개저장소가 널리 보급되고 개발자들은 이 소프트웨어 공개 저장소에서 필요한 소스 코드를 구하거나 소프트웨어 자체를 사용하거나 할 수 있다. 대표적으로 Source Forge에서는 100만 개 이상의 소프트웨어 프로젝트를 호스팅하고 있으며 프로젝트 간에 수많은 정보 교환이 이루어진다. 이 수많은 오픈 소스 프로젝트 가운데 일반 개발자들이 원하는 소스 코드를 찾거나, 오픈 소스 프로젝트 간에 협력하여 작업을 공유할 수 있기 위해 유사한 오픈 소스 프로젝트를 찾는 것은 일일이 손으로 하기엔 매우 어려운 일이다.
그러므로 이런 대규모 소프트웨어 아카이브를 효과적으로 탐색 및 검색하기 위해선 컨텐츠를 분류해야 하며, 이 과정 역시 수동으로 하기엔 인력의 낭비가 심하기 때문에 자동화하는 과정이 필수적이다[1].

이런 소프트웨어 분류 자동화에 있어 Jung은 합성곱 신경망(Convolutional Neural Network)을 이용하여 카테고리를 자동으로 분류하는 연구를 진행하였고, 이 연구는 합성곱 신경망이 다른 머신러닝 기법과 비교해 카테고리 분류에 있어 더 좋은 성능을 낸다는 것을 보여주었다[2].

또한 Linares-Vásquez 등은 소프트웨어 응용 프로그램들의 Application Programming Interface(API) 속성을 이용한 머신러닝 기법으로 소프트웨어의 분류 자동화하는 접근법을 제시하여 API를 이용한 접근법은 API의 숫자가 소스 코드 용어보다 적더라도 소스 코드 용어를 사용한 분류법과 비슷한 정확도를 보여주고, 안정적인 결과를 낼 수 있다는 것을 보여주었다[3].

우리는 이전 연구에서 [2]와 [3]의 연구에서 소스 코드의 제한된 속성만을 이용한다는 한계점을 극복하기 위해 API와 Abstract Syntax Tree(AST) 임베딩을 이용한 CNN 심층 신경망 모델을 제안했고 CNN모델을 사용했을 때보다 AST-API-CNN모델을 사용했을 때 성능이 향상됨을 보였다[4]. 하지만 이전 연구에서는 1개의 소프트웨어에 1개의 분류 레이블만을 사용하여 현실적이지 못했다는 한계점이 있었다[4]. 실제 다수의 소프트웨어는 1가지 분류 레이블만으로 분류할 수 없으며 여러 가지 기능과 특징을 가진 소프트웨어가 많다. 이번 연구에서는 소프트웨어의 이런 특성을 위해 다중 레이블 분류법을 이용한 소프트웨어 분류를 제안한다.

2. 관련 연구 및 기술

본 절에서는 본 논문에서 제안하는 모델과 관련된 연구와 기술에 대해 소개한다.

2.1 추상 구문 트리 (Abstract Syntax Tree)

Abstract Syntax Tree (AST)는 프로그래밍 언어로 쓰인 소스 코드의 구문을 분석하여 각 프로그래밍 언어의 문법에 따라 소스 코드의 구조를 표시하는 계층적 프로그램 표현이며, 구조는 그림 1과 같다. 일반적으로 컴파일러를 통해 일반적으로 코드의 AST를 추출할 수

있으며, 두 가지 단계를 통해 코드를 기반으로 AST를 생성하며, 단계에 대한 내용은 표1과 같다.

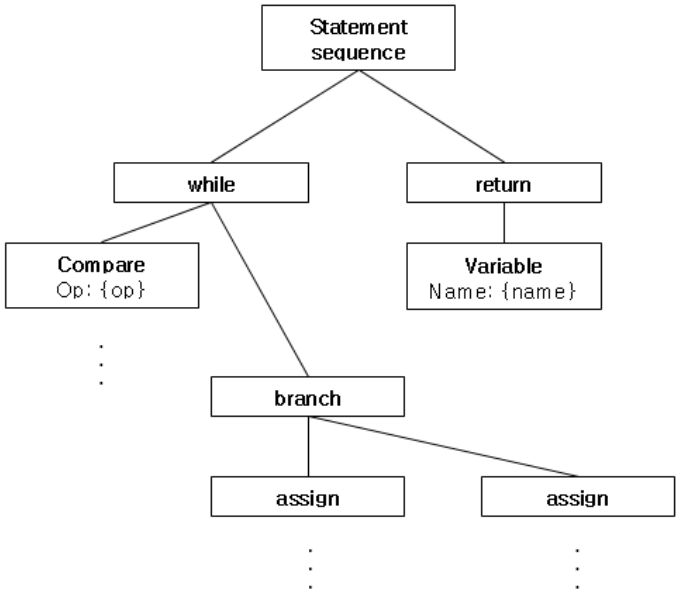


그림 1 추상 구문 트리 구조

표 1 AST 추출 단계

단계	내용
어휘 분석	프로그램마다 정의된 규칙을 따라 소스 코드를 읽고 공백이나 주석 등을 제거하고 토큰으로 결합한다.
구문 분석	토큰 목록을 가져와 언어 구문을 검증하고 트리 구조로 변환한다.

AST에서 구문은 실제 소스 코드에 나타나는 모든 세부사항을 나타내지 않고 개념적으로 소스 코드를 설명하게 된다. 본 논문에서는 Java Parser를 이용하여 AST와 API를 동시에 이용하여 소스 코드의 특징을 추출하였다.

2. 2 다중 레이블 분류 (Multi-Label Classification)

비교 연구인 [4]에서 다중 클래스 분류(Multi-Class Classification)을 이용하여 소프트웨어 카테고리를 분류하고 이에 대한 성과가 있었으나, 우리는 현실의 소프트웨어 응용프로그램의 카테고리 분류와 괴리가 있다고 생각하였다. 그래서 다중 레이블 분류(MLC)를 이용하면 소프트웨어 응용프로그램의 분류가 좀 더 현실적으로 이루어질 수 있다고 판단하였다. MLC를 위한 방법을 두 가지 범주로 분류할 수 있는데, 두 가지는 다음과 같다.

- 문제 변환 방법
- 알고리즘 적응법

이 중 우리는 문제 변환 방법 중 하나인 다중 클래스 분류 문제로 변환하는 방법인 Label Power set(LP)를 선택하였다. LP 변환은 학습 데이터 집합에 존재하는 모든 레이블 조합에 대해 하나의 이진 분류기를 만든다. 예를 들어 가능한 레이블이 A, B, C인 경우, LP 표현은 이진 분류를 통해 [0, 0, 0], [1, 0, 0], [0, 1, 0] 등과 같이 나온다. [1, 0, 1]인 경우, B의 레이블이 없는 경우를 나타낸다.

2. 3 정밀도 (Precision)

정밀도는 모델을 통한 예측값이 'True'라고 레이블을 분류한 것중에서 실제값이 'True'인 것의 비율이다. 이는 다중 레이블 분류 방식에 대한 적합한 평가 방식으로, 본래의 수식은 (1)과 같다.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

본 논문에서 제안하는 다중 레이블 예측 모델은 전체 데이터에 대해 (1)수식을 사용할 수 없다고 판단하였다. 이에 근거하여 하나의 인스턴스에 대한 정밀도를 계산한 뒤 각각의 값을 전부 더해 전체 데이터의 개수로 나누는 방식이 더 타당하다고 판단된다. 이에 따라 본 논문에서 사용한 다중 레이블 정밀도의 수식은 (2)와 같다.

$$MLabel - Precision = \frac{1}{|N|}\sum_{i=1}^{|N|}\frac{TP_i}{TP_i + FP_i} \quad (2)$$

2. 4 재현율 (Recall)

재현율은 실제 레이블이 'True'인 것 중에서 모델을 통한 예측값이 'True'인 것의 비율이다. 정밀도와 마찬가지로 다중 레이블 분류 방식에 대한 적합한 평가방식이며, 본래의 수식은 (3)과 같다.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

정밀도와 마찬가지로 재현율 또한 다중 레이블 모델에서 전체 데이터에 대해 (3)의 식을 사용할 수

없으므로 하나의 인스턴스에 대한 재현율을 계산한 뒤, 모두 더하여 전체 데이터의 개수로 나누는 방식을 사용하였다. 본 논문에서 사용한 재현율의 수식은 (4)와 같다.

$$MLabel-Recall=\frac{1}{|N|}\sum_{i=1}^{|N|}\frac{TP_i}{TP_i+FN_i} \quad (4)$$

2. 5 자카드 유사도 (Jaccard Similarity)

자카드 유사도는 두 유한 집합 사이의 유사도와 다양성을 측정하는 데에 사용되는 통계학적 측정 방법의 하나이며[5], 본래의 수식은 (5)와 같다.

$$Jaccard\ Similarity=\frac{\left|y^{true}\cap y^{pred}\right|}{\left|y^{true}\cup y^{pred}\right|} \quad (5)$$

하지만 MLC는 레이블의 값을 옳거나 옳지 않다고 나눌 수 없기 때문에 다중 레이블의 집합 간의 유사도를 계산하기 위해 자카드 유사도 또한 정밀도와 재현율에 대한 수식과 같은 방식으로 각 인스턴스에 대한 자카드 유사도를 계산한 뒤에 모두 더하여 전체 데이터의 개수로 나누어 주는 방식을 사용하였으며, 이에대한 수식은 (6)과 같다.

$$MLabel-Jaccard\ Similarity=\frac{1}{|N|}\sum_{i=1}^{|N|}\frac{\left|y_i^{true}\cap y_i^{pred}\right|}{\left|y_i^{true}\cup y_i^{pred}\right|} \quad (6)$$

2. 6 해밍 손실량 (Hamming loss)

해밍 손실량은 정확하게 분류된 데이터에 대해 계산을 하지 않고 예측된 데이터에서 발생한 손실량을 계산한다. 즉, 원래의 레이블과 예측된 레이블 사이에서 '배타적 논리합 (XOR)' 연산을 하고 레이블 집합 전체의 평균을 계산하며, 수식은 (7)와 같다.

$$Hamming\ loss=\frac{1}{|N|\cdot|L|}\sum_{i=1}^{|N|}\sum_{j=1}^{|L|}xor\left(y_{ij}^{T},y_{ij}^{P}\right) \quad (7)$$

3. 실험 내용

본 절에서는 본 논문에서 사용할 데이터에 대한 수집 및 정제 과정과 자바 언어로 작성된 응용프로그램의 코드에서 API와 AST를 추출한 뒤, 추출된 단어들을 각각 Word2Vec 기법을 이용하여 벡터화한 후, 심층 신경망 알고리즘을 사용하여 소프트웨어 응용 프로그램의 카테고리를 다중 레이블로 분류하는 모델을 소개한다.

3. 1 데이터 수집 및 정제

데이터를 수집하기 위해 소프트웨어 공개저장소인 Sourceforge.net에서 Java 언어로 작성된 소프트웨어 응용 프로그램을 모두 내려받았으나, 우리는 데이터 추출을 위한 코드가 존재하는 소프트웨어 응용 프로그램의 개수가 너무 적은 카테고리의 경우 데이터의 부족으로 학습이 잘 되지 않을 것으로 판단하였고, 이를 바탕으로 응용 프로그램의 개수가 35개 미만인 응용 프로그램들의 하위 카테고리는 모두 제거하였다.

표 2 데이터 수집 결과

카테고리		개수
Software Development	Build-tools	47
	Code-generators	43
	Compilers	45
	Frameworks	115
	Interpreters	84
	Object Brokering	42
	Libraries	36
	User-Interface	55
Internet	WWW/HTTP	148
Scientific Engineering	Bio-Informatics	64
	Human Machine Interfaces	47
	Mathematics	99
	Visualization	111
Business	Enterprise	54
	Financial	65
	Scheduling	51
System	Networking	105
	Storage	48
	System-administration	79
Communication	Chat	36
	Email	74
	File-Sharing	83
	Telephony	39
Multimedia	Cataloguing	35
Games	Borad Games	35
합계		**1640**

최종적으로 [4]에서 선별한 카테고리와는 달리 본 논문에서는 8개의 상위 카테고리의 하위에 있는 25개의 카테고리로 확장하였다. 그리고 운영체제와 같은 세부적인 조건은 [4]에서 선택한 기준과 같이하였고, 최종적으로 수집한 데이터는 표 2과 같다. 표 2는 sourceforge.net에서 분류한 카테고리별로 내려받았으며, 실제로 해당 카테고리 내에 포함 되어있는 소프트웨어 응용프로그램은 표 2에 표시된 카테고리를 포함하여 해당 응용 프로그램을 업로드한 프로그래머가 명시한 1개 이상의 카테고리를 추가적으로 지니고 있다.

추출을 진행한 후, 소프트웨어 응용프로그램에 따라 AST와 API 파일의 개수가 너무 많은 것들은 메모리 과다사용으로 학습이 이루어지지 않는 경우가 생길 수 있기 때문에 추출된 AST와 API 파일의 개수가 1000개 이상인 응용프로그램들은 모두 제거하였으며, 추출 과정은 그림 2와 같다.

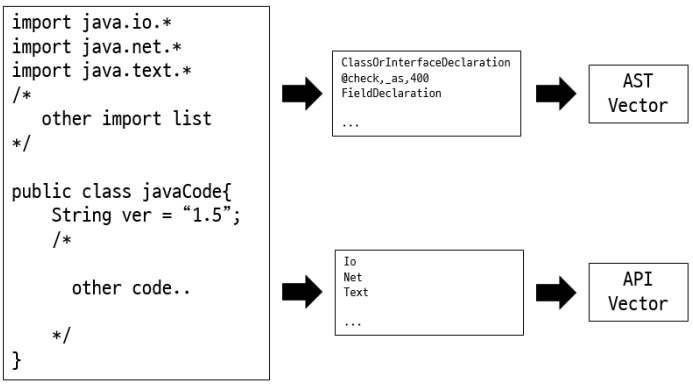


그림 2 AST와 API 추출 과정

그림 2와 같이 추출한 AST와 API 벡터들의 각 원소들에 대해 단어 표기법(예: Camel 표기법, Snake 표기법)에 따라 단어 집합으로 분리한 뒤, 대문자를 소문자로 변경하였다. 그리고 2글자 미만의 단어는 의미가 없거나 약어로 판단하여 실험의 복잡도를 줄이기 위해 모두 제거하여 정제하였다.

정제한 데이터들은 Word2Vec 기법을 이용하여 벡터화를 진행하였으며, 구글 뉴스에서 발췌한 300차원의 벡터 파일[6]을 이용하였다. 그리고 입력 데이터로 사용되는 단어 벡터들의 차원이 같아야 하기 때문에 더미 벡터를 추가하여 부족한 차원을 채웠다.

3. 2 다중 레이블 모델 설계

응용 프로그램들의 카테고리를 다중 레이블로 분류해주기 위해 합성곱 신경망 모델을 이용하여 구성하였다. 벡터화를 진행한 AST와 API에 대한 벡터를 각각 입력층에 넣는 다중 입력 방식으로 층(layer)을 구성하였으며, 본 논문에서 제안하는 모델은 그림 3와 같다.

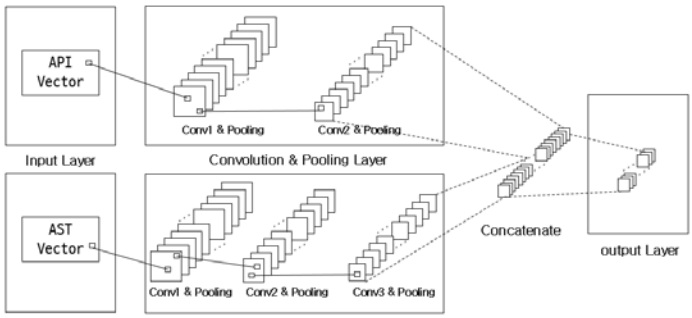


그림 3 본 논문에서 제안하는 합성곱 신경망 모델

합성곱(Convolution) 층에서 API의 필터 개수는 512개를 사용하였고, AST의 필터 개수는 256개를 사용하였다. 각각 필터의 크기는 2, 3과 3, 4, 5로 사용하였고, 마지막으로 두 층을 합하여 출력층으로 전달할 수 있도록 구성하였다.

4. 실험 결과

본 절에서는 본 논문에서 제안하는 모델에 대해 다중 레이블 분류에 대해 올바른 평가방식을 도입하여 해당 모델을 평가하고 분석한다.

실험을 시작하기에 앞서 미리 수집한 응용 프로그램을 9:1로 학습 데이터와 테스트 데이터로 구분하여 실험을 진행하였다. 평가 방식으로는 다중 레이블 분류에 적합한 재현율(Recall), 정밀도(Precision), F1-Measure, 자카드 유사도 그리고 해밍 손실량을 이용하며, 실험 결과는 표 3과 같다.

표 3 모델의 성능 평가 결과 [단위: %p]

평가 방식	결과
Precision	98.3
Recall	48.1
F1-Measure	63.4
Jaccard Similarity	98.5
Hamming Loss	6.4

표 3에서 알 수 있듯이 자카드 유사도가 98.1%p, 정밀도가 98.6p% 그리고 해밍 손실량이 6.4%p로 높은 성능을 보였지만 재현율과 F1-Measure 평가 방식으로는 다른 기준에 비해 낮은 결과를 보였다.

이러한 결과를 보았을 때, False Negative(FN)의 값이 높아 재현율이 낮은 것으로 판단되며, 레이블의 개수가 상대적으로 적은 것들을 제거하여 레이블의 수를 줄여 FN의 값을 줄임으로써 재현율을 향상 시킬 수 있고 이에 따라 F1-Measure도 향상될 것으로 보인다. 실제로 다중 레이블을 똑같이 예측한 비율은 높지 않으나, 실제값과 예측값을 비교하였을 때, 두 집합 사이에 겹치는 카테고리가 많다. 이는 다중 레이블 분류 방식이 실생활

의 카테고리를 예측하는 데에 타당한 것으로 생각될 수 있다.

5. 결론 및 향후 연구

본 논문은 응용 프로그램의 카테고리 분류에 대한 타당성의 관점에서 응용 프로그램을 자동으로 분류해주는 다중 레이블 분류 모델을 제안하였다. 이를 위해 [4]에서 레이블을 부여하는 방식과는 다르게 각각 프로그램마다 다중 레이블을 부여하였고, 그 결과 Recall과 F1-Measure 평가 기준을 제외한 다른 평가 기준에서 성능이 평이하게 나온 것을 볼 수 있었다. 이에 근거하여 본 논문에서 제안한 모델이 실생활에서 카테고리를 분류하는 모델로 적합하다고 판단하였다. 하지만 [2], [4]와 같이 본 논문에서 제안하는 모델로는 데이터가 자바 언어로만 작성된 것으로 한정되어 있기 때문에, 다른 프로그래밍 언어로 구현된 소프트웨어 응용 프로그램에 대해서는 적용되기 어렵다는 한계점을 가진다.

향후 우리는 다른 소프트웨어 공개저장소를 찾아 더 많은 카테고리를 확보하고 그에 대한 소스 코드가 있는 응용 프로그램을 수집하고자 하며, 작성 언어와는 관계없이 자동으로 분류해주는 모듈로 확장하여 사용자들이 카테고리를 탐색하기 위한 효율성을 향상하고 타당성을 부여하고자 한다.

Acknowledgements

본 연구는 한국연구재단 기초연구사업 (과제번호: NRF-2017R1E1A1A01075803), 한국원자력연구원 원자력 ICT 안전성 검증체계 구축 및 운영과제(과제번호: 524320-18), 산업통상자원부 및 산업기술평가관리원(KEIT) 산업기술혁신사업 (과제번호:20003580)의 지원을 받았음.

참고문헌

[1] S. Kawaguchi, P. K. Garg, M. Matsushita, and K. Inoue, “MUDABlue: An automatic categorization system for Open Source repositories”, Journal of Systems and Software 79(7): 939-953, 2006.

[2] S. Jeong, “Improving Performance of Automatic Software Categorization using Deep Learning”, Master Thesis, Chung-Ang University, 2019.

[3] M. Linares-Vásquez, C. McMillan, D. Poshyvanyk, M. Grechanik, “On Using Machine Learning to Automatically Classify Software Applications into Domain Categories”, Empirical Software Engineering 19: 582-618, 2014.

[4] M.-H. Kim, M.-S. Lee, and C.-G. Lee, “Automated Classification of Software Category using API and AST Embeddings”, Proc. of Korea Multimedia Society Conference, 2019.

[5] V. Labatut and H. Cherifi, “Evaluation of performance measures for classifiers comparison”, Ubiquitous Computing and Communication Journal 6: 21-34, 2011.

[6]Word2Vec, https://code.google.com/archive/p/word2vec

스테이트먼트 유형 정보를 활용한 옳은 패치 생성률 증대 기법

허진석○, 정호현*, 이은석
성균관대학교 전자전기공학부, 성균관대학교 전자전기컴퓨터공학과*
{mrhjs225, jeonghh89, leees}@skku.edu

A technique to increase rate of correct patches generation using statement type information

Jinseok Heo○, Hohyeon Jeong*, Eunseok Lee
Department of Electronic and Electrical Engineering, Sungkyunkwan University
Department of Electrical and Computer Engineering, Sungkyunkwan University*

요 약

최근 디버깅(Debugging) 작업의 효율을 높이기 위해 자동으로 패치를 생성하는 프로그램 자동 수정 기법(APR, Automated Program Repair)이 연구되고 있다. APR의 한 종류인 탐사기반(Search-based) APR의 경우 탐사 영역(Search space)에 따라 옳은 패치(Correct patch)의 생성률이 영향을 받는다. 본 논문에서는 옳은 패치의 생성률 향상을 위해 SeeType을 제안한다. SeeType은 과거의 패치로부터 AST(AST, Abstract Syntax Tree)와 AST 노드(Node) 유형이 담긴 템플릿(template)을 생성한다. 그리고 식별된 결함 위치의 스테이트먼트 유형(Statement type)을 분석하고, 사람이 작성한 패치에서 패치 전 결함 위치의 스테이트먼트 유형이 분석 결과와 일치하는 템플릿을 선별하여 수정 템플릿을 적용한다. 이를 통해서 사람이 작성한 패치와 유사한 패치를 자동 생성 할 수 있으며, 따라서 옳은 패치 생성률 향상 및 디버깅 효율 향상을 기대할 수 있다.

1. 서 론

소프트웨어 유지보수는 개발자의 수동적인 작업으로 인해 비용이 많이 소모되는 작업이다. 이러한 부분을 개선하는 방법으로 결함 추적부터 패치 생성까지 자동으로 진행되는 APR이 활발히 연구되고 있다[1, 2].

그러나 APR 기법의 하나인 탐사 기반 APR 기법이 생성한 패치의 질이 떨어진다는 지적이 존재한다[3, 4]. 탐사 기반 APR의 성능은 탐사 영역에 기반하므로[5], 이러한 점을 극복하기 위해서는 탐사 영역을 잘 구성해야 한다. 또한 [6]은 단순히 탐사 영역을 늘릴 경우 옳은 패치뿐만 아니라 테스트 케이스는 통과하나 개발자가 수용할 수 없는 패치인 그럴듯한 패치(plausible patch)가 증가한다고 밝혔다. 따라서 옳은 패치를 생성하기 위해서는 옳은 패치가 존재하는 질 높은 탐사 영역만을 추출하여 이를 기반으로 패치를 생성해야만 한다.

기존부터 옳은 패치와 관련하여 탐사영역을 어떻게 구성할지에 대한 연구가 진행되고 있다. 연구 [7]은 스테이트먼트와 같은 큰 단위(coarse-grained granularity)의 정보로는 많은 버그에 대해 옳은 패치를 생성하기 어렵고, AST 노드와 같은 작은 단위(fine-grained granularity)의 정보로는 옳은 패치를 생성할 수 있으나 효율적이지 못한다고 밝혔다. 이를 기반으로 본 논문에서는 옳은 패치의 생성률 향상을 목표로 기존의 연구[8, 9, 10]에서 사용하지 않은 작은 단위 정보인 사람이 작성한 패치의 스테이트먼트 AST 노드 유형 정보를 활용한 SeeType (**S**earch spac**e** r**e**duction using Statement **type**) 기법을 제안한다.

SeeType은 크게 두 단계의 처리 과정을 가진다: 1) 사람이 작성한 패치 정보 수집 및 분석을 통한 패치 생성용 템플릿 자동 작성, 2) 식별된 결함 위치의 스테이트먼트 유형에 따른 패치 생성 템플릿 적용 및 평가.

패치 생성에 사용될 템플릿을 생성하는 과정에서는 먼저 오픈 소스 소프트웨어 저장소의 프로젝트에서 사람이 작성한 패치를 수집한다. 해당 패치들의 패치 전, 후 스테이트먼트에 대해 AST 노드 유형 분석을 진행한 후, 패치 전, 후의 AST 노드 유형 쌍과 AST를 템플릿으로 취급하여 추출한다.

이후 버그가 발생한 프로젝트에 대해 APR 과정을 진행한다. 먼저, 버기 코드(Buggy code)와 테스트 케이스를 통해 결함 추적을 한다. 결함 추적 결과로 나온 의심스러운 스테이트먼트의 AST 노드 유형 분석을 진행한다. 미리 만들어 놓은 템플릿 중 의심스러운 스테이트먼트의 AST 노드 유형과 패치 전 AST 노드 유형이 같은 템플릿만 추출한다. 추출한 템플릿을 통해 후보 패치를 생성하고 평가 과정을 거쳐 유효한 패치(Valid patch)를 생성한다.

SeeType에서의 탐사영역은 패치 생성을 위해 사용되는 모든 템플릿을 의미한다. 패치 생성 시 모든 템플릿에 대하여 탐색하는 것이 아닌, 결함 추적의

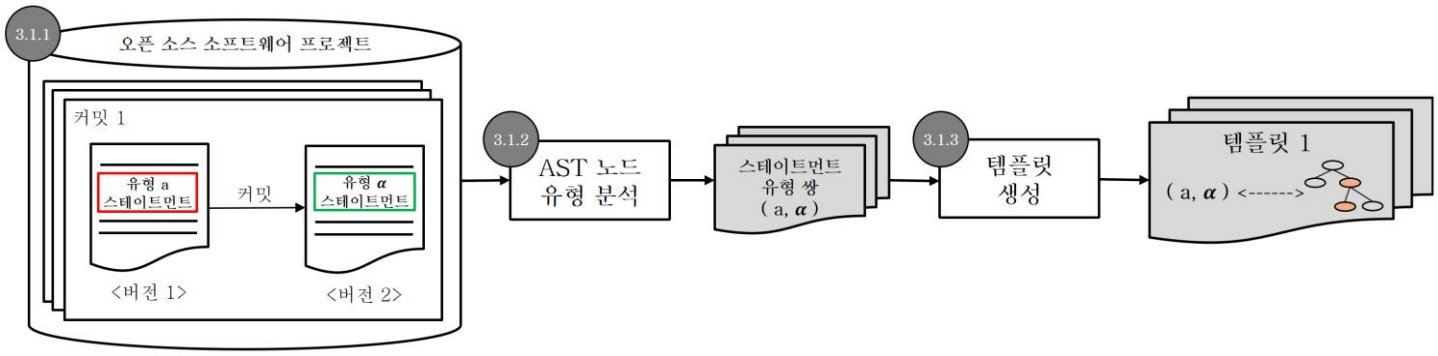


그림 1 SeeType의 템플릿 생성 프로세스

결과인 의심스러운 스테이트먼트의 AST 노드 유형과 동일한 유형을 가진 템플릿에 대해서만 패치 생성 및 변이(mutation)과정을 적용하므로 기존 APR 기법보다 옳은 패치 생성을 위한 탐사 효율이 높아질 수 있다. 또한 SeeType은 사람이 생성한 패치를 활용할 뿐만 아니라 해당 패치의 AST 유형 정보 또한 활용하기 때문에 기존보다 더욱더 많은 옳은 패치를 생성할 수 있다.

이 논문의 기여점은 다음과 같다.

- 기존의 사람이 작성한 패치에 대해 AST 노드 유형 정보를 활용한 새로운 탐사 영역 축소 기법 제안.
- 작은 단위 정보인 AST 기반 패치 생성으로 인한 높은 옳은 패치 생성률 기대.

이어지는 2장에서는 관련 연구를 설명하고 3장에서 본 논문에서 제안하는 SeeType 기법을 자세하게 설명한다. 4장에서는 실험 방법에 대해 논하고, 마지막으로 5장에서는 향후 연구와 함께 본 논문을 마무리 짓는다.

2. 관련 연구

2.1 과거 패치 이력 관련 연구

APR의 패치 생성 과정을 위해 과거에 작성 및 적용된 패치의 이력을 분석 및 활용하는 연구가 진행되었다. [10]은 사람이 작성한 패치를 수동으로 분석한 후 발견된 패턴으로 템플릿을 만들어 기존의 옳은 패치를 생성하지 못한 타깃 프로젝트(Target project)에 대해서 새로운 옳은 패치를 생성하였다. [8]은 오픈 소스 소프트웨어 저장소의 프로젝트에서 사람이 작성한 패치를 통해 옳은 패치의 특징을 추출하는 기계학습 모델을 생성했다. 생성된 기계학습 모델은 후보 패치를 우선 순위화하여 옳은 패치의 순위를 높이는 데 사용하여 이전 연구보다 더욱더 많은 옳은 패치를 생성하였다. 위의 연구를 통해 사람이 작성한 패치 이력에서 더욱더 많은 옳은 패치 생성이 가능함을 알 수 있다.

[11]은 과거 패치 이력을 코드 구조 관점에서 분석하였다. 과거 패치 간에 같은 프로젝트일 경우 약 90%의 AST 노드가 중복되고, 다른 프로젝트일 경우 약 40%가 겹치는 것을 확인했다. 따라서 타깃 프로젝트와 유사한 AST 및 코드 구조를 가진 프로젝트를 사용한다면 비슷한 과거 패치를 활용할 수 있기 때문에 효과적으로 옳은 패치를 생성 할 수 있다.

2.2 AST 노드 유형 정보 활용

기존부터 옳은 패치 생성을 위한 탐사 영역 구성 관련 연구가 진행되고 있다. [8]은 오픈 소스 소프트웨어 저장소의 사람이 작성한 패치를 통해 탐사 영역을 구성함으로써 옳은 패치를 생성했다. [9]은 프로그램 동작(Operation)과 패치 재료(patch ingredient)에 대한 경험적 규칙을 세워 규칙 기반 탐사 영역 축소 과정을 거친 후에 패치를 생성했다. 위의 연구들은 과거의 패치들을 기반으로 경험적인 규칙, 템플릿을 만들어 탐사 영역을 필터링하였다. 하지만 사람이 작성한 패치의 분석 정보 중 AST 노드 유형 정보를 활용하지는 않았다.

사람이 작성한 패치에 대한 분석을 내놓은 연구 [12]는 오픈 소스 소프트웨어 저장소에서 버그가 발생하는 AST 노드 유형 분석 시 스테이트먼트 관련 유형이 약 73%를 차지하고 있음을 밝혔다. 해당 73%를 전체로 두고 유형의 종류에 대해 분석 시 88%의 버그의 발생은 22가지의 스테이트먼트 유형 중 5가지에 의해서만 발생함을 밝혔다. 더불어 연구 [13]에서도 자바 기반의 패치를 분석하였는데 패치 간 중복되는 스테이트먼트 유형 및 패치 패턴이 존재한다고 밝혔다. 따라서 기존 패치에서 추출된 모든 템플릿을 탐사하지 않고 의심스러운 스테이트먼트의 AST 유형을 통해 필터링 된 템플릿으로 패치를 생성하는 것은 기존의 APR 기법보다 높은 효율을 얻을 수 있다.

3. SeeType : Search space reduction using Statement type

SeeType의 과정은 템플릿 생성(3.1절)과 해당 템플릿을 활용한 APR 과정(3.2절)으로 나뉜다. 템플릿 생성 과정에서는 과거의 패치 이력을 통해 템플릿을 생성하고, 템플릿을 활용한 APR 과정에서는 앞의 과정을 통해 생성된 템플릿을 활용하여 패치 생성 및 평가가 이루어진다.

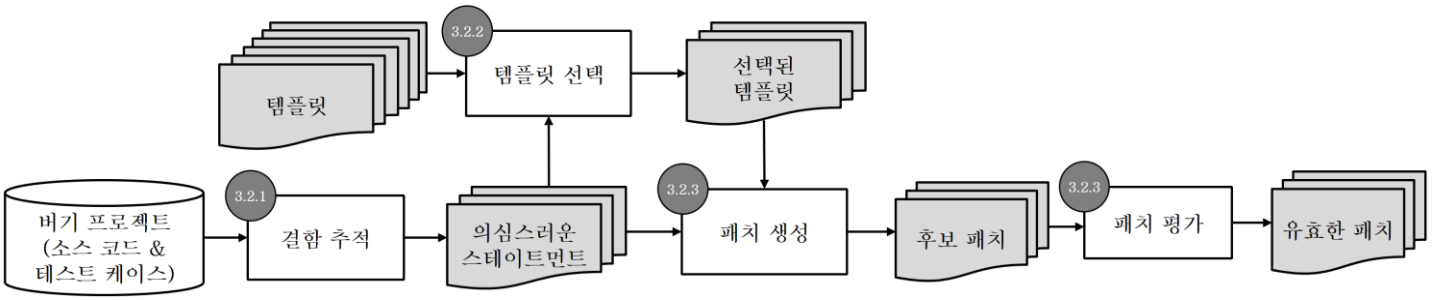
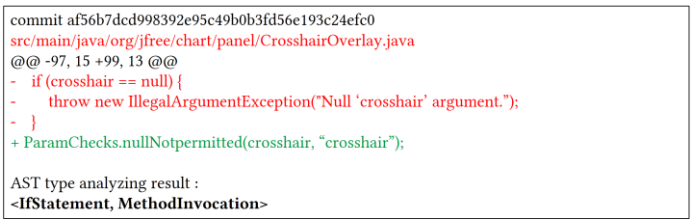
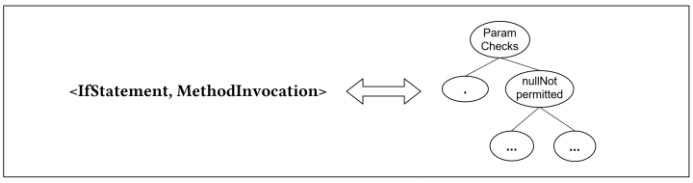


그림 2 SeeType의 템플릿 활용 APR 프로세스



a) 스테이트먼트 분석 결과



b) 템플릿 생성 예시

그림 3 템플릿 생성 과정 예시

3.1 템플릿 생성

그림 1은 SeeType의 템플릿 생성에 대한 프로세스를 보여주고 있다. 먼저 오픈 소스 소프트웨어 저장소의 프로젝트에서 사람이 작성한 패치를 수집한다(3.1.1절).

수집된 패치를 AST 형태로 변환 후, AST 노드 유형 분석을 한다(3.1.2절). 분석 결과로 패치 전, 후 스테이트먼트의 AST 노드 유형이 각각 나오게 되면 해당 쌍을 템플릿으로 취급하여 저장한다(3.1.3절).

3.1.1 과거 패치 이력 수집

과거 패치 이력 수집 단계에서는 템플릿을 생성하는데 필요한 과거의 패치들을 수집하게 된다. 우선 패치 생성 시 최대한 유사한 정보를 활용하기 위해 오픈 소스 소프트웨어 저장소 중 하나인 Github[14]에서 타깃 프로젝트와 유사한 프로젝트를 대상으로 수집한다. 이때 타깃 프로젝트와 동일한 프로그래밍 언어와 토픽(topic)을 가진 프로젝트를 유사한 프로젝트로 취급한다.

프로젝트를 선별한 이후, 패치를 수집하기 위해 타깃 프로젝트 및 선별된 프로젝트의 커밋(Commit)을 탐사한다. 커밋 메시지 중 패치 관련 키워드(예-fix, repair)를 포함하고 있는 경우 패치로 간주하여 선별한다.

3.1.2 스테이트먼트 유형 분석

스테이트먼트 유형 분석 단계에서는 템플릿 생성을 위해 패치를 AST로 변환 후 AST 노드 유형을 분석한다. AST 분석을 위해 Gumtree[15]라는 도구를 활용하는데, Gumtree는 코드를 AST로 변환하여 각각의 AST 노드에 대한 유형 분석 결과를 결과로 도출할 수 있다. AST 노드 유형에 대한 정의는 이클립스(Eclipse) 자바 문서에 미리 정의된 것을 사용한다[16] (정의의 일부를 부록에 첨부함).

기존 패치의 스테이트먼트 분석 시 패치 이전의 스테이트먼트, 패치 이후의 스테이트먼트를 각각 분석한다. 그리고 그림3의 a) 와 같이 **<패치 이전의 스테이트먼트 AST 노드 유형, 패치 이후의 스테이트먼트 AST 노드 유형>**의 형태로 쌍을 만든다. 이 쌍을 기반으로 템플릿을 도출하게 된다.

3.1.3 템플릿 생성

이 단계에서는 이전 단계(3.1.2절)의 결과를 가지고 템플릿을 생성하는 단계이다. 템플릿 활용 APR 과정에서 패치 생성 시 기존의 패치 정보 중 패치 이후의 정보를 가지고 패치를 생성한다. 따라서 패치 이후 코드의 AST를 템플릿과 연결 지어 최종적으로 템플릿을 생성한다. 예를 들어 그림 3의 b)처럼 스테이트먼트 유형 분석 단계의 결과가 <IfStatement, MethodInvocation>일 경우 **MethodInvocation의 코드 AST**와 연결 지어 하나의 템플릿으로 생각하게 된다.

자주 등장하는 패치의 유형을 패치 생성 시에 반영하기 위해 각각의 템플릿에 빈도수를 산출한다. 빈도수는 해당 템플릿이 수집한 패치의 목록에 등장한 횟수를 의미한다.

3.2 템플릿 활용 APR 과정

그림 2는 템플릿을 활용한 SeeType의 버그 수정 과정을 보여주고 있다. 결함 추적 과정(3.2.1절)에서는 타깃 프로젝트와 테스트 케이스를 이용해 의심되는 스테이트먼트를 도출한다. 이후 의심스러운 스테이트먼트의 AST 노드 유형 분석을 통해 템플릿을 선별한다(3.2.2절). 선별한 템플릿을 통해 패치를 생성하고 마지막으로 패치 평가를 거쳐 유효한 패치를 생성한다(3.2.3절).

3.2.1 결함 추적

결함 추적 단계에서는 타깃 프로젝트와 테스트 케이스를 통해 의심스러운 스테이트먼트를 도출한다. 결함 추적 알고리즘으로는 스펙트럼 기반 결함 추적 알고리즘 중 Tarantula[17]를 사용하되, 대상으로 하는 타깃 프로젝트에 따라 최적의 성능을 내기 위해 Ochiai[18], Jaccard[19]와 같은 다양한 알고리즘을 적용할 수 있다.

```
if (dataset != null) {

AST type analyzing result : IfStatement
```

a) 의심스러운 스테이트먼트 분석 결과

```
Buggy Statement Type : IfStatement
Template filtering result:
<IfStatement, MethodInvocation>
<IfStatement, VariableDeclarationFragment >
<IfStatement, ThisExpression>
...
```

b) 템플릿 필터링 예시

그림 4 템플릿 선택 과정 예시

표 1 Defects4j 실험 데이터 정보

프로젝트(식별자)	버그 수	라인 수 (단위: 천)	테스트 케이스
JfreeChart(Chart)	26	96	2,205
Closure compiler(Closure)	133	90	7,927
Apache commons-lang(Lang)	65	22	2,245
Apache commons-math(Math)	106	85	3,602
Mockito(Mockito)	38	11	1,457
Joda-Time(Time)	27	28	4,130
총합	**395**	**332**	**21,566**

3.2.2 템플릿 선택

템플릿 선택 단계에서는 패치 생성에 필요한 템플릿을 선별하여 탐사 영역을 줄이는 단계이다. 결함 추적 단계의 결과인 의심스러운 스테이트먼트에 대해 AST 노드 유형 분석을 진행하게 되는데 그림4의 a)와 같이 단일 결과로 나온다. 이후 템플릿 중 해당 AST 노드 유형을 통해 템플릿을 필터링해야 하는데, 결함 추적의 결과로 나온 의심스러운 스테이트먼트는 과거 패치와 비교할 때 패치 이전의 스테이트먼트와 상응하는 자리이다. 따라서 의심스러운 스테이트먼트의 AST 노드 유형과 패치 이전의 스테이트먼트의 AST 노드 유형이 같은 템플릿만 필터링한다. 예를 들어 그림4의 b)와 같이 의심스러운 스테이트먼트의 유형이 IfStatement라면 <IfStatement, MethodInvocation>, <IfStatement, VariableDeclarationFragment>와 같은 템플릿만 필터링 되게 된다.

3.2.3 패치 생성 및 평가

패치 생성 단계에서는 이전 단계에서 필터링 된 템플릿을 통해 패치를 생성하게 된다. 템플릿에 존재하는 빈도수 정보를 바탕으로 빈도수가 높은 템플릿부터 활용하여 패치를 생성한다. 구문적(Syntactic) 부분 확인을 위해 선택된 템플릿의 AST와 의심스러운 스테이트먼트의 AST에서 변수, 메소드 정보를 추출한다. 각각 추출된 정보를 통해 구문적 부분을 확인하고, 구문적 부분이 일치하면 템플릿을 활용하여 AST를 수정해 패치를 생성한다.

템플릿을 통해 후보 패치가 생성되지 않을 경우 변이 과정[20]을 거쳐 지정된 타임아웃(Timeout)까지 패치 생성을 시도한다.

생성된 후보 패치는 패치 평가과정을 통해 준비된 모든 테스트 케이스를 통과했을 때 유효한 패치로 판단한다. 그렇지 못할 경우 유효하지 못한 패치로 간주한다. 생성한 유효 패치의 개수가 지정한 수에 도달할 때까지 다음 순위의 템플릿을 사용하여 패치를 생성한다.

3.3 실행 과정

SeeType은 실험 시 조정할 수 있는 5개의 파라미터가 존재한다. 5개의 파라미터에는 결함 추적의 최대 결과 개수 N, 템플릿의 최대 개수 M, 유효 패치의 최대 개수 L, 타임아웃 관련 매개변수 T1, T2(단, T2 > T1)가 있다.

SeeType의 프로세스는 결함 추적 결과의 1순위부터 N 순위까지 순차적으로 진행된다. 결함 추적 결과의 1 순위부터 개별 추적 결과에 대해 템플릿을 최대 M개까지 추출한 후 패치를 생성한다. 템플릿들은 빈도수에 의해 우선순위가 부여되며, 1순위부터 패치를 생성한다. 패치를 생성하지 못했을 시 타임아웃 시간 T1 동안 변이과정을 수행하며, 타임아웃 이후에는 다음 순위의 템플릿으로 진행한다. M 순위까지의 템플릿 결과 활용 후 그다음 순위의 결함 추적 결과를 통해 위의 과정을 반복한다. 생성한 유효 패치가 L개가 되거나 전체 타임아웃 시간 T2에 도달할 경우 과정을 중단한다.

4. 실험계획

4.1 실험 데이터

4.1.1 벤치마크

평가에 사용되는 벤치마크는 자바 언어 기반 프로젝트들의 버그 및 테스트케이스를 모아놓은 Defects4j[21] 1.2.0버전을 사용한다.

표1의 항목 중 버그 수는 해당 프로젝트가 가지고 있는 총 버그 개수를 의미하고, 라인(line) 수는 프로그램의 크기를 천개 단위의 라인 개수로 나타낸 것이다. 마지막으로 테스트 케이스는 해당 프로젝트가 가지고 있는 총 테스트 케이스 수를 의미한다.

4.1.2 오픈 소스 프로젝트 수집

템플릿 생성을 위해 오픈 소스 소프트웨어 저장소에서 프로젝트를 수집하였다. 3.1.1절의 설명과 동일하게 타깃 프로젝트와 동일한 프로그래밍 언어와 토픽을 가진 프로젝트를 수집하였다. 그중 GitHub에서 유명도로 사용되는 척도 중 하나인 스타(star)가 많은 순으로 나열하여 Defects4j의 프로젝트당 5개의 프로젝트를 수집하였다. 해당 목록은 부록에 첨부하였다.

4.2 연구 질문

본 논문에서는 다음의 연구 질문에 대하여 답한다.

RQ1: SeeType은 기존의 다른 APR기법에 비해 얼마나 많은 옳은 패치를 생성할 수 있는가?

APR의 주요 목적은 옳은 패치를 생성하는 것이다. 따라서 첫 번째 연구 질문은 제안하는 기법의 옳은 패치 생성률을 살펴본다. 옳은 패치 생성률을 평가하기 위해서 SeeType이 생성한 유효한 패치가 옳은 것인지에 대한 평가가 필요하다. 이를 위해 생성한 유효 패치를 수동으로 확인하여 평가한다.

RQ2: SeeType은 옳은 패치 생성 시 얼마나 많은 시간이 소모되는가?

APR기법을 평가할 때 옳은 패치를 생성할 때 걸리는 시간 또한 중요하다. 따라서 두 번째로 제안 기법의 효율성을 살펴본다. 효율성은 두 가지 방법으로 확인한다. 먼저 각각의 기법들이 옳은 패치를 생성하는 데 걸리는 시간을 비교하고, 그다음 타임아웃의 길이에 따라 얼마나 많은 옳은 패치를 생성할 수 있는지를 비교한다.

RQ3: SeeType이 옳은 패치를 생성하지 못한 경우에는 어떤 것이 있는가?

제안하는 기법이 옳은 패치를 생성하지 못한 경우에는 크게 해당 버그에 대해 유효한 패치를 전혀 생성하지 못한 경우, 유효한 패치를 생성하였으나 수동으로 옳은 패치 평가 시 옳은 패치라고 판단되지 않는 경우가 있다. 따라서 향후 연구를 위해 어떤 버그에 대해서 옳은 패치를 생성하지 못했는지에 대해 분석한다.

연구 질문 1, 2의 비교 대상 기술은 기존의 패턴 기반의 APR 기법의 결과를 정리한 Tbar의 연구 결과[20]와 비교한다.

4.3 프로토타입 결과

SeeType 프로토타입에서 Defects4j의 Math 프로젝트를 대상으로 실험했을 때 기존 연구[7]보다 약 10%가량 새로운 버그에 대해 유효한 패치를 생성하는 것을 확인하였다.

5. 결론

탐사 기반 APR이 생성하는 패치의 질이 떨어진다는 점을 보완하기 위해 많은 연구가 진행되고 있다. 본 논문에서는 더욱더 높은 비율로 옳은 패치를 생성하기 위해 스테이트먼트 유형 기반 탐사 영역 축소 기법인 SeeType을 제안했다. 패치 생성 시 사람이 작성한 패치 정보와 스테이트먼트 유형에 따라 필터링 된 템플릿을 사용하기 때문에 더욱더 많은 옳은 패치와 높은 패치 생성 효율을 얻을 수 있을 것이다. 향후 연구로 기존의 연구에서 사용한 일반적인 벤치마크를 통해 실제 SeeType이 생성하는 패치에 대한 실험 결과를 실행 시간 및 생성하는 옳은 패치 수 측면에서 정량화하여 기존 연구와 비교하고자 한다.

Acknowledge

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업, 개인연구사업의 지원을 받아 수행된 연구임(No. 2017M3C4A7068179, No. 2019R1A2C2006411).

참고문헌

[1] Keigo Naitou, Akito Tanikado, Shinsuke Matsumoto, Yoshiki Higo, Shinji Kusumoto, Hiroyuki Kirinuki, Toshiyuki Kurabayashi, and Haruto Tanno. Toward introducing automated program repair techniques to industrial softwaredevelopment. In Proceedings of the 26th Conference on Program Comprehension, ICPC '18, pages 332–335, New York, NY, USA, 2018. ACM.

[2] L. Gazzola, D. Micucci, and L. Mariani. Automatic software repair: A survey. IEEE Transactions on Software Engineering, 45(1):34–67, Jan 2019.

[3] Edward K. Smith, Earl T. Barr, Claire Le Goues, and Yuriy Brun. Is the cure worse than the disease? overfitting in automated program repair. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, pages 532–543, New York, NY, USA, 2015. ACM.

[4] Ming Wen, Junjie Chen, Rong Wu, Dan Hao, Shing-Chi Cheung. An empirical analysis of the influence of fault space on search-based automated program repair. arXiv preprint arXiv: 1707.05172, 2017.

[5] Zichao Qi, Fan Long, Sara Achour, and Martin Rinard. An analysis of patch plausibility and correctness for generate and validate patch generation systems. In Proceedings of the 2015 International Symposium on Software Testing and Analysis, ISSTA 2015, pages 24–36, New York, NY, USA, 2015. ACM.

[6] F. Long and M. Rinard. An analysis of the search spaces for generate and validate patch generation systems. In 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), pages 702–713, May 2016.

[7] Jiajun Jiang, Yingfei Xiong, Hongyu Zhang, Qing Gao, and Xiangqun Chen. Shaping program repair space with existing patches and similar code. In Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2018, pages 298–309, New York, NY, USA, 2018. ACM.

[8] Fan Long and Martin Rinard. Automatic patch generation by learning correct code. SIGPLAN Not., 51(1):298–312, January 2016.

[9] Yuan Yuan and Wolfgang Banzhaf. ARJA: automated

repair of java programs via multi-objective genetic programming. CoRR, abs/1712.07804, 2017.

[10]Dongsun Kim, Jaechang Nam, Jaewoo Song, and Sunghun Kim. Automatic patch generation learned from human-written patches. In Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, pages 802-811, Piscataway, NJ, USA, 2013. IEEE Press.

[11]Hao Zhong and Na Meng. Towards reusing hints from past fixes. Empirical Software Engineering, 23(5):2521-2549, Oct 2018.

[12]K. Liu, D. Kim, A. Koyuncu, L. Li, T. F. Bissyandé, and Y. Le Traon. A closerlook at real-world patches. In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), pages 275-286, Sep. 2018.

[13]Eduardo C. Campos and Marcelo de A. Maia. Discovering common bug-fixpatterns: A large-scale observational study. Journal of Software: Evolution andProcess, 31(7): e2173, 2019. e2173 smr, 2173.

[14]Github, 2020 년 1 월 2 일 접속, https://www.github.com

[15]Jean-Rémy Falleri, Floréal Morandat, Xavier Blanc, Matias Martinez, and Martin Monperrus. Fine-grained and accurate source code differencing. In Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, ASE '14, pages 313-324, New York, NY, USA, 2014. ACM.

[16]ASTNode, clipse, 2020년 1월 2일 접속, https://help.eclipse.org/2019-12

[17]J.A. Jones and M. J. Harrold. Empirical evaluation of the Tarantula automatic fault-localization technique. In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering. ACM, pages 273-282, 2005.

[18]R. Abreu, P. Zoeteweij, and A. J. c. Van Gemund. An evaluation of similarity coefficients for software fault localization. In 2006 12th Pacific Rim International Symposium on Dependable Computing, pages 39-46, Dec 2006.

[19]M.Y. Chen, E. Kiciman, E. Fratkin, A. Fox, E. Brewer. Pinpoint: Problem determination in large, dynamic internet services. In Proceedings International Conference on Dependable Systems and Networks. IEEE, pages 595-604, 2002.

[20]Kui Liu, Anil Koyuncu, Dongsun Kim, Tegawende F. Bissyande. TBar: Revisiting Template-based Automated Program Repair. In Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2019, 2019.

[21]René Just, Darioush Jalali, and Michael D. Ernst. Defects4j: A database of existingfaults to enable controlled testing studies for java programs. In Proceedings of the 2014 International Symposium on Software Testing and Analysis, ISSTA 2014, pages 437-440, New York, NY, USA, 2014.

부록

부록 A. ASTNode 필드 정보(일부)

수정자와 자료형	필드(field)	설명
static int	ANNOTATION_TYPE_DECLARATION	AnnotationTypeDeclaration 형의 노드를 나타내는 노드 형 정수.
static int	ARRAY_ACCESS	ArrayAccess 형의 노드를 나타내는 노드 형 정수.
static int	ARRAY_TYPE	ArrayType 형의 노드를 나타내는 노드 형 정수.
static int	ASSIGNMENT	Assignment 형의 노드를 나타내는 노드 형 정수.
static int	BOOLEAN_LITERAL	BooleanLiteral 형의 노드를 나타내는 노드 형 정수.
static int	BREAK_STATEMENT	BreakStatement 형의 노드를 나타내는 노드 형 정수.
static int	CAST_EXPRESSION	CastExpression 형의 노드를 나타내는 노드 형 정수.
static int	CLASS_INSTANCE_CREATION	ClassInstanceCreation 형의 노드를 나타내는 노드 형 정수.
static int	CONTINUE_STATEMENT	ContinueStatement 형의 노드를 나타내는 노드 형 정수.
static int	FOR_STATEMENT	ForStatement 형의 노드를 나타내는 노드 형 정수.
static int	IF_STATEMENT	IfStatement 형의 노드를 나타내는 노드 형 정수.
static int	IMPORT_DECLARATION	ImportDeclaration 형의 노드를 나타내는 노드 형 정수.
static int	INFIX_EXPRESSION	InfixExpression 형의 노드를 나타내는 노드 형 정수.
static int	INITIALIZER	Initializer 형의 노드를 나타내는 노드 형 정수.
static int	METHOD_DECLARATION	MethodDeclaration 형의 노드를 나타내는 노드 형 정수.
static int	METHOD_INVOCATION	MethodInvocation 형의 노드를 나타내는 노드 형 정수.
static int	METHOD_REF	MethodRef 형의 노드를 나타내는 노드 형 정수.
static int	PARAMETERIZED_TYPE	ParameterizedType 형의 노드를 나타내는 노드 형 정수.
static int	PARENTHESIZED_EXPRESSION	ParenthesizedExpression 형의 노드를 나타내는 노드 형 정수.
static int	POSTFIX_EXPRESSION	PostfixExpression 형의 노드를 나타내는 노드 형 정수.
static int	PREFIX_EXPRESSION	PrefixExpression 형의 노드를 나타내는 노드 형 정수.
static int	TYPE_DECLARATION	TypeDeclaration 형의 노드를 나타내는 노드 형 정수.
static int	TYPE_DECLARATION_STATEMENT	TypeDeclarationStatement 형의 노드를 나타내는 노드 형 정수.
static int	TYPE_METHOD_REFERENCE	TypeMethodReference 형의 노드를 나타내는 노드 형 정수.
static int	TYPE_PARAMETER	TypeParameter 형의 노드를 나타내는 노드 형 정수.
static int	VARIABLE_DECLARATION_FRAGMENT	VariableDeclarationFragment 형의 노드를 나타내는 노드 형 정수.
static int	WHILE_STATEMENT	WhileStatement 형의 노드를 나타내는 노드 형 정수.

부록 B. 템플릿 생성을 위한 오픈 소스 프로젝트 목록

- JfreeChart(Chart)

프로젝트명	설명
OpenRefine/OpenRefine	자바 기반 데이터 관리 도구
jtablesaw/tablesaw	자바 데이터 프레임 및 시각화 라이브러리
shzlw/poli	SQL 보고 응용 프로그램
AnyChart/AnyChart-Android	안드로이드 어플리케이션 기반 데이터 시각화 라이브러리
KnowageLabs/Knowage-Server	빅 데이터 시스템 관리 라이브러리

- Closure compiler(Closure)

프로젝트명	설명
CalebFenton/simplify	안드로이드 어플리케이션 코드 최적화 라이브러리
Kyson/AndroidGodEye	안드로이드 어플리케이션 성능 모니터링 라이브러리
Kiegroup/optaplanner	인공지능 기반 constraint solver 엔진
Sable/soot	자바 최적화 프레임워크
amitshekhariitbhu/GlideBitmapPool	비트맵 메모리 재사용을 위한 메모리 관리 라이브러리

- Apache commons-lang(Lang), Apache commons-math(Math)

프로젝트명	설명
apache/commons-io	I/O 관련 기능 지원 라이브러리
apache/commons-collections	자바 콜렉션 처리 관련 라이브러리
apache/commons-pool	객체 풀링 API 및 객체 풀 구현 제공 라이브러리
Siznax/wptools	위키피디아(Wikipedia) 데이터 추출 프로그램
apache/commons-codec	자바 인코더, 디코더 구현 라이브러리

- Mockito(Mockito)

프로젝트명	설명
powermock/powermock	테스트 불가 코드 단위화 프레임워크
ihsanbal/LoggingInterceptor	요청, 응답을 위한 logger 가 있는 Okhttp 인터셉터 라이브러리
fabioCollini/DaggerMock	Dagger 2 오브젝트 오버라이드(override) 지원 라이브러리
yale8848/RetrofitCache	캐시 추가 지원 라이브러리
OpenConext/Mujina	OpenSAML 및 자바 Spring Boot 기반 서비스 제공 라이브러리

- Joda-Time(Time)

프로젝트명	설명
JodaOrg/joda-money	화폐 저장 관련 클래스 라이브러리
JodaOrg/joda-beans	JavaBeans 관련 프레임 워크
Gkopff/gson-jodatime-serialisers	Joda Time 엔티티(Entity) 처리 라이브러리
JodaOrg/joda-convert	표준 문자열 형식과의 변환을 지원하는 자바 라이브러리
JodaOrg/joda-time-hibernate	JDK 날짜 및 시간 관련 라이브러리

CCTV를 이용한 실시간 폭행 및 난동행위 인식 시스템 개발

권용휘, 전승원, 배재빈, 김성윤, 김석호, 정순기

경북대학교 컴퓨터학부

gupsin9611@gmail.com, tmddnjsez@naver.com, woqlsdlekz@naver.com,
rerewkd@gmail.com, shkim.vr.knu.ac.kr, skjung@knu.ac.kr

Development of recognition system of violent, disturbance behavior

Yong Hwi Kwon, Sung Won Jeon, Jae Bin Bae, Seong Yun Kim,

Seock Ho Kim, Soon Ki Jung

Computer Science & Engineering, Kyungpook National University

요 약

편의점과 같은 소규모 점포 환경은 대형 마트와 같은 대규모 점포 환경에 비해 범죄행위에 노출되는 빈도가 높다. 하지만 소규모 점포는 그러한 범죄행위에 대응 할 수 있는 CCTV 이상의 정밀한 보안 시스템을 구축할 역량이 없다. 그렇기에 이미 설치된 CCTV의 기능을 확장하여 별도의 장비를 도입하지 않고 범죄 행위에 대응 할 수 있는 시스템을 구축하고자 하였고, 머신 러닝 기술을 접목한다면 CCTV를 단순한 촬영 도구에서 실시간 분석 시스템으로 확장해 과제를 해결할 수 있을 것으로 여겼다. 따라서 본 논문에선 CCTV와 머신 러닝 기술을 응용해 폭력 행위를 실시간 인식하고 폭력 행위로 인식된 프레임을 무선 통신을 통해 모바일 기기로 전송할 수 있는 시스템을 구현하였다.

1. 서 론

통계조사 결과 편의점, 슈퍼마켓과 같은 소규모 점포의 범죄율이 대형 마트, 백화점과 같은 대규모 점포에 비해 월등히 높다는 결과가 도출되었다.[1] 특히 절도와 같은 경범죄보다도 폭행, 강도, 살인 등 중범죄율이 두드러지게 높은 것으로 나타났다. 하지만 이러한 현실에도 불구하고 소규모 점포는 대규모 점포와 달리 보안에 투자할 수 있는 역량이 부족하다. 별도의 고급 보안 시스템을 구축하기엔 비용 부담이 크며, 관리 인력 또한 아르바이트생 등의 저숙련·비전문 인원으로 구성된 경우가 많아 범죄 대응 및 예방은 미진한 실정이다.

따라서 본 논문에선 이미 대부분의 점포에 설치된 기초적인 보안 시스템인 CCTV를 활용한 머신 러닝 기술의 응용과 자동화를 통해 상기한 문제들을 극복할 수 있는 시스템을 구현하였다. 머신 러닝 기술을 통해 실시간으로 폭행 혹은 난동 행위를 인식할 수 있다면 폭력 행위로 인식된 프레임과 현장 위치 정보를 무선 통신 기술을 통해 협력 업체가 사용하는 모바일기기에 전송하여 신속한 범죄 대응을 가능케 할 것이다.

2. 본 론

2.1 시스템 구성 및 설계

본 논문에서는 자동 신고 절차를 가진 머신 러닝 기반 폭력 행위 인식 시스템을 개발하였다. 구현된 시스템에서 설치된 CCTV로 촬영된 영상을 폭력 행위 인식 모듈이 실시간 분석하여 폭력 상황을 인식하면 폭력 행위로 인식된 프레임을 위치 및 시간 정보와 함께 모바일로 전송한다. 모바일에서는 전송 받은 사건 정보를 사용자에게 노출시킨다. 이 때 노출된 정보는 임의의 사용자에 의해 접수될 수 있으며 접수 사건 정보의 현황은 처

리되어 사용자들에게 다시 노출된다. [그림 1] 은 시스템의 흐름을 도식화한 것이다.

그림 1. 시스템 흐름도

2.2 폭력 행위 인식 모듈

핵심 시스템인 폭력 행위 인식 모듈의 개발은 머신러닝 모델 중에서도 Tiny-yolo 모델을 사용하였다. Tiny-yolo에서 학습될 데이터셋은 YOLO-MARK[4]를 이용해 호환되는 형식으로 변환하고 Configure 파일을 수정하면서 Filter 와 Class를 임베디드 보드 환경에 맞게 수정하여 학습시킨다. [그림 2]는 이 과정을 도식화한 것이다.

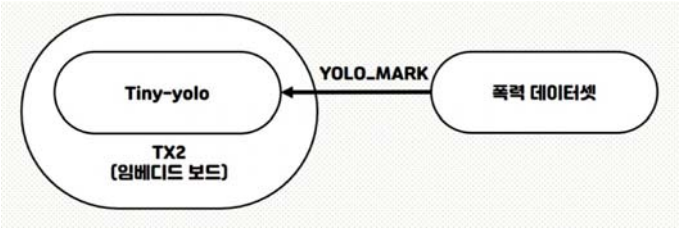


그림 2. 폭력 행위 인식 모듈

이러한 과정을 거쳐서 데이터 셋을 학습시킨 시스템에 통신 모듈을 부착하고 폭력 상황인지 프레임으로 판단하는 조건식을 넣고 조건에 맞게 절차를 실행한다. 폭력 행위 인식 시스템의 실행 결과로 검출된 폭력 행위의 인식은 [그림3]과 같다.

그림 3. 비폭력 및 폭력 상황 검출 결과화면

2.3 통신 모듈

통신 모듈은 web server와 DB로 쓰인 firebase의 웹 호스팅 기능을 연동하여 개발되었다. 먼저 폭력 행위 인식 모듈에서 폭력 행위로 인식된 프레임을 node.js를 통해 웹으로 전송한다. 전송된 프레임 이미지는 firebase 웹 호스팅 기능을 이용해 DB에 저장된다. 최종적으로 DB에 저장된 데이터를 다시 모바일 어플리케이션으로 전송하면 절차가 완료된다.

2.4 모바일 어플리케이션

통신 모듈을 거쳐 전송된 데이터는 모바일 기기의 푸시 알람 기능을 이용해 사용자에게 일부 정보를 노출한다. 전송된 데이터 중 각각 '위치', '시간', '접수현황' 이란 ID값을 가진 데이터 구성된 데이터 블록은 리스트 형태로 메시지 보관함에 디스플레이 된다. 이 때, 사용자가 메시지 보관함을 통해 사건 정보에 접근할 수 있도록 데이터를 디스플레이 하기 위해서 어플리케이션은 DB와 실시간 동기화 상태에 있어야한다. 사용자는 푸시 알람 혹은 메시지 보관함을 통해 상세 화면에 접근할 수 있으며 개별 사건의 완전한 정보가 이 화면에 디스플레이 된다. 또, 이 화면에서 사용자는 임의로 접수 버튼을 클릭함으로써 접수 절차를 밟을 수 있으며 접수 절차가 실행되면 '접수현황' ID를 가진 데이터 값을 '접수됨' 상태로 전환하는 UPDATE 요청이 DB로 전송된다.

또한 시스템의 악용을 방지하기 위해서 어플리케이션은 사용자에게 인증을 요구하며 이는 로그인 형태로 이루어진다.

그림 4. 모바일 어플리케이션 흐름도

2.5 폭력 행위 인식 실험 및 결과

본 시스템이 유용하게 활용 될 수 있는 장소 "Street", "Store", "School" 키워드로 이미지를 검색하여 57장의 폭력 이미지, 43장의 비폭력 이미지 총 100장의 테스트 이미지를 수집하여 실험하였다.
결과는 아래의 [그림 4], [그림 5]와 같다.

Total population		Predicted condition	
		Prediction positive	Prediction negative
True condition	Condition positive	46(TP)	22(FN)
	Condition negative	21(FP)	11(TN)

그림 4. 실험 결과 표

최종적으로 정확도는 위의 표 파란색 영역을 합한 수치인 68% 이며, 붉은색 영역을 합한 에러율은 32% 이다.

Total population		Predicted condition	
		Prediction positive	Prediction negative
True condition	Condition positive	24(TP)	14(FN)
	Condition negative	6(FP)	6(TN)

그림 5. 상점 내에서 일어난 폭력 검출 실험 결과 표

본 논문의 핵심인 CCTV 시점으로 계산이 이뤄지는 장면에 대한 테스트 이미지 50장을 검출한 결과 이다. 또한, 위의 [그림 4] 테스트결과의 단점인 낮은 정확도를 향상 시키기 위해 YOLO모델의 하이퍼 파라메터중 입력 이미지 사이즈를 기존 416에서 608로 늘려 속도는 3 FPS 감소가 있었지만, 8퍼센트의 정확도 향상을 확인 할 수 있었다.

3. 결론

본 논문에서는 CCTV 환경에서 머신 러닝 기술을 응용한 폭력 및 난동 행위 인식과 자동신고 시스템을 구현하였다. 테스트 결과 시스템이 폭력 상황을 인식하고 폭력 상황으로 인식된 프레임을 위치 정보와 함께 자동으로 모바일기기에 전송하는 것을 확인 할 수 있었다. 추가적인 데이터 학습을 통해 좀 더 정교한 상황 판단도 가능할 것이라 생각한다. 또한 자연어 처리기능 등을 추가해 폭력행위만이 아닌 협박·모욕 행위 따위로 기능 확장도 기대해 볼 수 있을 것이며 전송기기를 모바일로 한정하지 않고 전문시스템에 위탁하는 것도 고려해 볼 만하다.

Acknowlegement

“본 연구는 대한민국 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학사업의 연구결과로 수행되었음”(2015-0-00912)

“이 연구는 한국연구재단(NRF-2019R1A2C1010786) 사업의 지원을 받아 수행된 연구임”

4. 참 고 문 헌

[1] 경찰청 (2017). “경찰범죄통계2017”

[2] Joseph Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR, pp.1-10. (2016).

[3] "darknet/LICENCE at master . AlexeyAB/darknet . GitHub",https://github.com/AlexeyAB/darknet/blob/master/LICENSE

[4] "GitHub - AlexeyAB/Yolo_mark", https://github.com/AlexeyAB/Yolo_mark

블록체인 기반 CCTV 영상정보 무결성 지원 방안

강민구 0 홍준기 송성한 김태영 김순태
전북대학교 소프트웨어공학과
{201314090, rlwns012, 201314106, rlaxodud1200, stkim} @jbnu.ac.kr

A Blockchain Based Approach to Supporting Integrity of CCTV Video

Mingu Kang0, Joongi Hong, Seounghan Song, Taeyoung Kim, Suntae Kim
Dept. of Software Engineering, Jeonbuk National University

요 약

CCTV(Closed-Circuit Television)는 범죄 예방 및 재난 안전 등과 같은 공공 부분과 호텔, 은행 등 민간 부분에서도 광범위하게 사용되고 있다. CCTV 를 통해 수집된 영상은 VMS(Video Management System)에 기록되어 핵심적인 범죄 증거 기록물로 활용된다. 하지만, 내부의 악의적인 사용자나 허가 받지 않은 사용자의 외부 접근은 VMS 에 의해 관리되는 기록된 데이터의 무결성을 훼손시킬 수 있다. 이를 해결하기 위하여, 본 연구에서는 CCTV 영상정보 무결성을 위한 블록체인 플랫폼 cctV-i 를 제안하여 VMS 에 기록된 영상정보의 무결성을 보장한다. 제안된 시스템은 VMS 로 API 호출하여 영상정보를 요청, 수신하고, 수신된 영상정보에서 I-Frame 을 추출하여 VMS 에 기록된 영상정보의 무결성을 검증한다.

1. 서 론

CCTV(Closed-Circuit Television)는 일반 카메라와 달리 특정한 목적에 따라 한정된 공간에 카메라를 고정하고, 이를 통해 촬영된 영상물을 디지털 정보로 저장한다. 저장된 정보는 허락된 관계자들에게만 제공하는 폐쇄된 환경 속에서 운영된다. 최근 CCTV 는 범죄 예방 및 재난 안전 등과 같은 공공 부분과 호텔, 은행 등 민간 부분에서도 광범위하게 사용되고 있으며[1], 수집된 영상은 VMS(Video Management System)에 의해 기록, 관리된다.

VMS 는 Server-Client 구조로 이루어진 개방-분산형 보안 관리 시스템으로, VMS 의 주요 목적은 영상의 통합 관제이다. VMS 기반의 영상 감시 시스템은 CCTV 영상정보를 저장할 뿐만 아니라 제어, 검색, 모니터링을 효율적으로 관리할 수 있게 해주어 현대 사회 대부분의 생활 영역에서 이용되고 있다.

하지만, 이러한 VMS 기반의 영상 감시 시스템은 내부 악의적인 사용자나 허가 받지 않은 사용자의 외부 접근으로 인해 저장된 CCTV 영상 정보의 무결성이 훼손될 수 있다. 첫 번째로, 영상의 일부분에 대한 위조를 통해 실제 상황과 다른 정보를 넣어 사회, 범죄 등에 악용될 수 있다. 두 번째로, 범죄를 감추기 위하여 영상정보를 삭제할 수 있다.

따라서, 본 연구에서는 VMS 에 기록된 CCTV 영상정보의 무결성을 위한 블록체인 플랫폼 cctV-i(Blockchain Platform for CCTV Video Integrity)을 제안한다. 제안된 시스템은 VMS 로 API 호출을 폴링하여 영상정보를 요청, 수신하고, 수신된 영상정보에서 I-Frame 을 추출한다. 추출된 I-Frame 은 VMS 에 기록된 영상의 무결성을 검증하기 위한 핵심정보로 활용된다. 수집된 I-Frame 은 해쉬화 되고 블록체인에 저장되어 데이터 조작 가능성을 차단한다. 또한 영상의 핵심 정보를 저장하는 것 이외에 CCTV 별 검색, 조작 여부 확인 등의 여러 비즈니스 확장성을 지원하여 향후 다양한 CCTV 관련 소프트웨어 개발을 지원할 수 있도록 확장 가능한 블록체인 플랫폼으로 설계 및 구축한다.

2. 관련 연구

CCTV 는 물리적인 보안을 제공하고 높은 안전성을 제공하며, 범죄를 예방한다. VMS 는 다수의 CCTV 를 통합, 관제 가능하게 하여 범죄 현장을 조사하거나 용의자 및 피해자의 이동 경로를 추적하는 등 핵심적인 범죄 증거 기록물로 활용될 수 있다. 이러한 CCTV 영상정보의 신뢰성을 훼손하기 위한 다양한 공격이 이루어지고 있으며, 이에 따른 대응책도 함께 연구가 이루어지고 있다[2].

DaiKyung Hyun 등은 센서 패턴 노이즈를 이용해서 CCTV 영상정보의 위/변조를 탐지하는 기법을 제시하였다[3]. 디지털 이미지 장치에서 이미지 센서는 일반적으로 픽셀로 구성되어 있다. 이 픽셀은 빛에 대해 서로 다른 감도를 가져 고유한 패턴 노이즈를 발생시킨다. 해당 연구는 카메라마다 발생하는 고유한 패턴 노이즈를 추출하고 비교하여 CCTV 영상정보의 위/변조 유무를 판단한다. 하지만 Jinseok Park 은 센서 패턴 노이즈는 인위적으로 생성해 낼 수 있어, 영상정보에 위/변조를 가한 뒤 센서 패턴 노이즈를 생성해 조작 흔적을 감출 수 있음을 설명한다[4]. Jinseok Park 이 제시한 연구는 고정된 광원에 의해 촬영되는 적외선 이미지를 대상으로 하는 포렌식 기법을 제시하고 있다. 하지만 야간에

촬영된 적외선 이미지에 한해서만 제시한 기법이 적용 가능하다는 한계를 가지고 있다.

최근에, 블록체인 기술이 확산됨에 따라 다양한 비즈니스 네트워크에 블록체인을 접목하고 있다. 블록체인은 위/변조로부터 데이터를 보호할 수 있어 저장된 데이터의 신뢰성과 무결성의 보장이 필요한 시스템에 활용될 수 있다. Pierluigi Gallo 등은 스마트 시티 구축의 기반 기술로 블록체인의 활용을 소개하며 CCTV 카메라의 메타데이터를 통한 IoT 영상 감시 환경을 제안한다[5]. Yena Jeong 등은 신뢰할 수 있는 관리자로 구성된 블록체인 기반 영상 감시 시스템을 제안한다. 해당 연구는 영상정보의 메타데이터를 블록체인에 저장하며 별도의 IPFS 를 두어 영상정보를 저장한다[6]. 또한, Donghyeok Lee 등은 엣지 블록체인 기반의 지능형 영상감시환경을 제안한다. CCTV 영상정보에 담긴 개인정보를 마스킹하여 프라이버시 문제를 해결한다[7].

하지만 앞서 언급된 블록체인 기반 연구들은 기존의 VMS 기반 영상 감시 시스템과 함께 운용될 수 없다. 위 연구들은 CCTV 영상정보의 무결성을 위한 새로운 블록체인 기반의 영상 감시 시스템을 제안한다. 제안된 시스템들은 기존의 VMS 기반 영상 감시 시스템의 확장을 고려하지 않아 바로 접목 시키기엔 한계가 있어 영상 감시 시스템의 교체가 불가피하다. 시스템의 교체는 막대한 비용을 발생하게 하며, 기존의 VMS 의 CCTV 통합 관제와 제어와 검색 등과 같은 제공되는 서비스를 이용할 수 없게 된다. 따라서, VMS 와 독립적인 시스템으로 존재하면서 VMS 에 기록된 영상정보에 대한 무결성을 보장해줄 수 있는 확장 가능한 구조가 필요하다.

본 연구는 VMS 기반 영상 감시 시스템에 탈 중앙화 된 블록체인 기술을 적용하여 VMS 에 저장되는 CCTV 영상정보의 조작을 방지하기 위한 시스템을 구축한다. 또한, 추출된 영상정보를 조작하기 위한 분산 응용프로그램(DApp) cctV-i 를 개발하여 여러 비즈니스 확장성을 지원한다.

3. CCTV 영상정보 무결성을 위한 블록체인 플랫폼

3 장에서는 CCTV 영상정보의 무결성을 위한 블록체인 플랫폼의 개요를 설명하고, 제안하는 시스템에서 영상정보 처리에 사용되는 모듈에 대해 설명한다. 3.1 에서는 영상정보가 블록체인에 저장되는 과정을 다루고, 3.2 에서는 영상정보의 무결성 검증에 대해 소개한다.

그림 1 은 CCTV 영상정보의 무결성을 보장하기 위한 블록체인 플랫폼 개요를 나타낸다. 제안된 시스템에는 다수의 CCTV 를 통합 관제하기 위한 VMS 와 영상정보가 저장될 블록체인 네트워크 및 이들을 구동하기 위한 분산 응용프로그램인 cctV-i 로 구성되어 있다. VMS 는 다수의 CCTV 로부터 수집되는 영상을 기록, 관리하여 Client 가 각각의 CCTV 를 통합 관제 가능하도록 서비스를 제공한다. cctV-i 는 VMS 와 독립적으로 블록체인 네트워크에 영상정보를 저장하여 Client 에게 무결성 검증을 가능하게 한다.

cctV-i 는 VMS 로부터 영상정보를 요청, 수신하고 핵심 정보를 추출하여 블록체인 네트워크에 전달한다. cctV-i 를 이루는 모듈로서 VMS 와 통신하기위한 VMS Connector, 수신한 영상정보에 대한 처리를 담당하는 Video Manager, 영상정보의 I-Frame 을 추출하기 위한 FFMPEG, HyperLedger Fabric 으로 구성된 블록체인 네트워크와 통신하기 위한 Blockchain Connector 로 구성되어 있으며 Client 는 Presentation 모듈을 통해 영상에 대한 검증을 시각적으로 확인 가능하다.

본 시스템은 VMS 에 기록된 CCTV 영상정보의 무결성을 유지하기 위해 I-Frame 을 블록체인에 저장하는 단계와 VMS 에 기록된 영상이 블록체인에 저장된 영상과 다르지 않음을 검증하기 위한 단계로 이루어 진다.

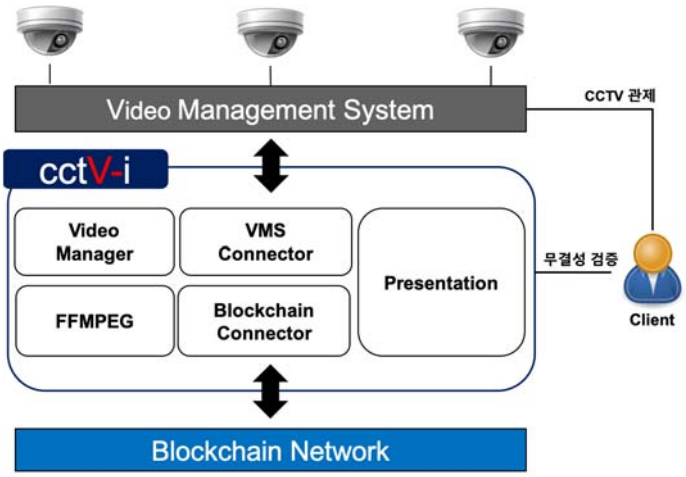


그림 1 CCTV 블록체인 플랫폼 개요

3.1 영상정보 저장

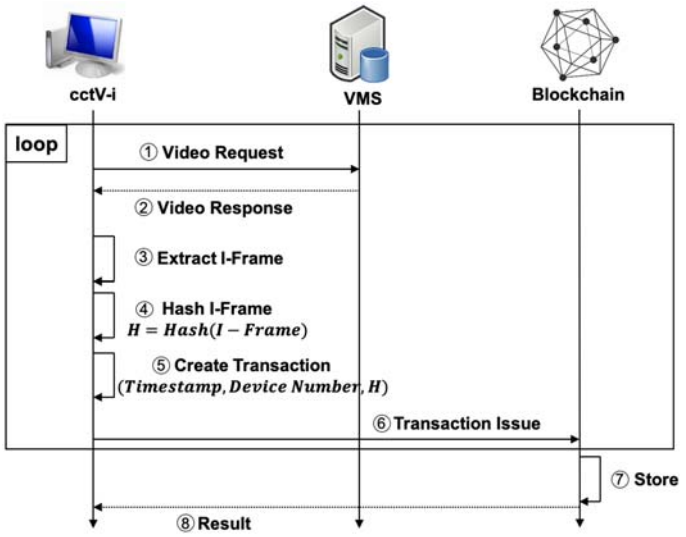


그림 2 영상정보 저장 흐름

CCTV 의 무결성을 증명하기 위한 핵심 정보로서 영상정보의 I-Frame 을 이용한다. I-Frame 은 연속적인 프레임 집합으로 이루어진 영상정보에서 해당 프레임을 표시하는데 필요한 모든 정보를 가지고 있는 단일 프레임이다. I-Frame 추출은 영상에 대한 요약 정보를 제공하여 영상 분석 및 관리에

필수적인 부분이다[8]. 또한, 추출된 I-Frame 을 시간 순서대로 추적하여 영상에 대한 내용을 인식할 수 있다.

이러한 I-Frame 의 특성은 영상정보의 무결성을 검증하는데 이용될 수 있으며, 네트워크 전송 스트레스와 대용량 스토리지의 필요성을 감소 시킬 수 있다. CCTV 의 영상정보는 고용량 데이터로서 블록체인에 영상을 그대로 저장할 경우, 이를 처리하기 위한 비용이 커지게 된다. 따라서, 영상정보를 그대로 저장하지 않으면서, VMS 에 저장된 영상과 블록체인에 저장된 정보가 다르지 않다는 것을 증명 가능하여야 한다. 또한, VMS 에 위/변조가 일어난 구간을 특정하고 시각화하여 확인할 수 있어야 한다. 이를 해결하기 위해 본 연구는 영상정보의 I-Frame 을 추출하여 저장, 검증하는데 활용한다.

그림 2 는 CCTV 영상정보가 블록체인 네트워크에 저장되기까지의 영상정보 저장 흐름을 나타낸다. cctV-i 는 VMS 로 API 호출을 폴링하여 특정 CCTV 장치 번호에 해당하는 MPEG 형식의 영상정보를 요청, 수신한다. cctV-i 와 폴링 방식으로 통신하는 VMS 는 일정한 주기를 가지고 요청의 수신과 반환을 반복한다. cctV-i 는 수신한 영상정보에서 블록체인에 저장하기 위한 정보인 I-Frame 과 해당하는 Timestamp 를 FFMPEG 을 통해 추출한다. 추출된 I-Frame 은 SHA-256 알고리즘을 통하여 해쉬화되고, 추출된 Timestamp 및 CCTV 장치 번호와 함께 트랜잭션으로 구성된다. cctV-i 는 만들어진 트랜잭션을 Blockchain Connector 를 통하여 HyperLedger 블록체인 네트워크로 전달하고, 블록체인 네트워크는 영상을 저장하기 위한 체인코드를 실행하는 트랜잭션을 발행한다. 블록체인 네트워크의 저장소에 저장되는 구조는 다음과 같다.

$$KEY = UnixTime(Timestamp)\_DeviceNumber$$
$$VALUE = Hash(I-Frame)$$

위 식에서 KEY 는 HyperLedger 블록체인 저장소의 키 값을 나타낸다. 추출된 I-Frame 에 해당하는 Timestamp 를 유닉스시간으로 변경하고 CCTV 장치 번호를 조합하여 KEY 를 나타낸다. VALUE 는 블록체인 저장소에 저장되는 실제 값으로, 추출된 I-Frame 에 대한 해쉬값으로 나타낸다. 영상정보의 무결성 검증 단계에서 KEY 를 이용해 블록체인 저장소에 대한 조회가 이루어지고, VALUE 는 VMS 에 저장된 영상정보와 비교하여 무결성을 검증하기 위해 사용된다.

3.2 영상정보 무결성 검증

무결성 검증 단계에서는 VMS 에 기록된 영상정보와 블록체인 네트워크에 저장된 영상정보는 다르지 않다는 것을 검증한다. 이를 위하여 VMS 에서 수신한 영상정보의 I-Frame 해쉬값과 블록체인에 저장된 해당하는 I-Frame 의 해쉬값을 비교하여 무결성을 검증한다.

그림 3 은 Client 에 의한 VMS 영상정보의 무결성 검증 흐름을 나타낸다. Client 는 검증을 원하는 CCTV 영상 구간과 CCTV 장치 번호를 입력하여 cctV-i 에게 무결성 검증을 요청한다. 검증을 요청 받은 cctV-i 는 VMS 로 API 호출하여 관련 CCTV 영상정보를 요청, 수신한 뒤 I-Frame 을 추출한다. 추출된 I-Frame 은 SHA-256 알고리즘을 통해 해쉬화되어 해쉬값 H_a 를 얻는다. 추출된 I-Frame 에 해당하는 Timestamp 와 CCTV 장치 번호를 통해 블록체인의 저장소를 조회하는 체인코드를 실행하고 해쉬값 H_a 를 반환 받는다. 마지막으로, VMS 로부터 얻은 해쉬값 H_a 와 블록체인으로부터 얻은 해쉬값 H_b 를 비교하여 영상정보의 무결성을 검증하고 결과를 Client 에게 반환한다.

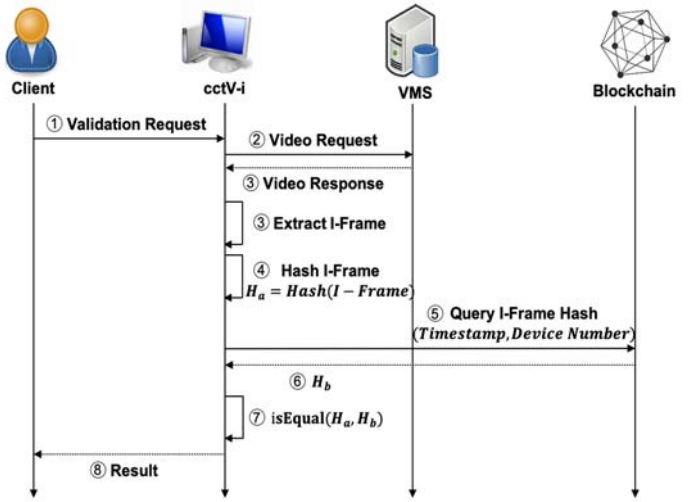


그림 3 영상정보 무결성 검증 흐름

4. 실험 및 결과

본 논문에서는 향후 확장성 및 다양한 기술지원이 가능한 대표적인 사설 블록체인 네트워크인 Hyperledger Fabric v1.4 블록체인 플랫폼 운용 환경을 구성하였다. 모든 실험은 2.3 Ghz Intel Core i5 CPU 와 8GB RAM 을 가진 macOS 에서 실행되었다. cctV-i 는 영상정보를 얻기 위해 VMS 로 폴링하며, 폴링 주기는 VMS 실험 대상인 VURIX VMS v1.4 의 API 호출 최소 단위인 1 분으로 지정하였다.

그림 4 는 CCTV 블록체인 플랫폼 화면을 보여준다. Client 는 검증하고자 하는 CCTV 장치 번호와 검증 구간을 입력할 수 있다. 입력된 값에 해당하는 CCTV 장치번호와 VMS 에 기록된 영상정보의 I-Frame 별 해쉬, 블록체인에 저장된 I-Frame 별 해쉬를 보여주고 위/변조 여부를 테이블화 하여 보여줄 수 있다. 또한, 기록된 I-Frame 해쉬값에 해당하는 화면을 시각적으로 보여줄 수 있도록 구축하였다.

본 연구에서 제안한 시스템은 VMS 에 기록된 영상정보의 I-Frame 과 블록체인에 저장된 I-Frame 을 비교하여 위/변조를 모두 식별해 내었다. 실험을 위해 블록체인에 영상을 저장한 뒤 해당 영상 앞에 영상을 추가하는 위/변조를 가하고 cctV-i 를 통해 영상을 검증하였다. 그림 4 에서는 위/변조된 영상에 대한 무결성 검증 결과를 테이블로 나타내고 있다. 블록체인에 저장된 I-Frame 의 Timestamp 와 해쉬값, 검증 요청한 영상의 해쉬값을 보여주고 위/변조 여부를 제공한다. 실험 결과 검증 요청 영상에 대해 앞 2 개의 I-Frame 에서 위/변조를 식별하였고, 블록체인에 저장된 정보와 일치하는 정보가 없어 영상이 추가되었음을 알 수 있다.

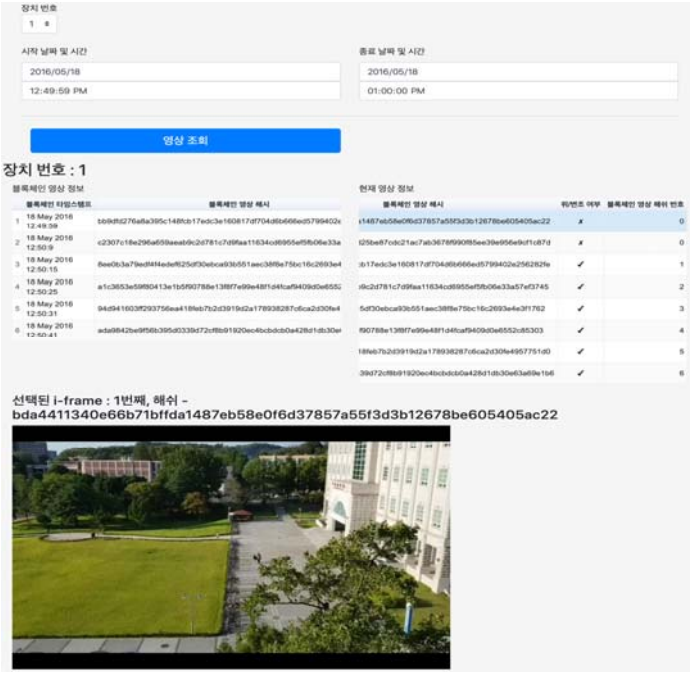


그림 4 cctV-i 블록체인 플랫폼 화면

API 호출 주기에 따라 영상정보가 블록체인에 저장되기까지의 평균 시간은 5 초가 소요된다. 표 1 은 실행 시간 1 분을 갖는 영상에 대한 영상정보와 실험 결과를 보여준다. 실험에 사용된 영상은 MPEG 형식을 갖고, 720x480 해상도와 15MB 의 크기를 갖는다. 해당 영상은 8 개의 I-Frame 이 추출되었으며, 저장 시간은 영상이 수신된 상태에서 블록체인에 저장되기까지의 시간을 나타낸다.

표 1 영상정보 및 실험 결과

영상 형식	MPEG
영상 해상도	720 x 480
영상 크기	15MB
I-Frame 수	8
저장 시간	5 초

5. 결론

본 연구는 VMS 에 기록된 CCTV 영상정보의 무결성을 검증하는 블록체인 플랫폼을 제안하였다. VMS 에 기록된 영상정보는 내부의 악의적인 사용자나 허가 받지 않은 사용자의 외부 접근은 기록된 영상정보의 무결성이 훼손될 수 있다. 본 연구는 VMS 기반의 영상 감시 시스템에 블록체인을 접목하여 저장된 영상정보의 무결성 검증을 지원하고 DApp 을 이용해 확장 가능한 비즈니스 플랫폼을 구축하였다. 제안된 영상정보 처리 기법은 영상의 I-Frame 을 이용해 블록체인 네트워크의 경량화를 이루면서 위/변조에 대한 무결성을 검증한다. 또한, VMS 와 독립적인 시스템으로 쉽게 확장 가능한 시스템으로 구축하였다. 제안된 시스템은 중앙 통제 관리되는 VMS 에 분산 원장 기술인 블록체인을 접목해 VMS 에 대한 신뢰성과 무결성 검증에 기여하였다.

향후 연구는 VMS 의 의존도를 없애면서 쉽게 확장 가능한 구조의 고려가 필요하다. IoT 기기를 통해 블록체인 네트워크에 저장되는 값의 원시 데이터부터 제어하고, CCTV 장치에 접근하는 허가되지 않은 사용자의 접근 기록과 카메라 조작 로그 기록을 함께 관리할 수 있는 방법에 대한 연구를 진행할 계획이다.

6. Acknowledgement

본 논문은 정부(정보통신기술진흥센터)의 재원으로 대학 ICT 연구센터 육성지원 사업의 지원을 받아 수행된 연구임 (No.IITP-2019-2017-0-01628)

7. 참고문헌

[1] 윤우석; 이창훈; 심희섭. 국내 CCTV 운영 현황 및 위치선정 가이드라인 제안에 관한 연구. *한국민간경비학회보*, 2017, 16: 109-144.

[2] COSTIN, Andrei. Security of cctv and video surveillance systems: Threats, vulnerabilities, attacks, and mitigations. In: *Proceedings of the 6th international workshop on trustworthy embedded devices*. ACM, 2016. p. 45-54.

[3] HYUN, Dai-Kyung, et al. Detection of upscale-crop and partial manipulation in surveillance video based on sensor pattern noise. *Sensors*, 2013, 13.9: 12605-12631.

[4] PARK, Jin-Seok, et al. Detecting digital image forgery in near-infrared image of CCTV. *Multimedia Tools and Applications*, 2017, 76.14: 15817-15838.

[5] GALLO, Pierluigi; PONGNUMKUL, Suporn; NGUYEN, Uy Quoc. BlockSee: Blockchain for IoT video surveillance in smart cities. In: *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*. IEEE, 2018. p. 1-6.

[6] JEONG, Yena; HWANG, DongYeop; KIM, Ki-Hyung. Blockchain-Based Management of Video Surveillance Systems. In: *2019 International Conference on Information Networking (ICOIN)*. IEEE, 2019. p. 465-468.

[6] LIU, Guozhu; ZHAO, Junming. Key frame extraction from MPEG video stream. In: *2010 Third International Symposium on Information Processing*. IEEE, 2010. p. 423-427.

[7] 이동혁; 박남제. 엣지 블록체인 기반의 CCTV 영상 프라이버시 보호 기법. *한국정보기술학회논문지*, 2019, 17.10: 101-113.

[8] LIU, Guozhu; ZHAO, Junming. Key frame extraction from MPEG video stream. In: *2010 Third International Symposium on Information Processing*. IEEE, 2010. p. 423-427.

STEAM 교육과 게임을 도입한 교육용 프로그래밍 언어 기본 설계

이정희, 김민우, 조성휘, 김정아
가톨릭관동대학교 컴퓨터교육과
dlwjdgmlrnt@cku.ac.kr, minwoo0805@cku.ac.kr, aqs456@cku.ac.kr, clara@cku.ac.kr

A Design of EPL for Game-based STEAM Education

Jung-Hee Lee, Min-Woo Kim, Sung-Hwi Jo, Jeong-ah Kim
Catholic Kwandong University

요 약

4차 산업혁명 시대를 들어서면서 과학중심의 창조력과 인문학적 상상력을 겸비한 창의·융합형 인재 양성을 필요로 한다. 2015 개정교육과정부터 초·중등학교에서 정보교과가 정규교과목으로 편입되고 학생들의 학습시간 및 학습의 필요성이 증대되면서 코딩 교육에 대한 수요가 늘어났지만, 코딩교구 종류가 다양하지 않고 이를 수용할 수 있는 환경을 갖추고 있지 못하다. 이러한 환경을 개선하고 코딩교육을 그 자체로의 의미보다는 융합형 인재양성의 중요한 수단으로 활용할 수 있는 방법이 필요하다. 본 연구에서는 학생들의 흥미를 유발하고 융합적 학습 효과를 높일 수 있는 새로운 교육용 프로그래밍 언어(EPL: Educational Programming Language)를 제안하고자 한다. 본 연구에서 제안하는 EPL은 실생활에 활용 가능한 지식과 실질적 문제해결에 적용할 수 있도록 여러 교과를 자연스럽게 연결하고 융합되는 STEAM(Science, Technology, Engineering, Arts, Mathematics)교육을 게임에 자연스럽게 녹여 컴퓨팅 사고력과 창의적 사고력을 키울 수 있는 방향으로 설계하였다.

1. 연구목적 및 연구문제

최근 4차 산업혁명 시대에 맞춰 정보과학기술 교육을 강화하려는 추세이다.

교육 선진국인 나라들의 교육정책에서 파악할 수 있듯이 융합인재교육은 4차 산업혁명을 준비해야 하는 현시점에 매우 중요한 교육이다. 한국에서도 이러한 문제를 해결하기 위해 2011년부터 과학기술·예술 융합(STEAM) 교육 활성화 방안을 발표하고 2015개정교육과정에 STEAM교육을 반영하였다[1].

4차 산업혁명 시대에 SW 역량을 갖춘 인재 양성의 목적은 단순 지식의 습득능력 보다 컴퓨팅 사고력(Computational Thinking)기반의 창의적 문제 발견 및 해결 능력 양성이다. 그러므로 디지털 경제 시대의 세계 흐름에서 주도권을 확보할 수 있는 SW 역량을 갖춘 인재 양성을 위한 혁신적인 교육이 필요하다.

따라서 본 연구에서는 기존 EPL들의 단점을 개선하고자 학습자들의 흥미를 유발하며 컴퓨팅 사고력을 키울 수 있는 STEAM교육과 게임에 EPL을 도입하여 학습자의 계속적 학습 유도를 위한 전략으로 EPL을 설계하고자 한다.

2. 연구 방법

우리나라의 현 사회에서는 미래융합인재와 SW 역량을 가진 인재를 양성하기위해 2015개정교육과정을 통해 코딩교육을 강화하고 그 방안으로 EPL을 도입했지만 현재 이 두가지 모두를 충족시킬 만한 연구가 제대로 이루어지지 않았다.

현재 대표적인 EPL와 현재 개발하고자 하는 EPL을 'Game요소', 'STEAM요소', '추상화', '코딩 방식', '접근성' 항목으로 비교하면 표 1과 같다.

	Game요소	STEAM요소	추상화	코딩 방식	접근성
Blockly	O	O	O	블록형	높음
Scratch	X	X	O	블록형	낮음
Code studio	O	O	O	블록형	낮음
Entry	X	X	O	블록형	높음
Aduino	X	O	X	텍스트	낮음
HAGO	O	O	O	블록형	높음

표 1. 기존 EPL와 새로운 EPL의 비교

본 연구는 이러한 기존의 EPL의 문제점을 보완하고자 새로운 EPL을 개발한다.

3. EPL 설계

3.1 EPL 설계 방향

본 설계는 실생활문제와 연결된 상황에서 자연스럽게 각 교과탐구를 가능하게 하는 SW가상실험실 블록형 코딩 교구(HAGO)를 설계한다.

HAGO는 STEAM교육과 GBL을 병합한 EPL이다. 일반 교과의 교육과정에서 배우는 교과 이론을 코딩게임을 통해 알아가는 학습 방식이고, 각 단계별로 배운 내용을 토대로 학습자들이 직접 아이디어 단계부터 적용 결과까지 구상할 수 있도록 프로그래밍 환경을 설계한다.

EPL의 특성상 프로그래밍 전문가를 양성하는 프로그래밍 언어가 아니고 프로그래밍을 처음 시작하는 입문 단계를 위한 교육용 프로그래밍 언어이기에 사용자그룹의 대상은 초등학교 3학년에서 6학년까지, 중학교 1학년에서 3학년까지를 대상으로 한다.

3.2 HAGO 컴포넌트 정의

EPL을 융합교육에 적용할 때 다양한 교과와 접목할 수 있도록 개발해야 하지만, 첫 단계로 과학과목과 연계할 수 있도록 개발한다. HAGO를 사용할 학습자들이 학습할 과학교과목에서 필요로 하는 실험용 도구를 EPL 환경에서 활용할 수 있도록 설계한다. 과학실험용 기구를 크게 6개의 범주로 구분하고 각 범주 별 필요한 도구를 SW컴포넌트로 개발하며, Entry의 오픈소스 (Licensed under the Apache License, Version 2.0.)을 블록형 코딩 교구의 기반 통합 환경으로 활용한다[2].

그림 1은 과학교과의 교육과정을 포함한 컴포넌트 예시이다. 과학교구 중 하나인 골드버그를 인용해 HAGO에 적용시킨다. 다양한 과학원리에 대해 배우고 실생활에서 흔히 볼 수 있는 도구를 이용해 고정관념에서 벗어난 다양한 아이디어를 게임을 통해 알아본다[3].

그림 1. 과학적이론(골드버그) 컴포넌트 예시

4. 프로토타이핑 결과

HAGO 개념을 프로토타이핑으로 개발한 HAGO 통합 환경은 그림 2과 같다. 환경을 구성하는 요소는 다음 표 2와 같다.

그림 2. HAGO 통합 환경

번호	기능	정의
1	이동버튼	원하는 스테이지 이동
2	실행화면	프로그래밍 결과 값 실행화면
3	도움말	교과과정 이론 및 부연설명
4	블록꾸러미	스테이지별 프로그래밍용 블록
5	블록조립소	프로그래밍 구현

표 2. HAGO 통합환경 구성요소

HAGO의 메인 화면으로 게임 클리어 조건에 따라 블록을 조립하고 프로그래밍을 목표조건에 올바르게 실행하였다면 실행화면에 결과를 볼 수 있도록 구상하였다.

HAGO에서 제공되는 게임 이외에도, 마지막 스테이지에서는 앞서 배운 내용을 토대로 학습자들이 직접 화면을 꾸며서 실행결과를 바꿀 수 있도록 설계하였다.

5. 결론

본 논문에서는 현 교육에 있어서 핵심인 융합인재교육과 SW역량교육에 대해서 이야기하고, HAGO 프로그램 설계 전략에 대해 이야기했다. 이에 본 설계에서는 STEAM교육과 GBL을 활용하여 학습자 중심의 EPL을 설계하였다. 이를 통해 다음과 같은 효과를 기대할 수 있다.

첫째, 학습자들이 쉽게 게임을 통해 STEAM교육과 코딩교육을 접근할 수 있다.

둘째, 정보교과에서 배우는 고정관념에서 벗어나 일반교과 융합적 관점을 제시하고 원리와 이론에 대해서 배울 수 있고, 실생활에서 제시되는 다양한 분야의 문제들을 해결할 수 있다.

셋째, 시대적 흐름에 맞춰 융합형 인재와 SW 역량 인재를 양성하고, 과학·공학의 사고능력을 벗어나 인문·예술을 아우를 수 있는 소양을 키울 수 있다.

6. 참고문헌

[1] STEAM교육, steam.kofac.re.kr (2020년 1월 확인)
[2] 재단법인 커넥트 entry, https://playentry.org/
[3] NCIC 국가교육과정정보센터, http://ncic.go.kr/

IoT 초보 개발자를 위한 검색어 향상

정 찬미 ○ 남 재창
한동대학교 전산전자공학부
migujeong@gmail.com, jcnam@handong.edu

Search Query Augmentation for Novice IoT Developers

Chan-Mi Jeong ○ Jae-Chang Nam
School of Computer Science and Electrical Engineering, Handong Global University

요 약

Internet of Everything(IoE)이 기술적 트렌드가 되어가면서 많은 개발자들이 다양한 목적을 가지고 IoT 분야로 뛰어들고 있다 [1]. 하지만 몇몇 개발자들은 선행지식 없이 프로젝트에 착수하여, 문제에 직면했을 때 검색 엔진에서 정보를 찾는데 어려움을 겪는다. 검색어는 기술적 용어가 포함된 적절한 단어의 조합이 이상적이나, 초보 개발자들의 검색어는 특정 문제에 대한 해결책을 찾기에 너무 짧거나 추상적이다. 본 연구에서는 Word2vec 모델을 사용하여 검색어 향상 기법을 제안하였다. 해당 연구를 통해 향상된 정보 탐색을 지원할 수 있는 확장된 검색어를 관찰할 수 있었다. 추후 검색어구 재생성 및 검색 결과 추천 연구로 이어질 수 있다.

1. 서 론

IoT 초보 개발자들은 대개 선행 지식이 없는 상태에서 기술적인 정보를 검색한다. 이들은 기술 용어가 포함된 유용한 정보들을 찾기 어려워하는 경향을 보이는데, 그들이 만들어내는 검색어의 조합이 너무 추상적이거나 짧은 경우가 많기 때문이다. 또한, 해당 검색어를 수정하는 과정에서 적절한 기술 용어를 떠올리기 어려워한다. 예를 들어, 한 개발자가 라즈베리 파이 보드에 연결된 USB 마이크를 통해 목소리를 녹음하고자 한다. 그러나 유효하지 않은 샘플 레이트라는 에러 메세지를 받는다. 그는 사전 정보가 없으나, 라즈베리 파이에서 마이크 샘플 레이트를 수정하는 것에 관심이 있다고 할 수 있다. 처음에는 'mic sample rate'로 검색하여 샘플 레이트에 대한 설명이 있는 결과를 얻었다. 하지만 이는 궁극적으로 개발자가 원하는 결과가 아니다. 그 다음에는 'mic sample rate raspberry pi'로 검색어를 바꾸어 검색하였더니 원하는 정보인 마이크 샘플 레이트 수정에 대한 튜토리얼을 얻을 수 있었다.

사용자의 의도에 맞는 정보를 얻기 위한 방법론은 정보 검색 분야에서 중요한 주제이며, 이와 관련하여 여러 가지 시도들이 있어왔다. 대부분의 접근 방식은 사용자의 기록을 분석하거나, 일반적인 주제에 집중하여 이루어졌다 [2,3,4,5]. 하지만 이러한 접근 방식은 많은 양의 데이터 경향성에 의지하기 때문에, 비교적 적은 양의 정보가 존재하는 IoT 같은 특정한 필드에서는 효과적인 방식이 아니다. Ma et al. [6]의 연구와 같이 적은 데이터로 시도한 사례도 있으나, 여전히 사용기록과 랜덤한 알고리즘에 의존하고 있다.

본 연구에서는 사용기록에 의존하지 않은 상태에서 IoT 관련 정보에 대한 접근성을 높일 수 있는 수단으로 검색어 향상 기법을 제안하였다. 기술적 용어는 재사용성이 높기 때문에, 단어 임베딩을 통한 단어간 코사인 유사도를 만들어내는 Word2vec 모델 [7]을 사용하여 원래 검색어와 코사인 유사도가 높은 단어를 추출하여 원래 검색어에 생성된 단어를 추가하는 방식으로 검색어 향상을 도모하였다. 이를 통해, 확장된 검색어가 사용자의 의도에 가까워진 것을 확인할 수 있었다.

2. 관련 연구

검색 결과와 검색어의 질을 향상시키기 위한 대표적인 시도들이 있다. 첫째, 유저의 검색어와 선택된 URL을 bipartite graph로 나타내고, 해당 그래프에 agglomerative algorithm을 적용하여 컨텐츠의 내용과 무관하게 유사한 URL과 검색어들을 클러스터링 하는 방법이다 [2]. 둘째, 미리 정의된 분류 체계에서 세부 분류를 선택하는 방법인 faceted search가 있다 [8]. 세번째, 검색어의 대안을 생성하여 제시하는 방법이다 [3,4,5,6]. 이 방법들은 URL, 검색어, 클릭한 페이지 등의 사용기록에 의존하고 있다. 기존 연구의 접근 방식들은 검색 결과의 향상을 꾀한다는 점에서 본 연구와 유사하다. 그러나, 기존 연구에서는 일반적인 분야에서 사용기록 등 통계 기반의 검색어 생성에 집중하거나 미리 정의된 속성이 있는 상태였다.

본 연구에서는 IoT 분야에서의 구체적인 해결책이나 정보 습득을 위해 적은 데이터로 의미 있는 기술적용어가 추가된 검색어를 생성하는데 초점을 두고

진행하였다. 또한, 문서 내용 기반의 검색어 향상 방안을 강구하였다는 측면에서 차이가 있다.

3. 접근 방식

3.1 구글 검색 결과 수집

검색어 향상 첫 단계로 사용자의 검색어에 따른 Google 검색 결과를 크롤링하였다. 먼저, 검색 결과에 포함된 문서 스니펫(snippet)을 수집한다. 이 때, 날짜 정보와 영어가 아닌 언어로 된 문서, 비디오 결과는 모두 제외하였다. 또한, 미리 크롤링이 허용된 사이트들을 분류하여 검색 결과에 해당 사이트들이 포함된 경우 그 문서의 불필요한 내용을 제거한 후 추가 수집하였다.

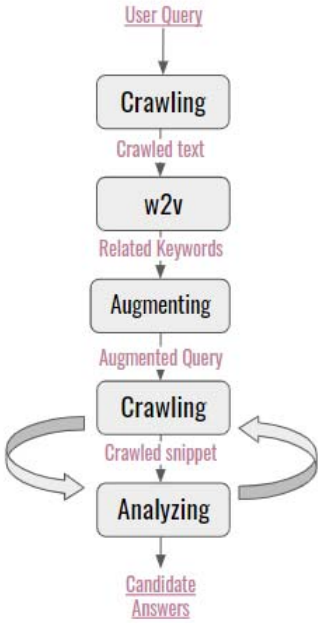


그림 1 접근방식 요약 차트

추가적으로, 수집하는 텍스트는 검색 결과의 10페이지까지의 범위로 한정하였다. Google 검색 엔진에서는 10페이지가 넘어가면 관련성이 낮은 문서들이 등장할 가능성이 높기 때문이다 [9]. 또한 Google 검색 엔진 자체적으로 검색어 오타를 보완하는 기능이 있으므로 사용자의 원래 검색어는 오타가 수정된 경우를 제외하고는 수정하지 않았다 [10].

3.2 Word2vec 모델로 학습

말뭉치에서 단어 벡터 형태로 변환하여 벡터 간 코사인 유사도를 추출해주는 Word2vec 모델을 이용함에 있어 다음과 같은 전처리를 진행하였다. 크롤링 단계에서 불필요한 정보를 제거한 후 모델에 말뭉치를 넣기 전에 추가적인 처리를 하였다. 먼저, 두 글자보다 짧은 단어는 제거하였다. 그 다음에는 Treebank 토큰화 규칙에 맞추어 토큰화 한 후 정규화 처리하였다. 그런 상태에서 불용어와 특수 문자를 제거하였으나, 말뭉치에 포함된 숫자는 제거하지 않았다. ‘mic sample rate’와 같은 몇몇 검색어의 경우에서는 숫자가 마이크의 일반적인 샘플 레이트를 의미할 수 있으며, 해당 케이스에서는 숫자 또한 유사도 높은 의미 있는 단어가 될 수 있기 때문이다. 단어 최소 등장 횟수를 5번으로 정하고, Word2vec 모델 중 하나인 Skip gram을 통해 말뭉치를 학습하였다. 이 때 모델 학습 결과로 생성된 단어집합 중, 검색어 토큰과 코사인 유사도가 가장 높은 단어들을 선정하여 확장 쿼리 집합에 추가시켰다.

3.3 검색어 향상

검색어를 향상시키는 방법에는 검색어 삭제, 검색어 대체, 검색어 확장이 대표적이다 [4]. 이 중 검색어 삭제와 검색어 대체는 사용자의 검색어에 담긴 의도를 훼손시키거나 바꿀 수 있기 때문에 적용하지 않았다 [4].

확장 쿼리 집합에 있는 쿼리 후보 중 동사와 형용사 형태는 제거하였다. 해당 품사들은 기술적 용어에 가깝기 보다 언어의 일반적 사용에 가까운 경우가 많았다. 반면에 명사와 숫자는 쿼리 후보로 선정하였다. 3.2절에서 언급하였듯이 사용자 의도와 관련이 있는 기술 용어일 가능성이 높다고 판단했기 때문이다. 해당 과정을 통해 확장 쿼리 집합 요소 개수를 감소시켰다. 마지막에 도출될 정답 후보의 개수 또한 감소시키기 위함이다.

위와 같은 분류 과정을 통해 얻어진 단어들을 원래 검색어에 각각 추가하는 방식으로 검색어를 확장하였다.

3.4 구글 검색 결과 재 수집 및 정답 후보 분석

확정된 확장 쿼리 집합을 검색어로 사용하여 검색 결과의 첫번째 페이지를 재 크롤링 한다. 이는 첫 번째 페이지에 나열된 10개의 문서를 정답 후보로 두고 분석하기 위함이다. 그후 정답 후보들을 검토하여 해당 문서가 검색어의 의도에 부합하는 경우 정답으로 표기하였다.

4. 실험 관련 설정

검색어 수집과 함께 사용자 의도 및 의도에 따른 정답의 조건을 정의하였다. 실험에 사용될 검색어는 기록만으로 의도를 정의할 수 없으므로, 개발자가 실제로 사용하여 의도를 정의할 수 있어야 했다. 결과적으로 의도가 파악된 라즈베리 파이 관련 실제 검색어 위주로 수집되었다.

다음으로 원래 문서에서 구글에서 제공하는 짧은 문장 또는 구들로 이루어진 문서 스니펫을 추출했다. 또한 크롤링 가능한 IoT 관련 사이트로부터 추가 수집하였다. 해당 사이트에서도 기술적 소통을 위한 특정 용어를 사용할 가능성이 높기 때문이다.

표 1 실험결과 요약표

	OQ	AQ1	AQ2	AQ3	AQ4	AQ5	AQ6	AQ7
case 1	raspberry pi mic sample rate (5/10)	bit 5/10:1	jasper 5/10:1	asound 5/10:1	file 4/10:0	audio 3/10:0	usb 5/10:0	microphone 4/10:1
	import error name pubnub (4/10)	pi 5/10:2	env 2/10:1	python 4/10:2	google 5/10:3	module 5/10:3	data 4/10:2	raspberry 6/10:4
	force raspberry pi turn hdmi (5/10)	monitor 5/10:1	boot 7/10:3	display 7/10:2	dmt 8/10:7	mode 8/10:4	tv 5/10:2	cable 6/10:2
	raspberry pi encrypt sensor (4/10)	ssl 4/10:1	gchq 2:10:0	api 5/10:2	rc4 1:10:0	password 2/10:1	aes 3/10:3	debian 4/10:3
	raspberry pi zigbee2mqtt start failed (5/10)	zigbee 4/10:0	update 4/10:2	test 4/10:1	addon 1/10:1	links 5/10:3	firmware 3/10:0	forum 3/10:0
case 2	rc522 raspberry pi (0/10)	rfid 0/10:0	com 0/10:0	spi 1/10:1	module 1/10:1	pin 0/10:0	tutorial 2/10:2	arduino 0/10:0
	raspberry pi zero ros melodic (3/10)	people 2/10:0	instruction 2/10:1	velocity 2/10:1	make2 1/10:0	fri 1/10:0	opencv 0/10:0	opencv3 2/10:1
	mqtt client no message (3/10)	synergy 1/10:1	documentation 1/10:1	token 1/10:1	file 0/10:0	response 3/10:1	transport 3/10:3	receiving 5/10:4
case 3	Pi incorrect date and time (8/9)	com 9/10:1	10 5/10:2	ntp 9/10:2	github 4/10:4	notification 6/10:5	clock 9/10:1	192 4/10:2
	raspberry pi no wifi (8/10)	adapter 8/10:5	cable 8/10:7	version 7/10:4	setting 6/10:6	headless 2/10:2	ssh 3/10:3	wlan0 7/10:6
	self signed certificate with openssl (9/10)	authority 8/10:4	testing 5/10:3	configuration 6/10:5	windows 2/10:2	san 7/10:7	private 9/10:4	liner 5/10:4

재 크롤링 단계에서는 확장된 쿼리로부터 검색 결과를 분석하여 검색 결과 첫번째 페이지에 출력된 결과 합집합으로부터 정답 후보를 구별하였다. 특히, 3.1절의 기조에 따라 재분석 단계에서도 영어가 아닌 문서와 비디오 결과를 제외하였다.

정답 후보는 사용자 검색어의 의도에 부합하는 구체적인 해결책이 있거나 사용자가 스스로 검색어를 향상시킬 수 있는 충분한 정보가 주어진 경우에 해당한다. 예를 들어, 'force raspberry pi turn hdmi'가 원래 검색어인데 검색어에 담긴 사용자의 의도는 모니터와 라즈베리 파이를 연결하였지만 A 모니터에는 출력결과가 있는 반면 화질이 좋은 B 모니터는 동작하지 않아 이에 대한 해결법을 찾고자 하는 것이다. 의도에 부합하는 문서는 라즈베리 내부적 설정을 통해 연결된 모니터를 강제로 켜게 하는 해결법이 있거나, 높은 해상도를 가진 모니터에 대해 호환되게 만드는 해결법을 언급하고 있어야 한다. 다른 검색어에 대해서도 위 사례와 같이 의도에 따른 정답 후보의 조건을 정의하여 실험을 진행하였다.

더불어 확장된 검색어의 성능을 평가하기 위해 다음과 같은 표기법을 만들어 이용하였다. 검색어에 해당하는 첫 페이지의 문서 인스턴스 전체 개수를 e로 나타내고, 그 중 의미 있는 문서 인스턴스 개수를 a로 나타낸다. 또한, 원래 검색어에서는 나타나지 않았지만 확장된 검색어 결과에서 나타난 새로운 의미 있는 인스턴스를 n으로 표기한다. 이 때 n의 개수가 많을수록 의미 있는 검색어 향상이 이루어졌다고 할 수 있다. 다시 말해, 원래 사용자 쿼리 결과(OQ)에 대해서는 a/e로, 확장된 검색어 결과(AQ)에 대해서는 a/e: n로 표기하였다.

$$a/e: n \quad (1)$$

5. 실험 결과

11개의 검색어들에 대해 단어 단위의 확장을 적용하여 실험을 진행하였으며, 검색 결과의 문서들을 분석하여 세 가지의 패턴을 추출하였다.

표1의 case 1에서는 검색어 확장 테크닉이 항상 의미 있는 비율의 결과를 도출하지는 않는 것이 관찰된다. 왜냐하면 원래 검색어 결과에도 a가 10개 중 5개를 차지하고 있었기 때문이다. 그러나, 확장

검색어의 결과에도 비슷한 비율의 a가 등장하였고 AQ1, AQ2, AQ3, AQ7의 경우에 n이 있기 때문에 사용자가 e를 통해 새로운 의미 있는 결과를 얻을 가능성이 높다.

case 2에서는 우리의 연구 목적에 가장 부합하는 결과가 관찰된다. 사용자가 구체적으로 검색어를 작성하지 않았지만 AQ에서는 n이 관찰되었기 때문이다. 예를 들어, 원래 검색어가 'rc522 raspberry pi'이었다. 사용자의 의도는 MFRC522가 Raspberry pi 3 보드의 SPI 인터페이스상에서 정상적으로 작동하지 않아서 이에 대한 해결책을 찾고 싶었던 것이라고 가정한다. 그렇다면 정답 후보의 조건은 다른 라이브러리를 찾을 수 있는 충분한 힌트가 등장하거나 해당 라이브러리를 업데이트하라는 내용의 조언이 포함되어 있어야 할 것이다. 즉, OQ에서는 a가 존재하지 않았으나 AQ의 정답 후보에서 n이 등장하였으므로 의미가 있다.

case 3는 유의미한 검색 결과가 충분할 때는 검색어 확장이 사용자의 원래 의도를 훼손시킬 수 있음을 시사하는 경우이다. OQ에 이미 충분한 a가 등장한 경우에 AQ에서는 오히려 a가 일정하지 않은 비율로 등장하는 것을 확인할 수 있다. 이런 경우에는 사용자가 검색어 확장을 신뢰할 수 없다. 즉, 이미 첫번째 페이지에 다수의 a가 등장하였으므로 검색어 향상 시도가 무의미하다고 볼 수 있다.

6. 결론

실험을 통해 검색어 확장 기법이 사용자의 원래 의도를 훼손하는 경우도 있었지만, [4]에서와 같이 검색어 삭제나 대체 보다는 확장기법이 검색어의 의도 훼손이 적은 것과, 유의미한 결과를 새로 만들어내는 것을 관찰할 수 있었다. 결론적으로 본 연구의 검색어 확장 테크닉이 추상적이고 짧은 검색어를 완전히 보완할 수는 없지만 초보 개발자의 검색을 지원하는 도구로는 사용될 수 있음이 관찰되었다.

추후 연구에서 [4]의 semantic rephrasing에서 착안하여 검색어 향상 방안으로 전문가들이 사용하는 용어로 이루어진 검색어구 재생성을 생각해 볼 수 있을 것이다. 또한, 우리는 하나의 AQ에서 유용한 인스턴스 a'가 등장하는 경우 또 다른 AQ에서 동일한 a'를 관찰하였다. 그러나 해당 a'는 OQ에서는 등장하지 않는 n에 속하므로 의미가 있다. 그러므로, 검색어 확장에 이어 정답 후보 추천으로도 연구가 이어질 수 있을 것이다.

※ 이 논문은 과학기술정보통신부의
소프트웨어중심대학 지원사업 (2017-0-00130)
의 지원을 받아 수행하였음.

참고문헌

[1] M. H. Miraz, M. Ali, P. S. Excell, and R. Picking. A review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT). In 2015 Internet Technologies and Applications(ITA). 2015. 219-224.

[2] Beeferman, Doug, and Adam Berger. Agglomerative clustering of a search engine query log. KDD. Vol. 2000. pp. 407-416.

[3] CUCERZAN, Silviu; WHITE, Ryen W. Query suggestion based on user landing pages. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2007. p. 875-876.

[4] Jeff Huang and Efthimis N. Efthimiadis. 2009. Analyzing and evaluating query reformulation strategies in web search logs. In Proceedings of the 18th ACM conference on Information and knowledge management (CIKM '09). ACM, New York, NY, USA, 77-86.

[5] SAHAMI, Mehran; HEILMAN, Timothy D. Generating query suggestions using contextual information. U.S. Patent No 7,725,485, 2010.

[6] MA, Hao, et al. Learning latent semantic relations from clickthrough data for query suggestion. In: Proceedings of the 17th ACM conference on Information and knowledge management. ACM, 2008. p. 709-718.

[7] J. Lilleberg, Y. Zhu and Y. Zhang, "Support vector machines and Word2vec for text classification with semantic features," 2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC), Beijing, 2015, pp. 136-140.

[8] DanielTunkelang. Faceted search. Synthesis lectures on information concepts, retrieval, and services 1,1 2009 .1-80.

[9] SU, Ao-Jan, et al. How to improve your Google ranking: Myths and reality. In: 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. IEEE, 2010. p. 50-57.

[10] Xin Li, Nawaaz Ahmed, Fuchun Peng, and Yumao Lu. Query rewriting with spell correction suggestions using a generated set of query features. USPatent 2009 7,630,978.

오픈소스 라이선스 충돌 여부 식별을 위한 TF-IDF 및 LDA 기법 성능 분석

남성국, 이동건, 서영석
영남대학교 컴퓨터공학과
sd05031@ynu.ac.kr, dklee77@ynu.ac.kr, ysseo@yu.ac.kr

A Performance Analysis of TF-IDF and LDA for Identifying Opensource License Compatibility

Seong-Guk Nam, Dong-Gun Lee, Yeong-Seok Seo
Department of Computer Engineering, Yeungnam University

요 약

현대의 소프트웨어 개발에 있어 많은 개발자들은 오픈소스 소프트웨어를 사용하여 효율적으로 소프트웨어 개발을 진행한다. 오픈소스는 모두에게 공개되어 있어 누구나 사용할 수 있지만, 라이선스에 명시된 사용자의 의무를 지켜주어야 한다. 만약 이러한 의무가 성실하게 수행되지 않을 경우 개발자에게 법적인 조치를 취할 수 있으며, 해당 문제에 대한 책임을 요구하게 된다. 따라서 본 논문에서는 TF-IDF와 LDA를 통해 실제 오픈 소스 소프트웨어에서 라이선스를 식별할 수 있는지 분석하고 이를 활용하여 위반을 검출할 수 있는지 평가한다.

1. 서 론

소프트웨어 개발을 진행함에 있어 오픈소스는 빼놓을 수 없는 요소 중 하나이다. 누구에게나 열람 및 수정, 사용, 재배포가 허락되어 있고 [1], 많은 개발자들의 참여로 인해 많은 발전을 이루어 다양한 종류의 오픈 소스 소프트웨어를 활용하여 원하는 프로젝트를 효율적으로 개발을 진행할 수 있다.

하지만 오픈소스는 라이선스의 방식을 통해 배포되며, 라이선스에서 요구하는 최소한의 조건을 반드시 충족시켜 주어야 한다. 그 중에서도 특히 GPL 2.0 라이선스의 경우 배포 조건이 매우 엄격하여 하나의 프로젝트에 타 라이선스와(e.g. Apache 2.0 License)같이 사용할 경우, 저작권자로부터 해당 문제에 대한 책임을 요구하는 상황이 발생할 가능성이 높다.

(Fig. 1)은 2018년 오픈 소스 라이선스의 분포 현황을 보여준다. 전체 라이선스 중 Apache가 22%, GPL 2.0이 10%로 총 32%를 차지하고 있었다. MIT 라이선스와 같은 Permissive License(최소한의 오픈 소스 성격만을 보장하는 것을 허용하는 오픈 소스 라이선스)의 비중이 증가하는 추세지만 Apahce와 GPL 2.0이 3개중 1개일 정도로 여전히 많은 비중을 차지 하고 있다 [2]. 따라서 해당 논문에서는 자연어 처리 기법인 LDA와 TF-IDF를 이용하여 프로젝트의 소스코드를 분석하여 라이선스 사용 검출을 진행하여, 초보 개발자나 라이선스의 배포조건을 모르는 개발자로부터 해당 실수의 발생을 사전에 방지하고, 라이선스를 명확히 명시하지 않은 프로젝트로부터 라이선스의 배포 조건을 지키지 않는 문제를 검출하는 것을 목표로 한다.

본 논문의 2장에서는 관련 연구에 대해 살펴보고 3장에서는 해당 논문에서 진행한 연구의 진행 과정에 대해 소개하며 4장에서는 실험 과정 및 결과 5장에서는 결론 및 향후 연구를 기술한다.

2. 관련연구

MIouki의 연구[3]는 오픈 소스 모바일 앱 개발자가 자주 사용하는 라이선스의 종류를 알아내기 위해, 857개의 모바일 앱을 분석했다. 분석 결과, 대부분의 오픈 소스 모바일 앱은 GPL, Apache 라이선스를 사용했고 857개중에 17개의 앱에서 라이선스 위반을 발견했고 7개의 앱은 여전히 최신 릴리즈에서 라이선스 위반을 발견했다. 또한, 첫 번째 릴리즈에서 많은 파일이 라이선스를 명시하지 않는다는 것을 보였다.

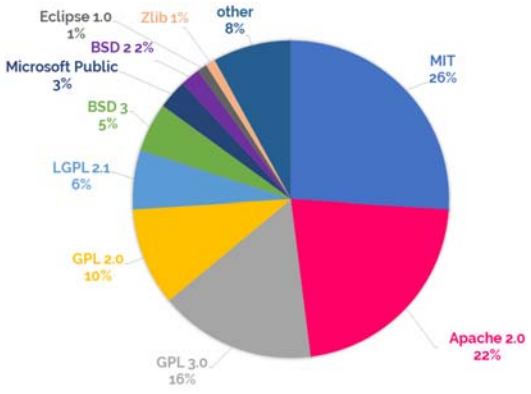


Fig. 1. Distribution of Licenses

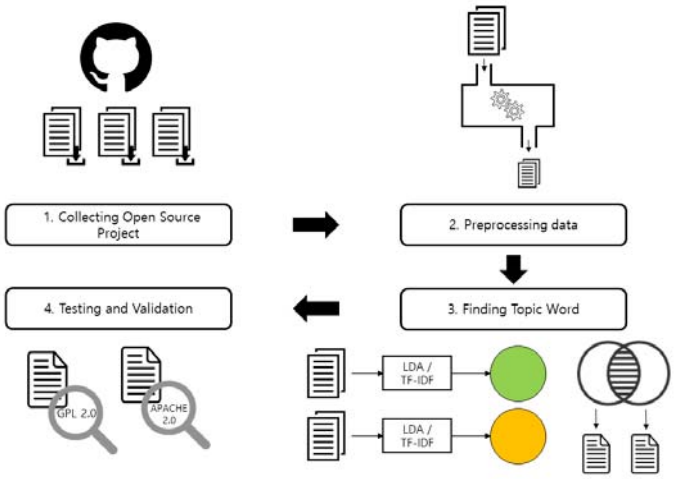


Fig. 2. Overall approach

이를 통해 많은 개발자들이 라이선스의 법적 제약을 이해하지 못하고 있다는 것을 제시했다.

Almeida [4]는 개발자가 그들이 사용하는 오픈 소스 라이선스를 제대로 이해하고 있는지를 평가하기 위해, 3개의 인기있는 라이선스를 단독으로 또는 조합해서 소프트웨어를 개발하는 시나리오에 대한 설문조사를 수행했다. 주로 개발자로 구성된 375명의 응답자에 대해 전체에서 62%가 법률전문가의 의견과 일치하는 결과를 보였다. 개발자는 라이선스가 단독으로 사용되었을 때는 문제가 없었지만 여러 라이선스가 조합해서 사용되었을 때는 이를 이해하는 데에 어려움을 겪었다. 따라서 개발자가 라이선스에 대해 잘 이해할 수 있도록 도구 지원이 필요함을 제시한다.

3. 접근법

본 논문에서 제안한 라이선스 위반 식별을 위해 (Fig. 2)와 같은 과정을 거친다. 다음 절부터는 각각의 단계에 대해 보다 구체적으로 소개한다.

3.1 데이터 수집

이 단계에서는 각 라이선스 별 특징 혹은 의미를 가진 단어를 추출하기 위해 유사한 주제를 가진 SW를 수집한다. 본 논문에서는 여러 개발자의 참여하여 다양한 특징을 보이면서, 질적으로 오픈소스 소프트웨어를 수집한다. 또한, 라이선스 위반을 식별하기 위해서, 검출을 원하는 라이선스의 특징을 갖는 단어를 추출할 필요가 있으므로, 사용자가 식별하기를 원하는 라이선스로 배포한 오픈소스 소프트웨어를 사용하는 것을 전제로 한다.

3.2 전처리화

해당 단계는 전단계에서 수집한 프로젝트 파일들을 효율적으로 처리하기 위해 다양한 전처리 과정을 수행하는 단계이다. 본 논문에서는 NLP모델인 LDA,

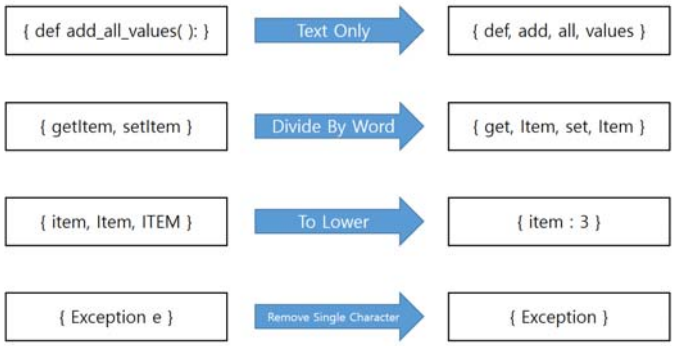


Fig. 3. 4 types of Preprocessing

TF-IDF를 수집한 프로젝트에 적용하여 라이선스가 가지는 특성을 대표하는 단어를 추출하게 된다. 그러나 소스코드 데이터 (Raw Source Code)를 그대로 적용하면 추출하는 단어의 라이선스 식별 정확도가 떨어지는 동시에 시간적 효율성도 떨어진다. 따라서 다음과 같은 전처리 과정을 거치게 된다.

소스코드 내에 포함되어 있는 특수기호 같은 불필요한 내용에 대한 학습(train)을 방지하기 위해 영어 및 숫자로 구성된 내용만 추출한다(1). 스네이크 케이스(Snake Case), 카멜 케이스(Camel Case) 등의 변수 및 함수 이름을 단어 단위로 분리(2)하여 처리량 감소와 동시에 라이선스 식별 정확도 향성을 기대한다. 같은 단어가 대문자로 인해 다른 단어로 인식됨을 방지하기 위해 모두 소문자로 변경하는 과정을 진행한다(3). 마지막 전처리는 단일 문자 제거이다. ‘a’, ‘x’와 같이 단일 문자로 이루어지는 단어는 ‘temp’ 변수와 같이 일시적으로 쓰이거나 의미가 없는 변수일 가능성이 높다. 따라서 단일 문자로 이루어진 단어는 추출목록에서 제거한다(4). (Fig. 3)은 본 연구에서 진행할 전처리 과정을 시각적으로 보여준다.

3.3 단어 추출

해당 단계는 전 단계에서 전처리된 단어를 이용하여 LDA, TF-IDF를 시행하는 단계이다. 충분히 수집된 오픈소스 소프트웨어에 대해 LDA와 TF-IDF를 이용하여 추출된 단어들은 해당 라이선스의 대한 특성을 내포하고 있어 이 단어들을 새로운 오픈소스 소프트웨어에서 찾아보는 것으로 그 오픈 소스의 라이선스를 예측해 볼 수 있다.

구체적으로 TF-IDF는 전처리된 오픈소스 소프트웨어 단어 문서군 (Corpus)에서 어떤 단어가 특이하게 나타나는 지를 조사한다. 즉, 특정 문서 내에서 단어 빈도가 높은 단어를 선택하면서 동시에, 전체 문서에서 공통적으로 포함하는 단어를 배제한다. 본 연구에서는 각 오픈 소스에 들어있는 모든 단어에 대한 TF-IDF 값을 구하고 그 값의 상위 30개의 단어를 라이선스 식별에 사용한다.

LDA는 전처리된 오픈 소스 소프트웨어 문서가 특정한 토픽이 확률적으로 선택되어 문서가 완성되었다고 가정하고 이를 역으로 추론하여 해당 문서의 토픽이 무엇인지 도출한다. LDA는 사용자가 미리 입력한 토픽 셋 개수에 따라 다른 분야의 주제를 나타내는 토픽들을 찾는다. 본 연구에서는 토픽 셋의 종류에 상관없이 가중치가 가장 높은 상위 30개의 단어를 이용하여 라이선스 식별을 진행한다.

3.4 라이선스 식별

이 단계는 LDA와 TF-IDF를 통해 얻은 단어들을 이용하여 실제로 라이선스를 식별하는 단계이다. 전 단계에서 얻은 TF-IDF/LDA별 상위 30개의 단어가 테스트 오픈 소스 소프트웨어에 얼마나 포함되어 있는지 평가하고 기준치 이상의 단어가 검출되면 해당 라이선스를 가진 다른 오픈 소스 소프트웨어를 임포트 했을 가능성이 높은 것으로 판단한다.

4. 평 가

4.1 실험 설계

오픈 소스 라이선스를 식별하기 위해 검증하고자 하는 라이선스를 대표하는 단어들을 추출할 때 사용할 오픈 소스 프로젝트들을 수집한다. 본 실험에서는 GitHub에서 Apache 2.0 License를 사용한 프로젝트 파일과 GPL 2.0 License를 사용한 프로젝트 파일들을 각각 5개씩 수집했다.

TF-IDF의 경우 파이썬의 Sklearn 라이브러리를, LDA의 경우 파이썬의 TOMOTOPY 라이브러리를 사용하여 단어들을 추출하였다. 각 라이선스별로 검출한 단어에는 프로그래밍 언어적 특징을 지닌 단어들이 유사하게 포함되어 있어 해당 결과로는 라이선스별 특징을 명확히 검출하기 어렵다는 판단을 하여, 교집합 연산을 이용하여 중복 단어들을 모두 제거한 후 실험을 진행하였다.

4.2 실험 결과

수집한 오픈소스 프로젝트에 대해 TF-IDF, LDA를 적용해본 결과 다음과 같은 (Fig. 4), (Fig. 5)와 같이 Apache, GPL 2.0 특징 단어들을 얻을 수 있었다.

```
TF-IDF Result

Apache 2.0 License Topic Result
[import, end, view, expect, let, red5, ruby, org, logstash, null,
config, client, override, conn, object, gsyv, player, final, level, frag,
class, state, media, server, type, java, rtmp, handshake, do, track, ]

GPL 2.0 License Topic Result
[box, av, thumbnailer, system, ui, include, codec, window, playlist, qstring,
std, mpv, image, frame, text, forms, widget, format, define, 370,
or, 366, set, param, bool, be, width, index, changed, item, ]
```

Fig. 4. Feature words obtained by TF-IDF

```
LDA Result

Apache 2.0 License Topic Result
[end, import, do, ruby, final, logstash, config, expect, class, view,
org, plugin, java, pipeline, let, stash, override, path, true, def,
util, exception, subject, settings, player, else, test, metric, state, boolean, ]

GPL 2.0 License Topic Result
[av, codec, box, const, audio, frame, define, format, system, fmt,
by, or, res, window, ui, qstring, text, include, mpv, pix,
playlist, filter, ffmpeg, sample, free, ff, version, width, bool, item, ]
```

Fig. 5. Feature words obtained by LDA

```
private void EnableAudioControls(bool enable)
{
    if (enable == true && AudioCodec == "copy" || AudioCodec == "pcm")
    {
        groupBoxAudioBitrate.Enabled = false;
    }
    else
    {
        groupBoxAudioBitrate.Enabled = enable;
    }

    groupBoxAudioEncoder.Enabled = enable;
    groupBoxAudioOutput.Enabled = enable;
}
```

Fig. 6. Example of identified feature words

얻은 단어를 테스트 하기위해 실제 오픈소스 소프트웨어의 해당 단어들이 얼마나 포함되어 있는지 알아보았다. (Fig. 6)는 실제 오픈소스 소프트웨어의 소스코드 상에서 단어가 실제로 검출됨을 보여준다.

추출한 단어로 오픈 소스 소프트웨어 위반을 식별할 수 있는지 알아보기 위해 수집한 소프트웨어가 추출한 단어를 얼마나 포함하는지를 알아보았다. (Table 1)과(Table 2)는 오픈 소스에 포함된 단어의 개수에 대한 전체 추출된 단어의 개수의 비율을 보여준다. 테이블의 Apache topics는 수집한 소프트웨어에 대해 Apache 라이선스의 토픽을 사용했을 때의 비율, GPL 2.0 topics는 GPL 2.0 라이선스의 토픽을 사용했을 때의 비율이다. (Table 1)은 TF-IDF를 사용하여 Apache와 GPL 2.0의 특징을 갖는 토픽을 추출하여 얻은 비율을 보여준다. 각각의 (같은 라이선스의 식별 비율 - 다른 라이선스의 식별 비율)의 값이 최소 0에서 최대 0.33까지 나타난다. (Table 2)는 LDA를 사용하여 Apache와 GPL 2.0의 특징을 갖는 토픽을 추출하여 얻은 비율을 보여준다. 각각의 (같은 라이선스의 식별 비율 - 다른 라이선스의 식별 비율)의 값이 최소 -0.13에서 최대 0.44까지 나타난다. 이를 통해, LDA는 TF-IDF에 비해 변동폭이 크게 나타나며 라이선스 식별에 사용하기에 불안정함을 볼 수 있다.

5. 결 론

해당 논문에서는 소스코드 내 라이선스 사용을 검출하기 위해, 자연어처리기법을 이용하여 소스코드내 라이선스와 연관된 토픽을 검출하여 포함여부를 비교하는 실험을 진행하였다. 해당 연구에서는 개발자가 작성하는 소스코드를 중심으로 연구를 진행하여, 사람이

Table 1. Rate of feature topics in open software with TF-IDF

Test Project \ Training Set	Apache topics	GPL 2.0 topics
jitsi-meet (Apache)	0.77	0.70
GSYVideoPlayer (Apache)	0.73	0.60
hls.js (Apache)	0.67	0.57
logstash (Apache)	0.63	0.57
red5-client (Apache)	0.53	0.33
JavaAV (GPL 2.0)	0.47	0.53
FFmpegAdapter (GPL 2.0)	0.67	0.73
FFmpegCapture (GPL 2.0)	0.40	0.50
ffmpegthumbnailer (GPL 2.0)	0.27	0.60
mpc - qt (GPL 2.0)	0.70	0.70

Table 2. Rate of feature topics in open software with LDA

Test Project \ Training Set	Apache topics	GPL 2.0 topics
jitsi-meet (Apache)	0.80	0.83
GSYVideoPlayer (Apache)	0.67	0.80
hls.js (Apache)	0.73	0.57
logstash (Apache)	0.77	0.67
red5-client (Apache)	0.67	0.60
JavaAV (GPL 2.0)	0.53	0.63
FFmpegAdapter (GPL 2.0)	0.77	0.90
FFmpegCapture (GPL 2.0)	0.50	0.43
ffmpegthumbnailer (GPL 2.0)	0.33	0.77
mpc - qt (GPL 2.0)	0.77	0.77

작성하는 코드를 대상으로 실험하기 위해 수집한 프로젝트의 소스코드내 자연어의 형태를 가진 단어들만 추출하는 전처리 과정을 진행하였고, 연구에 사용하지 않은 소스코드를 대상으로 단어 포함 여부를 검사하여 결과를 도출하였다. 해당 과정을 통해 Apache 2.0 License와 GPL 2.0 License를 사용한 프로젝트의 소스코드를 LDA와 TF-IDF 기법을 적용하여 각각의 토픽을 검출하였고, 해당 단어들의 소스코드 내 포함여부를 계산하여 라이선스 사용을 검출을 시도하였고 LDA는 TF-IDF에 비해 최고 성능치는 더 높았지만 최소값의 경우 오히려 0 이하로 떨어지는 등, 불안정한 모습을 보여 TF-IDF에 비해 라이선스를 식별하는 프로세스에는 부적합하다고 판단하였다.

해당 연구의 결과의 신뢰성 향상을 위해, 향후 다음과 같은 이슈들을 적용하여 연구를 진행하고자 한다. Apache 및 GPL 2.0 라이선스 기반의 오픈소스 소프트웨어 외에 다른 라이선스를 활용하고 있는 오픈소스 소프트웨어들의 추가적인 확보 및 실험을 통해 본 연구에서 도출한 실험 결과의 신뢰성을 향상시키고자 한다. 또한 서포트 벡터 머신 (Support Vector Machine: SVM)이나 신경망 네트워크 (Neural Network)와 같은 기계학습 기법들을 활용하여 오픈소스 소프트웨어에서 사용하고 있는 라이선스들을 높은 확률로 정확하게 식별해낼 수 있는 기법을 제안하고자 한다.

Acknowledgement

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2017R1C1B5018295).

Reference

[1] Open Source software Competency Plaza, OSS definition [Internet], https://www.oss.kr/en_oss_definition.

[2] White Source, Top 10 Open Source Licenses in 2018: Trends and Predictions [Internet], https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions-2019

[3] Ons Mlouki, Foutse Khomh, Giuliano Antoniol, On the detection of licenses violations in the android ecosystem, 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Vol. 1, pp.382-392, 2016.

[4] Daniel A Almeida, Gail C Murphy, Greg Wilson, Mike Hoye, Do software developers understand open source licenses?, 2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC), pp.1-11, 2017.

텍스트 마이닝을 사용한 Configuration 버그 리포트 예측

최정환, 최지원, 류덕산, 김순태

전북대학교

{jeongwhan.choi, wanny_94, duksan.ryu, stkim}@jbnu.ac.kr

Prediction for Configuration Bug Report Using Text Mining

Jeongwhan Choi, Jiwon Choi, Duksan Ryu, Suntae Kim

Jeonbuk National University

요 약

Configuration 버그는 소프트웨어 실패의 주요한 원인들 중 하나이다. 소프트웨어 조직들은 이슈 트래커 시스템을 통해 버그 리포트들을 수집하고 관리한다. 소프트웨어 개발자는 해당 버그가 Configuration 버그인지 인지하는데 막대한 시간을 소비할 수도 있다. Configuration 버그라는 것을 알 수 있는 접근 방법을 통해 개발자의 노력을 줄일 수 있다. 이러한 접근 방식을 Configuration 버그 리포트 예측이라고 칭하며, 해당 문제를 해결하기 위해 텍스트 마이닝 기법을 사용하여 분류 모델을 생성한다. 개발자는 이러한 모델을 사용하여 버그 리포트에 Configuration 또는 Non-configuration 버그로 레이블을 지정할 수 있다. 본 논문은 5개의 오픈소스 소프트웨어 프로젝트로부터 3,962개의 버그 리포트를 추출하고 텍스트 마이닝과 피쳐 엔지니어링 기법을 이용해 Configuration 버그 리포트를 분류하는 모델을 학습하고 예측 성능을 평가한다. 본 논문은 Bag of Words로 피쳐를 표현하고 Linear Discriminant Analysis를 이용하여 피쳐의 차원을 축소한 후 이를 Multi-Layer Perceptron 모델로 분류시에 가장 좋은 성능을 보인다. 이에 대한 ROC-AUC(Receiver Operating Characteristic-Area Under the Curve) 값은 0.9918이고 MCC(Matthews Correlation Coefficient)가 0.9811로 가장 좋은 성능을 보인다. 따라서 정확도가 높은 Configuration 버그 리포트 예측을 통해, 이를 사용하는 엔지니어들의 버그 수정 프로세스의 시간을 단축시킬 수 있다.

1. 서론

오늘날 급격하게 변화하는 대규모 소프트웨어 시스템들은 실패(failure)가 발생한다[2]. 이러한 영향을 완화하기 위해 롤백이나 복구 메커니즘을 널리 채택하는데 이러한 메커니즘은 메모리 버그와 같은 개별 시스템 실패를 처리하는 데는 성공하지만 Configuration 버그를 제어하는 데는 효과적이지 않다[3].

Configuration 버그는 소프트웨어 실패의 주요 원인 중 하나이다. Configuration 버그는 대규모의 소프트웨어 시스템의 버그를 고치는 시간을 지연시키는 주요한 원인 중 하나이다[3]. 또한 연구에 따르면 버그의 최대 31%가 소프트웨어 시스템의 Misconfiguration과 관련이 있다[4]. 그리고 최대 85.5%의 Misconfiguration은 Configuration 옵션의 설정 실수로 인한 것이다[5]. 또한, Configuration 버그는 시스템의 가동 중단 원인 중 하나로, 이러한 버그가 서비스 수준 장애의 약 28%를 차지한다[6]. 페이스북의 예로는 500만명의 사용자들이 Configuration 버그로 인해 페이스북 웹 사이트에 몇 시간 동안 접근하지 못했다[7]. 이러한 Configuration 버그들은 미리 식별하지 않으면 계속해서 잠재적인 오류를 가지고 있을 수 있으며 이러한 문제를 빠르게 인지하는 것이 중요하다[8].

Configuration 버그란 구성 관리를 위한 환경 변수에 의해 발생하는 버그를 의미하며 Configuration 파일을 포함하고 있다. 본 연구에서는 버그 리포트에 자연어로 기술된 부분을 텍스트 마이닝을 사용하여 분류기 모델을 구축한다. Configuration 또는 Non-Configuration 라벨이 있는 버그 리포트를 모델에 훈련시켜 버그 리포트가 Configuration 버그 리포트인지, Non-Configuration 버그 리포트 인지 분류한다.

버그 리포트의 자연어로 된 설명의 용어의 수는 많고, 이는 버그 리포트의 라벨 예측에 좋지 않은 영향을 초래할 수 있다. 따라서 본 연구에서는 피쳐 엔지니어링 기법으로 BoW(Bag of Word), TF-IDF(Term Frequency-Inverse Document Frequency), 워드 임베딩(Word Embedding) 방법을 사용하고 차원 축소를 위한 LDA(Linear Discriminant Analysis)와 PCA(Principal

Component Analysis) 방법을 사용한다. Xia et al.[1]는 BoW를 이용 후 피쳐 선택 방법을 이용한 방법을 제안하는데, 본 논문에서는 워드 임베딩 또는 차원 축소 기법을 실험하여 피쳐 선택 방법과 비교한다. 본 연구에서는 분류 모델로 SVM(Support Vector Machine)과 MLP(Multi-Layer Perceptron)을 사용한다.

본 논문에서의 데이터셋은 Xia et al.의 연구에서 선정한 오픈소스 프로젝트들을 선택하여 수집하고 버그 리포트들을 직접 분석하고 라벨링하는 과정을 가진다.

분류 모델들의 성능 평가로는 Accuracy, Configuration F-measure, Non-Configuration F-measure, ROC-AUC(Receiver Operating Characteristic-Area Under the Curve), MCC(Matthews Correlation Coefficient)를 사용한다. 실험 결과로는 BoW로 피쳐를 표현한 후 LDA를 이용해 차원을 축소한 피쳐로 MLP 모델을 사용할 때 가장 좋은 성능을 보인다. 이를 통해 Xia et al.에서 제안한 방법인 피쳐 선택 기법인 Chi-square와 Information Gain보다 피쳐 차원 축소 방법이 더 나은 것을 보여준다. 또한, 본 논문에서 차원 축소를 적용한 기법이 워드 임베딩 기법을 사용한 분류 모델 보다 더 좋은 성능을 보임을 보인다.

2. 관련 연구

Configuration 버그에 대한 많은 연구가 있었다[9], [10], [11], [12], [13], [14]. Attariyan과 Flinn은 데이터 흐름 분석을 활용하여 Configuration 버그를 추적하는 ConfAid를 개발하였다[9]. Zhang과 Ernst는 통계 분석을 활용하여 Configuration 오류를 식별하는 ConfDiagnoser라는 정적 분석과 동적 프로파일링을 결합해 Configuration 버그의 루트를 식별하고자 하는 노력을 기울였다[11]. Arshad et al.은 GlassFish와 JBoss로부터 Configuration 버그를 추출하고 해당 버그의 특징을 problem-type, problem-time과 같은 관점에서 나타내었다[13].

Jindal은 버그 리포트의 우선순위 중 하나인 심각성(Severity)를 다루었다[15]. 이 연구에서는 기계학습 기법과 피쳐 선택 기법을 사용하여 소프트웨어 버그 리포트의 텍스트 정보를 이용해 심각성을 예측하고자 하였다.

Xia et al.[1]의 논문에서는 Configuration 버그 리포트와 Non-configuration 버그 리포트를 키워드들을 기반으로 예측하는 접근을 하였다. 버그 리포트의 자연어 데이터를 BoW로 표현하고 피쳐 선택 기법들(Chi-square, Information Gain)을 이용한다. 그리고 피쳐들을 기계학습 모델들을 이용해 해당 버그 리포트가 Configuration 버그 리포트인지 아닌지 예측한다.

CoLUA를 소개한 Wen et al.[14]은 Mozilla, Apache, MySQL 3가지의 프로젝트로부터 Configuration 옵션들을 추출하여 Configuration 버그를 예측하는데, Xia et al.이 제안한 방법과 마찬가지로 Chi-square와 Information Gain을 이용하여 피쳐 선택을 한다. 그리고 기계학습 모델들을 사용하여 버그 리포트들을 예측한다.

3. 배경지식

3.1 Bag of Words

단어의 순서는 고려하지 않고, 단어의 출현 빈도를 기반으로 표현되는 방법이다. 본 연구에서는 전처리 과정을 거친 버그 리포트의 자연어 부분을 BoW(Bag of Words)로 표현하여 3,926개의 버그 리포트 데이터에서 21,130개의 벡터 크기로 나타난다. 본 연구에서는 피쳐 선택(Chi-square 또는 Information Gain) 또는 차원 축소(PCA 또는 LDA)로 이 크기를 줄이는 과정을 가진다.

3.2 TF-IDF

TF-IDF(Term Frequency-Inverse Document Frequency)는 단어가 문서에서 얼마나 중요한지를 반영하는 통계적 방법이다. 이는 TF(Term Frequency)와 DF(Document Frequency)의 역을 곱한 것이며 식(1)과 같이 표현한다. 여기서 TF(d,t)는 문서 d에 용어 t가 나타나는 빈도를 나타내며, N은 문서의 수이고 df(t)는 코퍼스에서 용어 t를 포함하는 문서 수이다.

$$W(d,t) = TF(d,t) \cdot \log \frac{N}{df(t)} \quad (1)$$

3.3 워드 임베딩

워드 임베딩은 단어를 벡터로 표현하는 것을 말한다. 희소 행렬 보다는 밀집된 표현으로 변환하는 방법이다. 데이터를 낮은 차원으로 변환하여 데이터 간의 관계가 성립되도록 처리하는 과정이다. 본 연구에서는 워드 임베딩 기법으로 Word2Vec, GloVe, FastText 방법을 사용한다.

Word2Vec. Word2Vec은 단어간 유사도를 반영하여 단어의 의미를 벡터화 할 수 있는 방법이다. 단어를 벡터로 변환해서 다음에 올 단어를 예측하는 모델이다.

GloVe. GloVe(Global Vectors for Word Representation)[16]는 카운트 기반과 예측 기반을 모두 사용하는 방법론으로 단어의 동시 등장 여부의 정보 값을 가지고 있다. Word2Vec과 GloVe 모델은 작동 방식이 매우 유사하며, 둘 다 문맥과 의미에 따라 주변 단어의 영향을 받는다는 공통점이 있다. 하지만 Word2Vec은 동시 출현 단어 쌍의 개별로 시작되지만, GloVe는 코퍼스의 모든 단어에서 전체 집계된 동시 출현에 관한 통계로 시작된다는 차이가 있다.

FastText. FastText는 텍스트 분류를 수행하는 프레임 워크이다. Word2Vec 모델은 일반적으로 각 단어의 형태적 구조를 무시했다면, FastText 모델은 각 단어를 Bag of Character n-gram으로 간주한다[17]. 따라서 문자에 따라 개별 단어에서 n-gram을 활용하는 효과로 인해 희소한 단어가 잘 표현된다. 또한 훈련데이터에 단어가 나타나지 않더라도 Character n-gram에서 단어에 대한 벡터를 구성할 수 있다. 반면에 Word2Vec과 GloVe는 이를 수행할 수 없다.

3.4 차원 축소

본 연구에서는 차원 축소 기법으로 PCA와 LDA를 사용한다.

Principal Component Analysis. PCA(Principal Component Analysis)는 차원 축소 방법의 대표적인 방법이다. 이는 사영 된 데이터의 분산을 최대화하는 행렬을 찾는 문제이다. 즉, 상관 관계가 없는 새로운 변수를 찾고 최대한 많은 변동성을 유지하기 위해 분산을 최대화하는 것이다.

Linear Discriminant Analysis. LDA(Linear Discriminant Analysis)는 통계, 패턴 분석, 기계학습에서 주로 쓰이는 방법이다[18]. 이는 클래스들을 최대한 잘 분리시키는 사영을 찾는다. 같은 클래스에 속하는 데이터들의 분산은 최대한 줄이고, 각 데이터들의 평균값들의 분산은 최대화하여 클래스들간의 거리를 멀리 떨어지게 한다.

3.5 분류 모델

Support Vector Machine. SVM(Support Vector Machine)은 학습 데이터를 비선형 매핑을 통해 고차원으로 변환한다. 각 트레이닝 버그 리포트는 다차원 공간에서 각 용어가 차원을 나타내는 점으로 표시된다. SVM은 각 라벨에 대한 서포트 벡터로 버그 리포트를 선택하고, 경계를 형성하기 위해 선형 또는 비선형 판별 함수를 작성한다.

Multi-Layer Perceptron. MLP(Multi-Layer Perceptron)는 입력층과 출력층 사이에 하나 이상의 중간층이 존재하는 신경망으로 퍼셉트론의 결과값과 실제 결과값을 비교한 뒤, 오류를 전파 하는 원리는 단층 퍼셉트론과 동일하다.

4. 접근 방법

그림 1은 본 논문의 전체적인 접근 방법을 나타낸다. 마이닝한 버그 리포트들을 입력 데이터로 사용하여 텍스트 마이닝 기법들을 이용하여 모델을 구축하는 것이 목표이다. 트레이닝 단계에서는 모델을 훈련하고 테스팅 단계에서는 학습된 모델을 통해 Configuration 버그 리포트와 Non-Configuration 버그 리포트를 예측한다.

먼저 수집한 데이터들에서 피쳐를 추출하는 단계부터 시작한다. 버그 리포트는 자연어인 요약문(Summary)과 설명문(Description) 항목을 포함하고 있다. 요약문과 설명문에 해당하는 데이터들을 단계1을 통해 필요 없는 텍스트들을 처리하는 과정을 가진다.

단계2에서는 피쳐 엔지니어링 기법을 이용하여 토큰화된 데이터들을 BoW, TF-IDF 또는 워드 임베딩으로 표현한다. 그리고 BoW와 TF-IDF를 사용한 경우에는 PCA 또는 LDA를 이용해 피쳐의 차원을 축소하는 과정을 가진다. 워드 임베딩의 경우 본 논문에서 Word2Vec, GloVe, FastText를 사용한다.

단계3에서는 단계2에서 표현한 데이터들을 지도 학습 알고리즘에 입력하여 학습한다. 본 논문에서는 SVM과 MLP, NBM(Naïve Bayes Multinomial)을 사용한다.

단계4에서 학습된 분류기 모델을 통해 테스트 데이터들이 Configuration 버그 리포트인지 아닌지 예측한다. 이때 입력되는 데이터들 또한 학습 데이터와 동일한 전처리와 피쳐 엔지니어링 과정을 거친다. 예측된 테스트 데이터와 실제 테스트 데이터의 라벨을 비교하여 마지막 단계5에서 모델의 예측 성능을 평가한다.

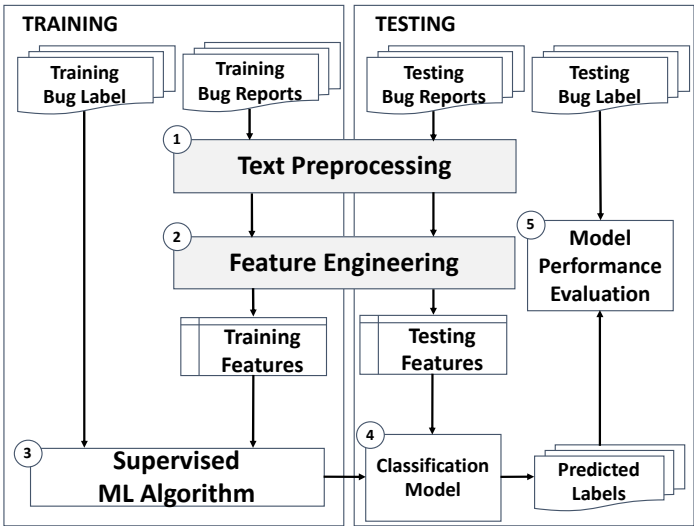


그림 1. 접근 방법

4,1 버그 리포트 데이터셋 구축

JIRA 시스템으로부터 총 5개의 오픈 소스 프로젝트들의 버그 리포트들을 수집한다. ACCUMUIO, ACTIVEMQ, CAMEL, FLUME, WICKET의 프로젝트에서 버그 리포트들을 수집한다. 각 프로젝트들의 버그 리포트에 해당하는 커밋 로그 데이터(Commit Log Data)를 수집한다. 그리고 실제 버그를 수정한 기록이 있는 리포트들로 필터링한다. 즉, 버그 리포트와 동일한 커밋(commit)에 의해 변경된 파일들이 있는지 확인한다. 또한, 설명문이

없는 리포트들을 제거하여 총 3,926개의 데이터를 생성하게 된다.

해당 버그 리포트가 Configuration 버그인지 아닌지 라벨링하기 위해 직접 분석하는 과정을 가진다. 이 과정에서 커밋에 의해 변경된 파일과 소스코드, 버그 리포트의 설명문을 분석한다. 분석 결과 348개의 Configuration 버그 리포트와 3,578개의 Non-Configuration 버그 리포트로 라벨링된 데이터셋을 구축하게 된다.

4.2 데이터 전처리

버그 리포트의 요약문과 설명문은 자연어이기에 필요 없는 텍스트가 존재한다. 설명문에 포함된 자연어가 아닌 코드 조각들을 정규표현식(regular expression)을 이용하여 제거한다. 그리고 텍스트들을 토큰화하고 'the', 'a', 'an'과 같은 불용어들을 제거한다. 단어들을 소문자로 모두 변환하고 특수문자들을 제거하고 어근화한다.

4.3 피쳐 엔지니어링

데이터 전처리 과정을 거친 데이터를 토대로 이들을 BoW, TF-IDF, Word2Vec, GloVe, FastText 형태로 나타낸다. BoW로 표현될 경우, 3,926개의 데이터에서 21,130의 크기로 나타난다. 피쳐 선택 또는 차원 축소(PCA 또는 LDA)로 이 크기를 줄이는 과정을 가진다. BoW으로 표현하고 Chi-square 또는 Information Gain을 사용하여 피쳐의 크기를 줄이는데, 상위 10%만을 이용한다. 또한 차원 축소 방법인 PCA 또는 LDA를 사용할 경우에는 BoW 뿐만 아니라 TF-IDF로도 표현하여 비교한다.

워드 임베딩 방법들 중 Word2Vec, FastText는 모두 2,000개의 피쳐로 설정하고 중심 단어에서 주변 단어를 예측하는 Skip-gram 메커니즘을 사용하여 실험한다. GloVe로 표현할 때는 384개의 피쳐로 생성된다.

4.4 차원 축소

BoW 또는 TF-IDF 형태로 생성된 데이터들을 차원 축소를 하는데, PCA의 경우 2,000개의 피쳐로 축소하여 사용한다. LDA는 본 연구에서의 데이터셋의 클래스가 두 개이기 때문에, 하나의 피쳐로 축소된다[18].

4.5 분류 모델

각 피쳐 선택 기법 또는 차원 축소를 진행한 데이터들을 SVM, MLP 모델을 사용하여 학습하고 해당 분류 모델의 성능을 평가한다. BoW로 표현한 피쳐를 피쳐 선택 기법을 사용한 경우, 피쳐의 값들이 음수가 아니기에 Xia et al.에서 제안한 모델인 NBM을 실험하여 비교한다.

5. 실험 및 결과

5.1 연구 질문

RQ1 *본 논문에서 제안하는 방법이 Xia et al.의 연구에서 제안하는 방법보다 얼마나 효과가 있는가?*

H_0 본 논문에서 제안하는 방법이 Xia et al.에서 제안하는 방법보다 성능이 같거나 낮다
H_A 본 논문에서 제안하는 방법이 Xia et al.에서 제안하는 방법보다 성능이 높다

본 논문에서는 세 가지 워드 임베딩 또는 두 가지 차원 축소를 이용한 두 가지 분류 모델을 제안한다. 따라서 총 10 가지의 조합 중에서 가장 좋은 성능을 실험한다. Xia et al.와 동일한 오픈 소스 프로젝트들의 버그 리포트들을 추출하여 데이터셋을 구성하였기에 Xia et al.이 제안한 피쳐 선택 기법 두 가지를 적용한 모델을 함께 비교한다.

이를 위해 귀무 가설로 위의 H_0을 정의하고 대립 가설인 H_A를 채택 할 수 있는지 분석한다. 이에 대한 분석은 5.4.1절에서 관련 연구와 본 논문에서 제안하는 방법을 사용한 두 모델을 비교하여 통계적 검정을 한다.

RQ2 *차원 축소의 효과는 워드 임베딩과 비교할 때 어느 정도 효과가 있는가?*

H_0 차원 축소 기법이 워드 임베딩 기법보다 성능이 같거나 낮다
H_A 차원 축소 기법이 워드 임베딩 기법보다 성능이 높다

본 논문에서는 워드 임베딩 방법으로 Word2Vec, GloVe, FastText를 사용하고 차원 축소 기법으로 PCA와 LDA를 사용한다. 이 때 본 연구에서는 빈도수 기반(BoW, TF-IDF)으로 표현된 피쳐들을 차원 축소한 결과가 워드 임베딩보다 좋은 효과가 있음을 가설을 통해 분석한다.

귀무 가설로 위의 H_0와 대립 가설로 H_A를 정의하여 귀무 가설을 기각 할 수 있는지 분석한다.

5.2 실험 환경

실험은 3.1 GHz CPU, 8 GB Ram의 MacOS 환경에서 Python 3.7을 이용하여 진행한다. 그리고 4.1절과 4.2절에서 구축한 Configuration 버그 리포트 데이터셋의 개수와 각 프로젝트의 버그 리포트 기간은 표1과 같다. 실험에서는 전체 프로젝트의 버그 리포트 3,926개를 모두 사용하여 실험한다. 데이터를 전처리하는데 Pandas, re, NLTK(Natural Language

Toolkit) 패키지를 사용한다. 전처리하는 과정에서 NLTK가 정의한 영어 불용어 리스트를 통해 불용어들을 제거하고 re 패키지를 이용하여 소문자로 변환하고 특수문자와 코드 조각들을 제거한다.

표1. 수집한 프로젝트들의 이름과 해당 기간, 버그 리포트 개수와 Configuration 버그 리포트의 개수

Project	Time	Bug Reports	Config. Bug Reports
ACCUMULO	2011.10~2017.09	532	55
ACTIVEMQ	2009.01~2018.10	913	81
CAMEL	2009.01~2018.10	1249	158
FLUME	2010.06~2017.07	336	40
WICKET	2009.01~2018.07	896	14
Total		3,926	348

본 논문에서의 분류 문제는 클래스의 분포에서 큰 불균형이 나타난다. 불균형한 데이터들을 다루기 위해 Stratified K-fold cross validation을 사용하며 계층화된 샘플링을 사용해 각 fold 에서 클래스의 빈도수가 유지되도록 한다. 해당 실험에서는 데이터셋을 10개의 fold로 나누어 하나의 fold는 테스트셋으로 나머지 9개의 fold는 트레이닝셋으로 사용한다. 이러한 방법을 10 번 반복하며 모델 성능을 평균 내어 기록한다. 모델들은 Scikit-Learn 패키지를 이용하여 SVM과 MLP, NBM을 사용한다.

5.3 평가 척도

표1에서 본 논문에서의 데이터셋은 불균형한 클래스를 가지고 있음을 알 수 있다. 불균형한 데이터 때문에 Accuracy와 F-measure만으로 모델의 성능을 평가하기에는 적절하지 못하다. 그래서 ROC-AUC Score와 MCC 두 척도를 사용한다.

5.3.1 F-measure

F-measure는 분류기 모델이 얼마나 정확하게 분류를 하는가를 판단하는 평가 척도 중 하나이다. 이를 위해 precision과 recall을 알아야 하며 F-measure는 두 값의 조화평균이다[19]. Configuration F-measure와 Non-configuration F-measure는 식(2), (3)과 같이 정의한다.

$$F(C) = \frac{2 \cdot Precision(C) \cdot Recall(C)}{Precision(C) + Recall(C)} \quad (2)$$

$$F(NC) = \frac{2 \cdot Precision(NC) \cdot Recall(NC)}{Precision(NC) + Recall(NC)} \quad (3)$$

5.3.2 ROC-AUC

ROC-AUC(Receiver Operating Characteristic-Area Under the Curve) 값은 ROC 곡선 아래에 해당되는 면적을 의미하는 것이며 불균형한 데이터셋에 적합하다[20]. ROC 곡선은 모든 분류 임계 값에서 분류 모델의 성능을 보여주는 그래프인데 이 그래프는 TPR(True Positive Rate)와 FPR(False Positive Rate)로 구성된다. TPR과 FPR은 식(4),(5)와 같이 표현된다. AUC는 ROC 전체 곡선의 2차원의 면적을 계산하는데 범위는 0에서 1사이다. 예측이 100% 잘못된 모델은 0.0의 AUC 값을 가지게 되고 예측이 100% 옳은 모델은 1.0의 값을 가지게 된다.

$$TPR = \frac{TP}{TP + FN} \quad (4)$$

$$FPR = \frac{TP}{FP + TN} \quad (5)$$

5.3.3 MCC

MCC(Matthews Correlation Coefficient)는 불균형한 클래스의 분류에 적합한 평가 척도이다[21]. MCC는 식(8)과 같이 정의하며 -1에서 +1 사이의 값을 갖는다. 0일 경우 랜덤한 예측이라 판단하며 1일 경우 완벽한 예측, -1일 경우 예측이 전혀 안된다고 할 수 있다.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \quad (6)$$

5.4 실험 결과

5.4.1 *RQ1: 본 논문에서 제안하는 방법의 성능*

Xia et al.에서는 Information Gain 또는 Chi-square를 이용하여 피쳐들을 선택한다[1]. 하지만 이 경우 여전히 방대한 희소 행렬 문제가 있다. 이에 반해 본 논문에서는 이를 해결하기 위해 워드 임베딩 또는 차원 축소 방법을 이용한다.

표2와 표3은 Xia et al.의 방법을 사용하여 실험 한 결과이다. 표3은 BoW로 표현한 각 피쳐들의 Chi-square를 계산한 뒤 상위 10%의 피쳐들을 사용해 구축한 각각의 분류 모델에 대한 평가 척도들이다. 표3 또한 Information Gain을 토대로 피쳐의 10%만을 사용하여 학습된 각 모델들의 분류 성능을 나타낸다. 표3에서의 BoW+NBM이 ROC-AUC 값 0.8748, MCC 값 0.7061로 Xia et al.가 제안한 방법 중에서 가장 높은

성능을 보여준다.

본 논문에서 제안하는 차원 축소를 사용한 실험 결과는 표4와 표5에 나타나 있다. 표4에서는 PCA를 이용한 차원 축소 결과와 표5에서는 LDA를 이용한 실험 결과를 보여준다. LDA를 사용한 성능이 가장 좋은 성능을 보여주며, ROC-AUC와 MCC 값은 모두 0.9 이상이다. TF-IDF보다는 BoW일 때 조금 더 높은 ROC-AUC와 MCC 값을 가지며 MLP와 SVM 두 모델 모두 비슷하다. ROC-AUC 값은 각각 0.9918, 0.9893이고 MCC는 각각 0.9811, 0.9826 이다. 그리고 또한 각각에 대한 Config. F-measure는 0.983, 0.984이고 Non-Config. F-measure 는 0.998, 0.998의 결과가 나타난다.

"본 논문에서 제안하는 방법이 Xia et al.에서 제안하는 방법보다 성능이 같거나 낮다"인 귀무 가설을 분석하기 위해, BoW+NBM 모델과 LDA를 이용한 BoW+MLP 모델을 비교한다. 그리고 각 모델의 ROC-AUC에 대한 성능 차이를 대응 표본 윌콕슨 랭크섬 검정(Paired Sample Wilcoxon Test)을 통해 통계적 유의성을 검정한다. 그 결과는 표7에서와 같이 유의 수준 5%에서 p-value 값은 2.94×10^{-3}으로 유의 확률인 0.05보다 낮기 때문에 ROC-AUC의 차이가 통계적으로 유의함을 보인다. 이에 대한 효과 크기(Effect Size)는 Cohen's D의 값이 2.13이다. 이는 Sawilowsky에 따르면 D 값이 2.0보다 크기 때문에 영향이 크다는 것을 알 수 있다[22]. MCC에 대한 성능 차이 또한 9.77×10^{-4}으로 두 모델의 성능 차이가 통계적으로 유의함을 보이며, 효과 크기는 3.75로 효과가 크다는 것을 알 수 있다.

효과 크기는 D 값이 2.0보다 큰 값을 가지기 때문에 귀무 가설을 기각할 수 있으며, 대립 가설인 "본 논문에서 제안하는 방법이 Xia et al.에서 제안하는 방법보다 성능이 높다"를 채택할 수 있다. 이를 통해서 Xia et al.에서 제안한 피처 선택을 이용하는 것 보다 본 논문에서 제안하는 차원 축소 방법인 LDA를 사용하는 모델이 더 좋은 성능을 보인다는 것을 검정할 수 있다.

표2. Chi-square 피쳐 선택 기법을 이용한 모델에 대한 실험 결과인 척도 정보: Accuracy, F-measure, ROC-AUC, MCC

Evaluation	BoW+NBM	BoW+MLP	BoW+SVM
Accuracy	0.949	0.934	0.92
Conf. F-measure	0.732	0.49	0.484
Non-Conf. F-measure	0.972	0.965	0.957
ROC-AUC	0.8748	0.6733	0.6955
MCC	0.7061	0.4993	0.4557

표3. Information Gain 피쳐 선택 기법을 이용한 모델에 대한 실험 결과인 척도 정보: Accuracy, F-measure, ROC-AUC, MCC

Evaluation	BoW+NBM	BoW+MLP	BoW+SVM
Accuracy	0.951	0.963	0.905
Conf. F-measure	0.728	0.455	0.44
Non-Conf. F-measure	0.973	0.963	0.948
ROC-AUC	0.8587	0.6585	0.6857
MCC	0.7009	0.4628	0.3943

표4. PCA를 이용한 각 모델에 대한 Accuracy, F-measure, ROC-AUC, MCC 결과

Evaluation	TFIDF +SVM	TFIDF +MLP	BoW +SVM	BoW +MLP
Accuracy	0.908	0.936	0.91	0.93
Conf. F-measure	0.505	0.552	0.45	0.433
Non-Conf. F-measure	0.949	0.965	0.951	0.963
ROC-AUC	0.7367	0.7141	0.686	0.6462
MCC	0.4843	0.5407	0.4073	0.4427

표5. LDA를 이용한 각 모델에 대한 Accuracy, F-measure, ROC-AUC, MCC 결과

Evaluation	TFIDF +SVM	TFIDF +MLP	BoW +SVM	BoW +MLP
Accuracy	0.989	0.99	0.997	0.997
Conf. F-measure	0.937	0.945	0.984	0.983
Non-Conf. F-measure	0.994	0.995	0.998	0.998
ROC-AUC	0.9692	0.9688	0.9893	0.9918
MCC	0.9319	0.9403	0.9826	0.9811

5.4.2 *RQ2: 차원 축소를 이용한 방법과 워드 임베딩을 이용한 모델 성능 비교*

BoW 만으로 피쳐를 표현하는 경우, 희소 행렬로 인한 문제가 있다. 이를 해결하기 위해 워드 임베딩 방법들이 사용된다. 해당 연구 질문에서는 이러한 워드 임베딩 방법들과 빈도수 기반으로 표현하고 차원 축소하는 방법들을 비교한다. 빈도수 기반인 BoW 또는 TF-IDF 를 이용하여 피쳐를 표현하고 해당 피쳐들을 축소하여 희소하지 않은 형태로 만든다. 이에 대한 실험

표 6. 워드 임베딩을 이용하여 표현한 피쳐와 각 모델에 대한 Accuracy, F-measure, ROC-AUC, MCC

Model	Word2Vec+SVM	Word2Vec+MLP	FastText+SVM	FastText+MLP	GloVe+SVM	GloVe+MLP
Accuracy	0.9246	0.9319	0.9121	0.9333	0.823	0.919
Conf. F-measure	0.532	0.522	0.47	0.539	0.245	0.37
Non-Conf. F-measure	0.959	0.963	0.952	0.964	0.9	0.957
ROC-AUC	0.7247	0.7006	0.6983	0.7105	0.5979	0.625
MCC	0.4975	0.5046	0.4621	0.5173	0.237	0.3659

결과는 워드 임베딩을 사용하였을 때보다 더 나은 성능을 보여준다.

표6는 Word2Vec, FastText, GloVe의 워드 임베딩을 사용한 각각의 모델에 대한 실험 결과를 Accuracy, Configuration F-measure, Non-Configuration, ROC-AUC, MCC의 평가 척도로 모델 예측 성능을 나타낸 것이다. FastText+MLP의 MCC가 0.5173으로 워드 임베딩 방법 중에서 가장 높게 나타고 ROC-AUC는 0.7105으로 Word2Vec+SVM 다음으로 높은 값을 가진다.

귀무 가설인 “차원 축소 기법이 워드 임베딩 기법보다 성능이 같거나 낮다”를 기각할 수 있는 지 분석하기 위해, FastText+MLP 모델과 LDA를 이용한 BoW+MLP 모델의 성능을 통계적으로 비교 검정한다. 각 모델의 ROC-AUC에 대한 성능 차이를 대응 표본 윌콕스 랭크섬 검정을 통해 통계적 유의성을 검정한 결과 표7에서처럼 p-value 값이 9.77×10^{-4}으로 성능 차이에 대한 유의함을 보인다. 이에 대한 효과 크기는 6.3으로 효과가 크다는 것을 알 수 있다. MCC에 대한 성능 차이 또한 9.77×10^{-4}으로 두 모델의 성능 차이가 통계적으로 유의함을 보이며, 효과 크기는 Cohen’s D의 값이 5.13으로 효과 크기가 크다는 것을 알 수 있다

표7. RQ1와 RQ2에 대한 각각의 윌콕슨 랭크섬 검정 결과인 p-value와 효과 크기

RQ1: BoW+NMB vs. BoW+LDA+MLP		
Metrics	ROC-AUC	MCC
p-value	2.94×10^{-3}	9.77×10^{-4}
Cohen's D	2.13	3.75
RQ2: FastText+MLP vs BoW+LDA+MLP		
Metrics	ROC-AUC	MCC
p-value	9.77×10^{-4}	9.77×10^{-4}
Cohen's D	6.3	5.13

2.0 보다 큰 효과 크기를 통해 귀무 가설을 기각할 수 있으며, 대립 가설인 “차원 축소 기법이 워드 임베딩 기법보다 성능이 높다”를 받아 들일 수 있다. 이를 통해 본 논문에서 제안하는 LDA를 이용한 차원 축소 기법이 워드 임베딩 기법을 이용한 방법보다 더 성능이 좋다는 것을 알 수 있다.

5.5 위협 요소

버그 리포트가 Configuration 버그 리포트인지 아닌지 직접 라벨링 한 것이 내부적인 유효성에 대한 위협이다. 이러한 위협을 완화시키기 위해 라벨링 후 데이터셋에 대해 검토 과정을 가졌다.

그리고 5개의 오픈소스 프로젝트에서 3,926개의 버그 리포트를 사용하였는데 이 외부적인 유효성에 대한 위협을 줄이기 위해 좀 더 많은 데이터를 수집하여 분석할 계획이다.

불균형한 클래스의 데이터셋 또한 유효성에 대한 위협이다. 이 위협을 줄이기 위해 F-measure 보다는 불균형한 클래스를 가진 데이터셋에 적합한 ROC-AUC와 MCC를 채택하였다. 또한, 이러한 위협을 줄이기 위해서 Stratified 10-fold 교차 검증을 사용하였다. 또한 향후에 불균형한 데이터를 다루기 위한 기법들을 추가하여 적용할 계획이다.

6. 결론 및 향후 연구

본 논문은 5개의 오픈소스 소프트웨어 프로젝트로부터 3,962개의 버그 리포트를 추출하고 직접 라벨링 하는 과정을 거쳐 데이터셋을 생성하였다. 텍스트 마이닝, 피쳐 엔지니어링과 차원 축소 기법들을 이용해 Configuration 버그 리포트를 예측하는 분류 모델을 구축하였다. 그리고 해당 모델에 대한 예측 성능을 불균형한 클래스를 가진 데이터셋에 적합한 ROC-AUC와 MCC로 평가한다. 피쳐 엔지니어링 기법으로 BoW, TF-IDF, Word2Vec, FastText, GloVe를 사용한다. 분류 모델로는 SVM과 MLP를 사용하여 실험한다. 본 논문에서는 워드 임베딩 방법과 차원 축소 방법 그리고

Xia et al.[10]에서 제안한 방법들을 실험하고 비교한다. 본 논문에서 차원 축소 기법을 사용하여 제안하는 모델 중에서는 BoW로 피쳐를 표현하고 LDA를 이용하여 피쳐의 차원을 축소한 후 이를 MLP와 SVM에 사용한 모델의 예측 성능이 가장 뛰어난 성능을 보인다. 각각에 대한 ROC-AUC 값은 0.9918, 0.9893이고 MCC가 0.9811, 0.9826이다. 그리고 연구질문 RQ1의 통계 검정 결과, 효과크기가 크기 때문에 본 논문에서 제안하는 기법이 Xia et al.의 방법보다 더 성능이 좋다는 것을 보인다. RQ2에서의 효과 크기가 크다는 것을 통해 워드 임베딩을 이용한 방법보다 빈도 기반으로 추출한 피쳐를 차원 축소 기법을 이용하는 방법이 더 성능이 좋다는 것을 보인다.

정확한 Configuration 버그 리포트 예측을 통해, Configuration 버그라는 것을 인식하는데 시간이 단축될 수 있다. 이는 엔지니어들이 버그 수정 프로세스에서 낭비하는 시간을 줄일 수 있는 기대효과가 있다.

향후 연구로 불균형 데이터 문제를 다루는 기법들을 적용해 볼 필요가 있다. 분류 모델로써 딥 뉴럴 네트워크 모델들을 사용하여 실험할 계획이다.

Acknowledgement

이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2019R1G1A1005047)

참고 문헌

[1] X. Xia, D. Lo, W. Qiu, X. Wang, and B. Zhou, “Automated configuration bug report prediction using text mining,” *Proc. - Int. Comput. Softw. Appl. Conf.*, pp. 107–116, 2014.

[2] B. MAURER, “Fail at Scale: Reliability in the Face of Rapid Change,” *Commun. oft he ACM58*, no. 11, pp. 44–49, 2015.

[3] D. Oppenheimer, D. a. Patterson, and A. Ganapathi, “Why do Internet Services fail, and what can be done about it?,” *Proc. 4th Conf. USENIX Symp. Internet Technol. Syst. 4*, 2003.

[4] Z. Yin, X. Ma, J. Zheng, Y. Zhou, L. N. Bairavasundaram, and S. Pasupathy, “An empirical study on configuration errors in commercial and open source systems,” in *SOSP’11 - Proceedings of the 23rd ACM Symposium on Operating Systems Principles*, 2011.

[5] H. J. Wang, J. C. Platt, Y. Chen, R. Zhang, and Y. Wang, “Automatic Misconfiguration Troubleshooting with PeerPressure,” *Symp. A Q. J. Mod. Foreign Lit.*, 2004.

[6] L. A. Barroso and U. Hölzle, “The datacenter as a computer: An introduction to the design of warehouse-scale machines,” *Synth. Lect. Comput. Archit.*, 2009.

[7] “More Details on Today’s Outage | Facebook.” [Online]. Available: https://www.facebook.com/notes/facebook-engineering/more-details-on-todays-outage/431441338919/. [Accessed: 03-Jan-2020].

[8] T. Xu *et al.*, “Early Detection of Configuration Errors to Reduce Failure Damage This paper is included in the Proceedings of the,” in *Osdi*, 2016, pp. 619–634.

[9] M. Attariyan and J. Flinn, “Automating configuration troubleshooting with dynamic information flow analysis,” *Design*, 2010.

[10] X. Xia, D. Lo, W. Qiu, X. Wang, and B. Zhou, “Automated configuration bug report prediction using text mining,” *Proc. - Int. Comput. Softw. Appl. Conf.*, pp. 107–116, 2014.

[11] S. Zhang and M. D. Ernst, “Automated diagnosis of software configuration errors,” in *Proceedings - International Conference on Software Engineering*, 2013.

[12] T. Xu *et al.*, “Do not blame users for misconfigurations,” in *SOSP 2013 - Proceedings of the 24th ACM Symposium on Operating Systems Principles*, 2013.

[13] F. A. Arshad, R. J. Krause, and S. Bagchi, “Characterizing configuration problems in Java EE application servers: An empirical study with GlassFish and JBoss,” in *2013 IEEE 24th International Symposium on Software Reliability Engineering, ISSRE 2013*, 2013.

[14] W. Wen, T. Yu, and J. H. Hayes, “CoLUA: Automatically Predicting Configuration Bug Reports and Extracting Configuration Options,” in *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, 2016.

[15] R. Jindal, R. Malhotra, and A. Jain, “Prediction of defect severity by mining software project reports,” *Int. J. Syst. Assur. Eng. Manag.*, vol. 8, no. 2, pp. 334–351, 2017.

[16] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global vectors for word representation,” in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014.

[17] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of tricks for efficient text classification,” in *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, 2017.

[18] A. Tharwat, T. Gaber, A. Ibrahim, and A. E. Hassanien, “Linear discriminant analysis: A detailed tutorial,” *AI Commun.*, 2017.

[19] J. Han and M. Kamber, *Data Mining Concept and Tehniques*. 2006.

[20] J. Davis and M. Goadrich, “The relationship between precision-recall and ROC curves,” in *ACM International Conference Proceeding Series*, 2006.

[21] S. Boughorbel, F. Jarray, and M. El-Anbari, “Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric,” *PLoS One*, 2017.

[22] S. S. Sawilowsky, “New Effect Size Rules of Thumb,” *J. Mod. Appl. Stat. Methods*, 2009.

BERT를 이용한 중복 버그 보고서 검색 모델

문석암, 강민구, 류덕산, 김순태
전북대학교
mon823@jbnu.ac.kr, kang.mingu94@gmail.com, duksan.ryu@jbnu.ac.kr, stkim@jbnu.ac.kr

Duplicate Bug Report Retrieval Model Using BERT

Seokam Moon, Mingu Kang, Duksan Ryu, Suntae Kim
Jeonbuk National University

요약

중복 버그를 탐지하는 것은 새로 보고된 버그가 기존 버그의 중복인지 확인하고 원본 또는 유사한 버그를 이전 데이터에서 검색하는 것이다. 이것은 비용이 많이 드는 중복작업을 피하기 위해서 필요하다. 일반적으로 버그 보고서의 중복은 수천, 수백개가 될 정도로 많은 중복 버그가 나타나고 있고 따라서 여기에 드는 비용이 많이 든다. 지금까지 다양한 접근 방법들이 연구되고 성능이 발표되었지만 성능이 더욱 증가될 수 있다. 본 논문에서는 중복 및 유사한 버그를 검출하기 위해서 pre-training 모델인 BERT를 기반으로 한 Siamese 네트워크 통해 검색 모델을 제안한다. BERT는 pre-training 모델로 자연어 부분에서 비교적 적은 데이터로도 높은 성능을 기대할 수 있다. 우리는 94%에 가까운 $Accuracy$와 $Recall$비율을 보고하여 높은 성능으로서 보다 중복 버그 보고서를 더 잘 탐지하고 분류할 수 있을 것이다.

1 서론

소프트웨어 유지보수는 소프트웨어 개발 생명 주기의 주요 부분이다. 유지보수 활동은 소프트웨어 시스템의 수명 주기 비용의 2/3 이상을 차지한다.[1] 모든 대형 소프트웨어 프로젝트는 버그질라와 같은 버그 추적 시스템을 호스트하여 버그와 그 해결책을 관리하고 추적한다. 테스트 실무자와 소프트웨어 시스템의 최종 사용자는 시스템의 버그를 정형화되지 않은 방식으로 보고하게 되고 중복 버그를 보고한다. 더하여 이전에 발생한 수정되었던 버그는 나중에 동일하거나 다른 이유로 다시 나타날 수 있다. 두 시나리오에서 이전에 보고된 유사한 버그에 대한 버그 매핑은 필수적이다. 이러한 매핑은 버그를 수정하는 엔지니어의 불필요한 재작업을 방지하고 적절한 버그 우선순위 설정을 수행하는 데 도움이 될 것이다. 또한 유사한 버그 식별은 지원 엔지니어가 과거 분석에서 해당 버그 수정을 식별하여 생산성을 높이는 데 도움이 될 수 있다. 중복되거나 유사한 버그를 수동으로 식별하는 것은 시간이 오래 걸리기 때문에 비용이 많이 든다. ThunderBird2와 같은 대형 프로젝트에서 중복 버그 비율은 보고된 총 버그 중 거의 50%를 차지한다.[2] 따라서 중복된 버그 식별의 자동화와 유사한 과거 버그의 검색은 소프트웨어 유지관리에 있어 중요하다.

따라서 중복 버그의 식별을 자동화하고 기계학습 및 정보검색 학습기법을 사용하여 유사한 버그를 검색하기 위한 여러 접근 방식이 제시되어 있다.[3][4]

최근 트랜스포머 기반의 언어 모델인 BERT (이하 BERT, Bidirectional Encoder Representations from Transformers)를 이용하는 경우가 많아졌다. 자연어에 대해서 pre-training 되어 있어 원하는 분야에 대해 fine-tuning을 진행하여 다양한 자연어처리 분야에 적용시킬 수 있다.

본 연구에서는, 중복과 비슷한 버그에 대한 검출 및 검색 정확성을 향상하기 위해서 자연어 처리 부분에서 pre-training 모델로서 비교적 적은 데이터 세트로 더 좋은 성능을 기대할 수 있는 BERT를 활용하고, Siamese 네트워크를 적용한 모델을 제안한다.

본 논문의 주요 기여는 다음과 같이 요약할 수 있다.

1) 제안한 우리 모델은 93%에 가까운 $Accuracy$와 94%에 가까운 Recall 비율을 보여준다. Recall 비율은 같은 데이터 세트 연구 들 중 가장 높은 수치이다.
2) 제안된 모델은 중복 버그 탐지 연구분야에 최초로 BERT와 Siamese네트워크를 적용한다. BERT는 자연어 처리에서 가장 좋은 성능을 보여주며 비교적 적은 데이터로도 높은 성능을 보일 수 있다.

2 배경 지식 및 관련 연구

버그 보고서는 여러 가지 기능을 제공한다. 결함을 제기하고, 기능을 제안하고, 유지보수 작업을 기록하는 데 사용된다. 버그 보고서는 여러 개의 필드로 구성된다.

일반적으로 소프트웨어 프로젝트에서 버그 추적 시스템은 모든 최종사용자가 접근할 수 있다. 버그가 나타나면, 사람들은 버그의 세부사항을 설명하는 버그 보고서를 제출할 수 있으며, 이전 버그 보고서와 동일한 버그에 해당할 경우 이를 중복버그 보고 문제라 한다. 처음 발견된 버그의 보고서는 마스터라고 하며 다른 중복 버그 보고서를 복제라고 부른다.

2.1 BERT

BERT는 트랜스포머(Transformer) 기반의 모델이며 트랜스포머의 인코딩 부분만 사용된다. 트랜스포머는 셀프 어텐션을 바탕으로 하는데 양방향(Bidirectional)의 언어표현이 가능하다.[5] 또한 BERT는 pre-training 모델로 fine-tuning으로 비교적 적은 학습을 진행하여도 더 좋은 성능을 보일 수 있다. 즉 데이터가 비교적 적은 상황에서 BERT는 더욱더 좋은 성능을 보일 수 있다는 것이다.

2.2 기존 접근 방식

중복 버그 보고서에 대해서 여러 가지의 접근 방법이 있다.

초기 접근방식은 버그 보고서가 TF-IDF 용어 가중(term weighting measurement)측정을 통해 계산된 텍스트 벡터로 모델링 되는 벡터 공간 모델(Vector Space Model)에 정보 검색기법을 적용하였다.[6]

또 다른 접근 방식은 기계학습을 이용한 것으로 이진 분류기 모델을 사용하고 용어를 빈도수로 계산한 버그 보고서의 텍스트 특징에 선형 회귀 분석을 적용한다. Support Vector Machine(이하 SVM)은 Sun et al에 의해서 활용되었다. [7] SVM 분류를 훈련하기 위해, 모든 중복 버그 보고서를 긍정값으로 간주하고, 나머지 버그 보고서를 부정값으로 라벨링하여 사용한다.

기계학습 중 다른 접근 방식은 Random Forests 방식을 이용한 것으로 훈련 하는 동안에 여러 개의 의사결정 트리를 생성하는 앙상블 학습 방법으로 당시 논문에서의 여러 방법 중 가장 높은 성능을 보여주었다.[8]

다른 접근 방식은 정보 검색 기반뿐만 아니라 Topic Model을 적용한 버그탐지 모델인 DBTM(duplicate bug report detection model)을 소개하고 있다. 여기에서는 LDA(latent Dirichlet Allocation)를 확장하여 사용한다.[8]

최근 접근 방법으로는 딥러닝 방법 중 Convolutional Neural Networks (이하 CNN)과 Long Short-Term Memory (이하 LSTM)을 활용한 방법으로 중복 버그 보고서 검색 및 분류를 하였다.[3] CNN과 LSTM을 사용하여 90%에 가까운 $Accuracy$를 보여주었으며 recall 수치는 80%에 가까운 성능을 보여주었다.

기존 연구의 한계는 BERT와는 다르게 양방향(Bidirectional)성을 가지지 못하는 CNN, LSTM 등의 정적인 언어 모델을 사용한다. 본 논문에서 사용하고자 하는 BERT는 CNN을 사용하지 않고서 더 좋은 성능을 보여준 것으로 우리는 BERT를 사용하여 보다 좋은 성능을 기대할 수 있다

3 접근 방법

이 절에서는 중복 및 유사한 버그 분류 및 검색을 위해 제안된 딥러닝 학습 모델을 설명한다. 우리는 BERT를 사용하여 자연어로 된 버그 보고서를 숫자로 된 벡터로 변환한다. 제안된 모델은 이러한 숫자로 된 벡터를 가지고 분류 및 검색을 진행한다.

자연어 처리에서 기계 학습 모델을 적용할 때 처음 해야 할 것은 단어에 대해서 정확한 수치적 표현 즉 숫자표현으로 바꾸는 것이다. 단어를 숫자 벡터로 표현하기 위해 사용되는 모델들은 문맥에 대한 정보를 바탕으로 한다. 이 논문에서는 최근에 자연어 처리에서 높은 성능을 보여준 BERT를 활용하여 단어를 임베딩(embedding) 하여 진행한다.

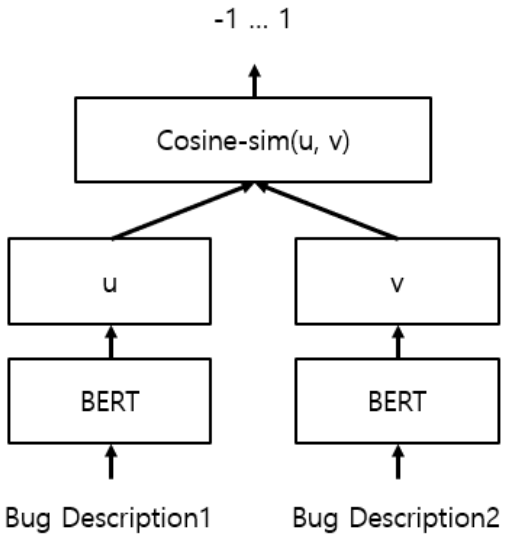
검색 모델에서의 훈련 데이터 세트는 두 개의 문장과 두 문장 사이의 관계를 보여주는 것으로 구성되어 있다. 문장은 BERT 통해 학습을 진행하게 된다. 이때 우리의 모델은 BERT를 미세조정하기 위해 Siamese 및 Triplet networks를 생성한다. 생성된 문장이 내재적으로 의미가 있고 코사인 유사도(Cosine similarity)과 비교할 수 있도록 가중치를 업데이트한다.

Triplet ranking loss는 anchor 문장을 긍정적인 문장 그리고 부정적인 문장과 비교하는 손실 함수(loss function)이다. 긍정적인 문장과의 거리는 최소화되어야 하며, 부정적인 문장과는 거리가 최대가 되어야 한다.

Siamese의 네트워크 구조는 입력 문장의 고정 벡터를 도출할 수 있도록 한다.

[그림 1]은 새로운 버그 n을 접하였을 때 n의 문장을 먼저 기존 훈련된 검색 모델로 임베딩하고 이후 코사인 유사도(Cosine similarity)에 기초하여 버그들이 있는 마스터 세트와의 거리를 계산한다. 여기서 상위 K개의 유사한 버그를 기반으로 하거나 유사성 값의 임계값(Threshold)을 적용하여 중복 버그를 검색할 수 있다.

그림 1. 모델



4 실험 및 평가

이 절에서는 3절에서 기술된 검색 모델의 실험에 대한 평가를 설명한다.

4.1 연구질문

RQ1 최신 기법에 비해 우리의 기법이 더 좋은가?
H0: 우리의 기법은 최신기법보다 성능이 같거나 낮다
H0: 우리의 기법은 최신기법보다 성능이 높다

RQ2 Siamese 네트워크가 성능에 주는 영향이 얼마인가?
H0: Siamese 네트워크 여부는 성능에 영향이 없다.
HA: Siamese 네트워크 여부는 성능에 영향이 있다.

Open Office, Ecilpse 프로젝트에서 수집한 두 개의 대형 버그 데이터 세트를 사용했는데 이 두 프로젝트의 데이터 세트

는 MSR2014의 데이터 세트[9]를 사용하였다. Eclipse는 자바를 위한 오픈소스인 통합 개발 환경을 제공하는 소프트웨어이며, Open Office는 마이크로소프트 오피스(Microsoft Office)와 같은 오픈소스 생산성 소프트웨어이다. 버그 보고서에는 표 1과 같이 다양한 정보들이 존재한다. 우리는 표 1의 버그의 고유 식별자인 'bug_id', 버그를 간략하게 설명하는 'short_desc', 버그 중복 목록인 'dup_id'를 사용하였다. 우리는 짧은 문장으로 서술된 부분만을 가지고서 모델을 구성할 것이다. 데이터 세트는 각각 dup_id를 기반으로 두 개 버그를 쌍으로 묶어서 'bug_id1', 'bug_id2' 로 구분하였고 여기에 중복인지 비 중복인지에 대한 구분으로 구성하였다. 이 구성된 데이터 세트로 훈련을 진행하였다. 다음 표 2 는 훈련 데이터 세트와 테스트 데이터 세트의 수를 보여준다. 훈련에 사용되는 데이터의 자연어는 아무런 전처리를 진행하지 않았다.

표 1. 버그 보고서 예시

"bug_id"	"104255"
"product"	"Mylyn"
"description"	"These are groupings of task contexts, e.g. everything under a category, and can be useful for views like content assist and open type."
"bug_severity"	"enhancement"
"dup_id"	"104362"
"short_desc"	"support 'context categories'"
"priority"	"P3"
"version"	"0.3"
"component"	"Core"
"delta_ts"	"2005-08-29 13:08:09 -0400"
"bug_status"	"RESOLVED"
"creation_ts"	"2005-07-18 14:50:00 -0400"
"resolution"	"DUPLICATE"

4.2 성능 척도

성능 척도는 $Recall$, $Accuracy$, $Precision$ 을 사용할 것이다. 각각의 척도는 confusion matrix의 TP, TN, FP, FN를 사용한다.

TP : 긍정적인 값을 올바르게 예측한 경우
TN : 부정적인 값을 올바르게 예측한 경우
FP : 긍정적인 값을 부정적으로 예측한 경우
FN : 부정적인 값을 긍정적으로 예측한 경우
아래는 성능 척도에 대한 설명이다.

$Recall$: 중복 버그들 중에서 모델이 중복 버그라고 예측한 비율

$$\frac{TP}{TP + FN}$$

$Precision$: 모델이 중복버그라 예측한 것 중 진짜 중복 버그인 비율

$$\frac{TP}{TP + FP}$$

$Accuracy$: 모든 예측 값들 중 올바르게 분류한 비율

$$\frac{TP + TN}{TP + FP + TN + FN}$$

두 모델의 Recall 비율은 테스트 세트에서 예측한 중복 버그 보고서 수를 원래 중복 버그 보고서 수로 나눈 비율이다. 이전 연구들에서 사용된 지표인 Recall 비율을 통해 이전 연구와의 성능 지표를 비교할 수 있었다.

표 2. 훈련과 테스트로 나눈 데이터세트

	훈련	테스트
Open Office	84574	21144
Eclipse	88458	22114

그림 2. K 변수에 따른 Recall 비율

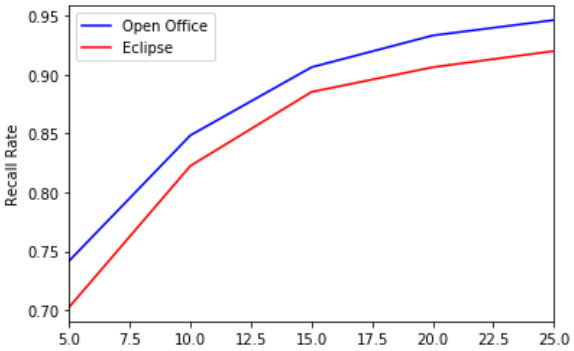


그림 3. 검색 모델 Eclipse데이터 결과

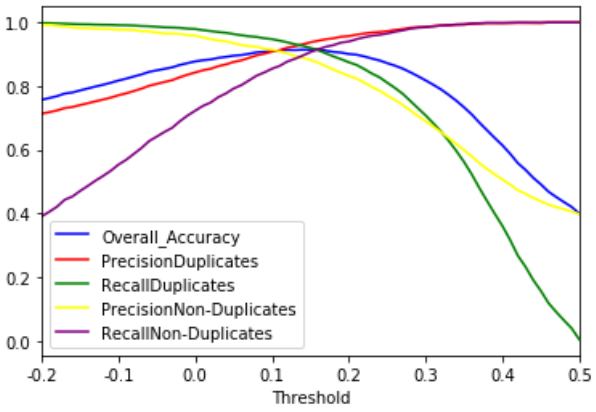


그림 4. 검색 모델 OpenOffice 데이터 결과

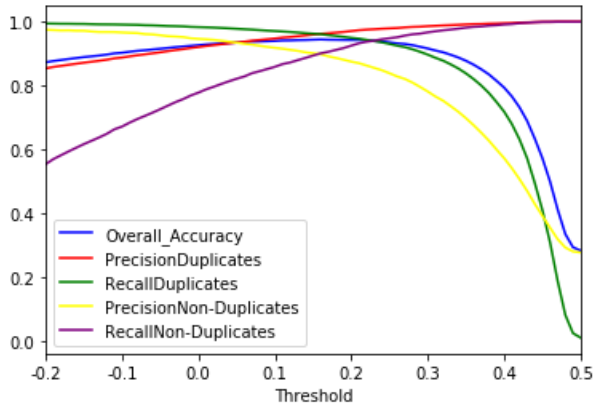


그림 5. Siamese 네트워크 미 적용 모델 Eclipse데이터

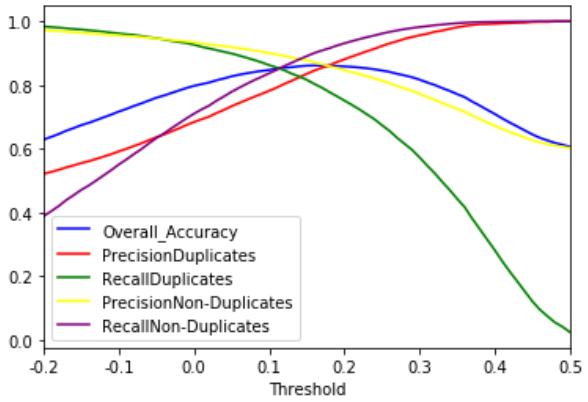


표 3. 모델의 ***Accuracy***

	검색 (Siamese적용)	검색 (Siamese 미적용)
Open Office	0.9460	0.9057
Eclipse	0.9148	0.8614

표 4. 기존 연구와 Recall 척도 비교

이름	데이터세트	Recall
$BM25F_{ezt}$[7]	Open Office	0.69
	Eclipse	0.67
DBTM (IR + TM)[3]	Open Office	0.81
	Eclipse	0.87
Siamese CNN + LSTM[4]	Open Office	0.79
	Eclipse	0.82
Our Model	Open Office	0.96
	Eclipse	0.92

4.3 결과

표 3은 검색 모델과 분류 모델의 ***Accuracy***를 보여준다 검색 모델은 93% 가까운 ***Accuracy***를 보여준다 그림 1은 검색 모델에 대해서 K 가 변화할 때 Recall 비율이 어떻게 달라지는지 보여준다. K가 10개만 넘어가도 모든 데이터가 80%가 넘는 Recall Rate를 보여주며 20개가 넘어가면 모든 데이터가 90%가 넘는 Recall Rate를 보여준다. 그림 3과 그림 4는 각 데이터의 ***Accuracy*** 결과를 보여준다 0.16의 임계값(Threshold)에서 최고의 ***Accuracy***를 보여준다. 또한 Siamese 네트워크 적용 여부에 따라서도 ***Accuracy*** 값이 다른 것을 확인할 수 있다. 이전 연구들은 [3,4,7]은 표 4와 같이 Eclipse 및 Open Office 데이터 세트의 ***Recall***을 보여준다. 이들의 연구는 동일한 데이터 세트 중 어느 부분을 훈련과 테스트에 사용했는지 명확 하지 않아 직접적인 비교는 어렵다. 하지만 앞에서 설명한 우리의 정확성 결과는 완전한 데이터 세트 기준으로 93%에 가깝고 Recall 또한 94%에 가깝다. 즉 RQ1의 질문에 대해 최신 기법들보다 더 좋은 성능 수치를 보여준다. 표 3과, 그림 5를 확인하면 Siamese 네트워크를 적용하지 않은 경우 성능이 약 5% 정도 감소하는 것을 확인할 수 있다. 즉 RQ2. Siamese 네트워크가 주는 영향이 5% 정도이다.

5 결론

많은 이전 연구[3,4]들이 검색 범위를 나누고 ***Accuracy***를 높이기 위해 구성 요소 ID와 같이 버그 보고서 데이터에 사용할 수 있다. 구조화된 정보에 기초한 휴리스틱(heuristics)을 사용하며 이 경우의 정보는 실제의 경우 많은 버그가 보고되지 못한다. 또한 이러한 버그는 프로젝트별로 달라질 수 있다. 우리의 모델은 구조화되어 있는 정보를 바탕으로 몇 가지 필수 정보를 가지고 데이터로부터 정보를 직접 학습하며 배운다. 더하여 BERT를 사용하였기 때문에 버그에 대한 정보 중 자연어라면 학습할 수 있다

결과적으로 BERT를 활용한 Siamese 네트워크 모델은 ***Accuracy*** 부분에서는 비교적 큰 성능의 향상은 없었으나 ***Recall*** 성능 수치 부분에서는 큰 성능의 향상을 확인할 수 있었다.

Acknowledgement

이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2019R1G1A1005047)

6 참고문헌

[1] Boehm, B., & Basili, V. R. (2005). Software defect reduction top 10 list. *Foundations of empirical software engineering: the legacy of Victor R. Basili*, *426*(37), 426-431.

[2] Cavalcanti, Y. C., de Almeida, E. S., da Cunha, C. E. A., Lucredio, D., & de Lemos Meira, S. R. (2010, March). An initial study on the bug report duplication problem. In *2010 14th European Conference on Software Maintenance and Reengineering* (pp. 264-267). IEEE.

[3] Nguyen, A. T., Nguyen, T. T., Nguyen, T. N., Lo, D., & Sun, C. (2012, September). Duplicate bug report detection with a combination of information retrieval and topic modeling. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering* (pp. 70-79). ACM.

[4] Deshmukh, J., Podder, S., Sengupta, S., & Dubash, N. (2017, September). Towards accurate duplicate bug retrieval using deep learning techniques. In *2017 IEEE International conference on software maintenance and evolution (ICSME)* (pp. 115-124). IEEE.

[5] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[6] Hiew, L. (2006). *Assisted detection of duplicate bug reports* (Doctoral dissertation, University of British Columbia).

[7] Sun, C., Lo, D., Khoo, S. C., & Jiang, J. (2011, November). Towards more accurate retrieval of duplicate bug reports. In *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering* (pp. 253-262). IEEE Computer Society.

[8] Alenezi, M., & Banitaan, S. (2013, December). Bug reports prioritization: Which features and classifier to use?. In *2013 12th International Conference on Machine Learning and Applications* (Vol. 2, pp. 112-116). IEEE.

[9] Lazar, A., Ritchey, S., & Sharif, B. (2014, May). Generating duplicate bug datasets. In *Proceedings of the 11th working conference on mining software repositories* (pp. 392-395). ACM.