한국정보과학회
KOREAN INSTITUTE OF INFORMATION SCIENTISTS AND ENGINEERS

제 23 권 제 1 호
Vol. 23 No. 1

SOFTWARE ENGINEERING SOCIETY

소프트웨어공학회지 KIPS

2021

제23회 한국 소프트웨어공학 학술대회

Proceedings of the 23rd Korea Conference on Software Engineering (KCSE 2021)

- **일시: 2021년 2월 1일(월) ~ 2월 2일(화)**
- **장소: 온라인 개최**

주최: 한국정보과학회, 한국정보처리학회
주관: 한국정보과학회 소프트웨어공학 소사이어티
한국정보처리학회 소프트웨어공학연구회
후원: ETRI 한국전자통신연구원 Electronics and Telecommunications Research Institute, SOLUTIONLINK
㈜비트컴퓨터, ㈜이에스지,
한국소프트웨어기술진흥협회(KOSTA)
STA 테스팅컨설팅㈜, T3Q㈜,
다한테크㈜, 슈어소프트테크㈜

초대의 글

소프트웨어공학 학술대회(KCSE 2021) 참가자 여러분을 환영합니다.

KCSE (Korea Conference on Software Engineering)는 기업, 연구소 및 학계에서 활동하고 계신 소프트웨어공학 분야 전문가들의 모임으로, 한국정보과학회 소프트웨어공학 소사이어티와 한국정보처리학회 소프트웨어공학연구회가 소프트웨어공학 기술의 발전 및 적용 확산을 위하여 1999 년부터 매년 개최하는 학술대회입니다.

이번 제 23 회 학술대회는 “비대면 시대를 지원하는 소프트웨어공학 기술”을 주제로, 기조 연설, 튜토리얼, 신진 연구자 발표, 우수 논문 발표 등의 초청 세션과 소프트웨어공학 분야의 각계에서 제출한 엄선된 40 편의 논문으로 구성하였으며, 2021 년 2 월 1 일부터 2 일간에 걸쳐 비대면으로 진행하게 되었습니다.

이번 학술대회가 비록 온라인으로 진행되어 아쉽지만, 소프트웨어공학의 학문적 발전과 소프트웨어 산업기술 발전의 장이 되고, 학술 교류 및 기술 협력을 위한 활발한 토론장이 될 수 있도록 가상 공간에서의 즐거운 만남으로 여러분을 초대하오니 많은 참여를 부탁드립니다.

제 23 회 KCSE 학술행사를 위해 수고해 주신 조직위원회와 학술위원회 위원들, 후원 기관 관계자 여러분, 그리고 기조 연설을 포함한 학술대회 모든 발표자분들께 깊이 감사드리며 건승을 기원합니다.

한국정보과학회 소프트웨어공학 소사이어티 회장 홍장의
한국정보처리학회 소프트웨어공학연구회 운영위원장 김정아

학술대회 준비 위원회

공동대회장: 홍장의 교수(충북대), 김정아 교수(가톨릭관동대)

조직위원장: 김순태 교수(전북대)
조직위원: 고인영 교수(KAIST), 김정아 교수(가톨릭관동대), 류덕산 교수(전북대),
백종문 교수(KAIST), 유준범 교수(건국대), 이선아 교수(경상대),
이정원 교수(아주대), 이찬근 교수(중앙대), 한종대 교수(상명대),
정효택 박사(ETRI), 민상윤 대표(솔루션링크), 전진옥 사장(비트컴퓨터),
이해서 대표(이에스지), 권원일 대표(STA테스팅 컨설팅), 박병훈 대표(T3Q),
박준성 회장(한국소프트웨어기술진흥협회)

학술위원장 유신 교수(KAIST)
학술위원: 김동선 교수(경북대), 김문주 교수(KAIST), 김미정 박사(홍콩과기대),
김윤호 교수(한양대), 김진대 교수(서울과학기술대), 김택수 박사(삼성전자),
남재창 교수(한동대), 박수진 교수(서강대), 배경민 교수(POSTECH),
서영석 교수(영남대), 윤회진 교수(협성대), 이선아 교수(경상대),
이우진 교수(경북대), 이주용 교수(UNIST), 이찬근 교수(중앙대),
지은경 교수(KAIST), 채흥석 교수(부산대), 최윤자 교수(경북대),
한종대 교수(상명대), 홍신 교수(한동대), 홍장의 교수(충북대학교)

문의사항 연락처

학술대회 홈페이지 : http://www.sigsoft.or.kr/KCSE2021/
조 직 : 김순태 교수 (stkim@jbnu.ac.kr 063-270-4788)
학 술 : 유신 교수 (shin.yoo@kaist.ac.kr 042-350-3567)

KCSE 2021 프로그램

<table>
<tr><th colspan="3">2월 1일 (월)</th></tr>
<tr><th>시 간</th><th colspan="2">행 사 내 용</th></tr>
<tr><td>9:30 – 10:00</td><td colspan="2">개회식
개회사 - 홍장의 회장 (한국정보과학회 소프트웨어공학 소사이어티), 김정아 운영위원장(한국정보처리학회 소프트웨어공학연구회)
사회: 김순태 조직위원장 (전북대)
장소: Session 1</td></tr>
<tr><td></td><td colspan="2">기조강연 I
사회: 유신 학술위원장 (KAIST)
장소: Session 1</td></tr>
<tr><td>10:00 – 10:50</td><td colspan="2">On Non-Functional Requirements in Software Engineering - Lawrence Chung (UT Dallas)</td></tr>
<tr><td></td><td>T1: 튜토리얼
좌장: 한종대 (상명대)
장소: Session 1</td><td>N1: 신진 연구자 초청세미나
좌장: 유신 (KAIST)
장소: Session 2</td></tr>
<tr><td>11:00 – 11:50
(50분)</td><td>산업 AI 를 위한 소프트웨어 공학
박찬진 (오브젠)</td><td>다중언어 안드로이드 앱의 정적 분석
이성호 (충남대학교)</td></tr>
<tr><td>11:50 – 13:10</td><td colspan="2">중식</td></tr>
<tr><td></td><td>T2: 튜토리얼
좌장: 한종대 (상명대)
장소: Session 1</td><td>N2: 신진 연구자 초청세미나
좌장: 지은경 (KAIST)
장소: Session 2</td></tr>
<tr><td>13:10 – 14:00
(50 분)</td><td>경험적 연구를 위한 GitHub 오픈소스 소프트웨어 데이터 수집
김진대 (서울과학기술대학교)</td><td>모델-검증기법을 적용한 자가-적응 소프트웨어 프레임워크
이의종 (충북대학교)</td></tr>
</table>

	초청 세미나, 논문 발표 A	
	A0: Software Testing 1 좌장: 유신 (KAIST) 장소: Session 1	**N3: 신진 연구자 초청세미나** 좌장: 지은경 (KAIST) 장소: Session 2
14:10-15:00 (50 분)	[초청 발표] Property-based Testing for LG Home Appliances using Accelerated Software-in-the-Loop Simulation (ICSE - SE in Practice, 2020) 박민규 (LG 전자), 장훈 (현대자동차), 변태준 (U. of Minnesota), 최윤자 (경북대학교) [일반 논문] 머신러닝을 사용한 코드와 요구사항 기반 테스트케이스 우선순위 지정방법 범준석, 백종문 (KAIST) [일반 논문] 유스케이스를 이용한 키워드 기반 테스팅 방안 최승훈, 최은만 (동국대학교) [단편 논문] 국지적 경로 공간 탐색 문제를 완화하기 위한 다방향 Concolic 탐색 전략 최한솔, 홍신 (한동대학교)	Better, Cheaper, and Faster Than Humans: Automated Test Generation for Improving Software Quality 김윤호 (한양대학교)
	논문 발표 A, B	
	A1: Repair & Synthesis 좌장: 배경민 (POSTECH) 장소: Session 1	**B1: Safety and Security** 좌장: 김윤호 (한양대) 장소: Session 2
15:10 – 16:00 (50 분)	[초청 발표] SAVER: Scalable, Precise, and Safe Memory-Error Repair (ICSE 2020) 홍성준, 이준희, 이정수, 오학주 (고려대학교) [초청 발표] The effectiveness of context-based change application on automatic program repair (Journal of Empirical Software engineering, 2020) 김진대 (서울과학기술대학교), 김정호, 이은석 (성균관대학교), 김성훈 (HKUST) [일반 논문] 코드 블록과 장단기 메모리를 활용한 프로그램 버그 정정 기법 호혜민, 양근석, 이병정 (서울시립대) [단편 논문] 임베디드 소프트웨어 도메인에서의 프로그램 합성 기법의 성능 확인을 위한 사례연구 김요엘, 최윤자 (경북대학교)	[일반 논문] 근로계약서를 위한 SPL 기반 실행 가능한 리카르디안 컨트랙트 생성 프레임워크 홍준기, 김순태, 류덕산 (전북대학교) [단편 논문] 온톨로지 기반 위험 구성 요소 분석을 통한 보안 요구사항 추천 방법 : 심층 보안 방어 관점 정지욱, 이석원 (아주대학교) [단편 논문] 비상호적 차분 프라이버시 모델의 노이즈 파라미터 분배 기법 김성민, 이원석 (연세대학교) [단편 논문] CCTV 영상 무결성 지원을 위한 대용량 이미지 블록체인 저장 방안 김태영, 김순태, 강민구, 송성한, 홍준기 (전북대학교)

	A2: Mining Software Repository 1 좌장: 류덕산(전북대) 장소: Session 1	**B2: SE & Machine Learning 1** 좌장: 남재창 (한동대) 장소: Session 2
16:10 – 17:00 (50 분)	[일반 논문] 적합도와 메타-학습을 결합한 하이브리드 학습 기반 소프트웨어 신뢰도 예측 모델 선정 기법 이낙원 (KAIST), 류덕산 (전북대학교), 조일훈 (LIG), 송재건 (ADD), 백종문 (KAIST) **[우수 학부생 논문]** 비지도 학습 기반 결함 예측 방법들의 성능 비교 연구 권수진, 남재창 (한동대학교) [학부생 논문] 이상치탐지 모델 학습에서 신경망 하이퍼파라미터가 모델의 성능에 미치는 영향 분석 고영진, 한성원 (KAIST)	[일반 논문] ML 을 활용한 Adaptive Streaming 재생 동기화 송기혁, 백종문 (KAIST) [일반 논문] 데이터 변형이 딥러닝 모델의 안전에 미치는 영향 측정 연구 김한동 (상명대학교) **[최우수 일반 논문]** DNN 을 이용한 효율적 컴파일 에러 위치식별 방법 배민지, 백종문 (KAIST) [일반 논문] 테스트 커버리지 기반 장단기 메모리 모델 구조 결정 김민하, 이민수, 이찬근 (중앙대학교)
	A3: Software Testing 2 좌장: 홍신 (한동대) 장소: Session 1	**B3: Maintenance** 좌장: 김정아 (카톨릭관동대) 장소: Session 2
17:10 – 18:00 (50 분)	[일반 논문] 다중 사용자 모바일 엣지 클라우드 환경에서 강화학습을 사용한 효율적 서비스 마이그레이션 방법 신준규, 고인영 (KAIST) [학부생 논문] TF-IDF 기반 유사도 측정 기법을 이용한 KAI 산업체 요구사항 매핑 테스트케이스 추천 시스템 주가희, 이효원, 박지성, 구나영 (경상대학교), 장혁 (KAI), 이선아 (경상대학교) [후원 업체발표] AI Testing based on ISO 29119-11 최영재 팀장/수석 (STA 테스팅컨설팅 주식회사)	**[최우수 일반 논문]** Access log data 기반의 IT 인프라 이상징후 감지 모델 김정원, 최호진 (KAIST) [단편 논문] eBPF 기반 System Monitoring Application 조사 연구 송소민 (경북대학교)

2월 2일 (화)		
시 간	**행 사 내 용**	
9:30 – 9:50 (20 분)	학술대회 공지 사회: 김순태 조직위원장(전북대) 장소: Session 1, 2	
	T3: 튜토리얼 좌장: 김윤호 (한양대) 장소: Session 1	**N4: 신진 연구자 초청 세미나** 좌장: 김문주 (KAIST) 장소: Session 2
10:00 – 10:50 (50 분)	Unit Test Fuzzing for C/C++ 홍신 (한동대학교)	Automated Mobile Security Solutions for End-users 이윤규 (서울여자대학교)
10:50 – 11:00	**휴식**	
	논문 발표 C, D	
	C1: CPS and IoT 1 좌장: 고인영 (KAIST) 장소: Session 1	**D1: Software Testing 3** 좌장: 이주용 (UNIST) 장소: Session 2
11:00 – 11:50 (50분)	**[우수 일반 논문]** Safety Assurance in Collaborative Cyber-Physical System through Fault Criticality Analysis Manzoor Hussain, 홍장의, Nazakat Ali (충북대학교) [일반 논문] 스위치 모드 파워 컨버터에 적용된 다양한 지능형 제어기에 대한 설계 및 성능 비교 장진행, 최호진 (KAIST) [일반 논문] 군집주행을 위한 ISO/PAS 21448 표준의 확장 김영재, 홍장의 (충북대학교)	[초청 발표] MAESTRO: Automated test generation framework for high test coverage and reduced human effort in automotive industry (Journal of Information and Software Technology, 2020) 김윤호 (한양대학교), 이동주, 백준기 (현대모비스), 김문주 (KAIST) [일반 논문] A Systematic Translation from PAT-based Counterexamples to Viable Test Sequences Zelalem Mihret Belay, Lingjun Liu, 지은경, 배두환 (KAIST) **[우수 단편 논문]** 입력 키워드 추출을 통한 뮤테이션 기반 Fuzzing 의 성능 향상 조정인, 홍신 (한동대학교)
11:50 – 13:10	**중식**	

	논문 발표 C, D	
	C2: Mining Software Repositories 2 좌장: 남재창 (한동대) 장소: Session 1	**D2: Software Architecture** 좌장: 김동선 (경북대) 장소: Session 2
13:10 – 14:00 (50분)	[초청 발표] Predicting just-in-time software defects to reduce post-release quality costs in the maritime industry (Journal of Software Practice and Experience, 2020) 강종구 (KAIST), 류덕산 (전북대학교), 백종문 (KAIST) **[우수 일반 논문]** 마코프 모델을 이용한 PM 추론 및 행위 분석 김소연, La Tung, 권기현 (경기대학교) [일반 논문] 소프트웨어 결함 분석을 위한 딥 메트릭 러닝 강민구, 최지원, 류덕산, 김순태 (전북대학교)	[학부생 논문] 협동 로봇 프로그래밍의 효율성 증대를 위한 마이크로서비스기반 플랫폼 설계 한성일, 고인영 (KAIST) [학부생 논문] 협동 로봇 고장 진단을 위한 요구사항 분석 및 블랙박스 설계 김양곤, 유동연, 박예슬, 이정원 (아주대학교) [단편 논문] Design of Serverless Clusters with stateless computing environment 정대욱, 백종문, 지은경 (KAIST)
	C3: Maintenance & Repair 좌장: 이찬근 (중앙대) 장소: Session 1	**D3: Maintenance & Architecture** 좌장: 김진대 (서울과기대) 장소: Session 2
14:10 – 15:00 (50분)	[일반 논문] BERT 기반 자연어 처리 딥러닝 모델을 활용한 효과적인 버그 배정 최예람, 백종문 (KAIST) [일반 논문] 협동 로봇의 예지보전 요구사항 기반 건전성 평가 기법 김진세, 유동연, 박예슬, 이정원 (아주대학교) **[우수 단편 논문]** 패치 우선순위화를 위한 소스 코드 표현방식에 따른 유사도 계산 기법별 영향도 분석 허진석, 정호현, 이은석 (성균관대학교)	**[최우수 일반 논문]** 이슈 관리를 위한 다중 레이블 분류 봇 박도제, 양혜진, 최광현, 이선아 (경상대학교), 강성원 (KAIST) [일반 논문] 효율적인 웹 정보 수집을 위한 강화학습 기반 스케줄러 설계 정대욱, 백종문 (KAIST) [후원 업체 발표] 대학 SW 프로젝트 산출물 통합 관리 박형권 이사 (주식회사 이에스지)

	C4: SE & Machine Learning 2 좌장: 백종문 (KAIST) 장소: Session 1	D4: Mining Software Repositories 3 좌장: 유신 (KAIST) 장소: Session 2
15:10 – 16:00 (50분)	[학부생 논문] 단어 임베딩 유사도 측정을 통한 함수 구성요소에 사용된 단어 간 적합성 평가 남성국, 서영석 (영남대학교) [단편 논문] Variational AutoEncoder 를 활용한 여행자 후기 기반 관광지 추천 알고리즘 조원희, 김창영, 김은수, 김윤민, 양형정 (전남대학교) [단편 논문] 데이터 증강 기법을 활용한 심층 신경망 강건성 평가 방법 최영원, 이영우, 채흥석(부산대학교)	**[우수 일반 논문]** 교차 버전 결함 예측을 위한 베이지안 최적화 프레임워크 최정환 (연세대학교), 류덕산 (전북대학교) [학부생 논문] 결함 데이터 수집 통합 프레임워크 한주희, 윤수빈, 양수진, 남재창 (한동대학교)
16:00 – 16:10	**휴식**	
	기조강연 II 사회: 유신 학술위원장(KAIST) 장소: Session 1	
16:10 – 17:00	Adaptive Software Engineering - 배두환 (KAIST)	
17:00 – 17:50	**상장수여식 및 경품** 사회: 김순태 조직위원장(전북대) 장소: Session 1	
17:50 – 18:00	**폐회식** 폐회사 - 홍장의 회장 (한국정보과학회 소프트웨어공학 소사이어티) 사회: 김순태 조직위원장(전북대) 장소: Session 1	

KCSE 2021 튜토리얼

튜토리얼 T1: 산업 AI 를 위한 소프트웨어공학

- 일시: 2월 1일(월) 11:00~11:50
- 장소: Session 1
- 제목: 산업 AI를 위한 소프트웨어 공학
- 연사: 박찬진 (Obzen)
- 튜토리얼 초록:

  많은 기업들이 AI 기술 확보 노력을 많이 하고 있으나, 실세계 문제에 적용하여 성과를 얻고 있는 기업은 많지 않은 것 같다. 이미지 분류, 얼굴 인식, 언어 변역 등의 많은 AI 기술 사례는 AI 가 할 수 있는 많은 일이 있음을 보여주고 있지만, 이들 기술이 실제 공정이나 환경에 적용되어 성과를 얻기 위해서는 많은 엔지니어링 차원의 문제를 극복해야 한다. AI 에게 일을 맡기기 전에 신뢰가 중요하다. AI 시스템은 데이터로부터 학습된 새로운 타입의 소프트웨어 시스템이므로 통합, 테스팅, 배포, 모니터링 및 유지보수 (AI 모델의 재학습과 업그레이드)와 같은 소프트웨어 엔지니어링 활동이 필수적이다. 하지만, AI 시스템 엔지니어링에 대한 체계적인 방법론은 아직 정립되어 있지 않은 것으로 보인다. 이 튜토리얼에서는 반도체 기업에서 AI 과제를 추진하면서 직면했던 엔지니어링 문제들과 이에 대한 극복 내용을 다룬다. 또한, AI 모델을 만들고 이를 현장에 적용하면 얻은 교훈을 공유하고자 한다. 본 튜토리얼은 크게 네 부분으로 구성되어 있으며, 먼저, 제조 디지털 혁신을 위해 소프트웨어 공학 기술이 적극적으로 활용되고 있음을 소개하고, 두번째로는 AI 개발과 전통적 SW 개발 간의 차이와 AI 개발 방법론의 필요성을 설명한다. 세번째로, 반도체 불량 시각 검사 자동화 과제에 AI 적용을 통해 배운 교훈을 소개하며 특히 현장에 AI 가 활용되기 위해서는 AI 모델 개발 뿐만 아니라 AI 의 운영 시스템화가 중요함을 강조한다. 마지막으로 불량 검사 AI 는 불량 분류 모델 하나 만으로 완성되지 않으며 다양한 모델의 조합이 필요함을 설명한다.

- 약력:
  - 2021 ~ 현재: 오브젠, 상무, AI Lab
  - 2017 ~ 2020: SK Hynix, 상무, Data Science Team, 데이터사이언스 담당
  - 2014 ~ 2016: 서울대 차세대 융합기술원 , 책임연구원, 공공데이터센터, 융합연구과제 텍스트분석, 전력수요예측, 빅데이터 인프라 및 분석 플랫폼
  - 2006 ~ 2014: LG 전자, 수석연구원, LG webOS TV 아키텍트로 설계 및 성능 개선 리더
  - 1994: ~ 1998: LG Software, 주임연구원, 소프트웨어 개발
  - 1994 서울대학교 계산통계학과 졸업
  - 2000 서울대학교 전산학과 석사 (소프트웨어 공학)
  - 2006 서울대학교 전기 컴퓨터 공학부 박사 (소프트웨어 공학)
  - 박사논문: 객체지향 프로그램을 위한 층위구조 아키텍처 복구 및 일치성 검사 방법. A Recovery and Conformance Checking Technique of Layered Architecture for Object-Oriented Programs
  - 한국소프트웨어공학회 이사연구분야: 소프트웨어 테스팅(AI 테스팅, 테스팅 프로세스 등), 소프트웨어 안전, 애자일 개발 방법론

튜토리얼 T2: 경험적 연구를 위한 GitHub 오픈소스 소프트웨어

- 일시: 2월 1일(월) 13:10~14:00
- 장소: Session 1
- 제목: 경험적 연구를 위한 GitHub 오픈소스 소프트웨어 데이터 수집
- 연사: 김진대 (서울과학기술대학교)
- 튜토리얼 초록:

  GitHub 은 가장 성공적인 오픈소스 소프트웨어 호스팅 서비스 중의 하나로, 막대한 양의 프로젝트를 포함하고 있는 데이터의 보고입니다. 따라서 경험적 연구를 위해 소프트웨어 개발과 관련된 데이터를 수집할 때 반드시 고려해 보아야 할 선택지라고 할 수 있습니다. 본 튜토리얼에서는 이런 GitHub 에서 어떻게 유용한 소프트웨어 관련 데이터를 수집할 수 있는지, 또 그 과정에서 주의해야 할 점은 무엇인지 연구과정에서 겪었던 경험을 공유하고자 합니다
- 약력:
  - 서울과학기술대학교 컴퓨터공학과 조교수
  - PhD in CSE, Hong Kong University of Science and Technology
  - 서울대학교 소프트웨어공학 석사
  - 서울대학교 물리학/컴퓨터공학 학사

튜토리얼 T3: Unit Test Fuzzing for C/C++

- 일시: 2월 2일(화) 10:00~10:50
- 장소: Session 1
- 제목: Unit Test Fuzzing for C/C++
- 연사: 홍신 (한동대학교)
- 튜토리얼 초록:

  최근 Greybox 퍼징 기법이 고도화되고 완성도 높은 오픈소스 퍼징 도구가 개발됨에 따라 퍼징을 통한 자동 테스팅이 새로운 품질 보장 관행으로 빠르게 자리매김하고 있다. 이번 튜토리얼에서는 C/C++ 프로그램 대상 퍼징 도구로 널리 사용되고 있는 libFuzzer 도구를 중심으로 Greybox 퍼징의 원리를 소개하며, libFuzzer를 활용하여 C/C++ 프로그램에 대해 유닛 테스팅 수준에서 퍼징을 적용하는 방법을 실습과 함께 소개한다.

- 약력:
  - 2016~현재. 한동대학교 전산전자공학부 조교수
  - 2011~2015. KAIST 전산학 박사
  - 2007~2010. KAIST 전산학 석사
  - 2003~2007. KAIST 전산학 학사

KCSE 2021 기조강연

기조강연 I: On Non-Functional Requirements in Software

- 일시: 2월 1일(월) 10:00~10:50
- 장소: Session 1
- 제목: On Non-Functional Requirements in Software Engineering Lawrence Chung
- 연사: Lawrence Chung (UT Dallas)
- 초록:

  Non-Functional Requirements (NFRs) are important not only in software engineering but also in just about any engineering discipline, but often times much more challenging to deal with than functional requirements (FRs). In this talk, I will start with a brief description of what NFRs are and then go through some key characteristics of NFRs, some key rules of thumb for dealing with NFRs, and how to represent and reason about them. In this talk, I will present NFRs in a more casual and conceptual, and less technical and formal, manner.
- 약력:

  Lawrence Chung has been working in Requirements Engineering, System/Software Architecture and Systems Engineering. He was the principal author of the research monograph "Non-Functional Requirements in Software Engineering", and has been involved in developing "RE-Tools" (a muiti-notational requirements tool) , "HOPE" (a smartphone app project for people with difficulties), "Silverlining" (a Google-award winning project on cloud computing and big data), etc. He has been a keynote speaker, invited lecturer, co-editor-in-chief for Int. Journal of Software Innovative (IJSI) and Int. Journal of Big Data Intelligence and Applications (IJBDIA), associate editor for Requirements Engineering Journal (REJ), editorial board member for Int. Journal of Networked and Distributed Computing (IJNDC), editor for ETRI Journal (ETRIJ), and program co-chair for various international events. Prof. Chung is currently on the faculty of Computer Science (and Software Engineering) at University of Texas at Dallas. He received his Ph.D. in Computer Science in 1993 from University of Toronto.

기조강연 II: Adaptive Software Engineering

- 일시: 2월 2일(화) 16:00~17:00
- 제목: Adaptive Software Engineering
- 연사: 배두환 (KAIST)
- 초록:

  4차 산업혁명 시대를 맞이해서 '소프트웨어가 세상을 먹어버린다'는 말이 정말 실감이 날 정도로 우리 주변에 많은 변화를 소프트웨어가 주도하고 있다. 최근 들어 AI에 대한 수요가 폭발적으로 증가한 것 역시 소프트웨어 시스템의 활용 영역과 범위를 빠른 속도로 넓히고 있다. 결과적으로 많은 소프트웨어 개발 및 관련 연구가 필요한 시점이다. 소프트웨어에 몸담고 있는 우리들에게 더 많은 기회가 올 것이라는 막연한 기대를 가지고 있다. 한 편, 1968년 NATO Conference에서 'Software Engineering'이라는 용어가 처음 소개된 이후로 지난 약 60여년간 소프트웨어 개발 패러다임도 많은 진화를 이루어 왔다. 소프트웨어공학 역시 연구 개발에서 많은 성장을 이루어 왔으나, 미래에는 더 빠른 속고도 변화할 것이며, 소프트웨어공학자로서 이러한 변화에 적응하지 못하면 도태할지도 모른다는 두려움 역시 적지 않다. 본 주제 강연에서는 미래의 소프트웨어 개발자로서 성공하기 위해 필요한 역량을 알아 보고, 또 소프트웨어공학 연구자로서 더 이상 추적 연구가 아닌 선도적인 연구를 하기 위한, 즉 fast mover가 되기 위한 연구 방향 중의 하나로 Adaptive Software Engineering을 정의하고 그 의미와 내용들을 알아본다.
- 약력:

  배 두환 교수는 1992년 University of Florida에서 전산학 박사 학위를 마치고, 1995년부터 카이스트 전산학부의 교수로 부임하였다. 현재 주요 연구 분야는 Modeling and Verification of System of Systems로 Model-Based Software Engineering 기법들을 대규모 SW 시스템에 적용하는 연구로, SW 스타랩으로 선정되어 관련 연구를 하고 있다. 지금까지 약 200여편의 국제 우수 저널 및 학술 대회에 논문을 발표 및 게재하였으며 약 150명의 석/박사 졸업생을 배출하였다. 2002년부터 2018년까지 KAIST SW Process 개선 센터장, 2005년부터 2011년까지 SW 전문가 과정 책임교수, 2008년부터 2009까지 초대 한국 소프트웨어공학 소사이어티 회장, 2008년부터 2012년까지 Asia-Pacific Software Engineering Conference의 Steering Committee Chair, 2012년부터 2016년까지, 카이스트 전산학부장을 역임하였으며, 현재는 KAIST 전산학부의 ICT 석좌교수로 SW 교육 센터장을 맡고 있으며 한국공학한림원 회원으로 활동하고 있다. 많은 공로상 및 장관상, 대통령 표창을 수상하였으며, 2019년 SW 산업 유공자 근정 포장을 수상하였다.

KCSE 2021 신진 연구자 초청세미나

신진 연구자 초청세미나 1: 다중언어 안드로이드 앱의 정적 분석

- 일시: 2월 1일(월) 11:00~11:50
- 장소: Session 2
- 제목: 다중언어 안드로이드 앱의 정적 분석
- 연사: 이성호 (충남대학교)
- 세미나 초록:

  SW 개발 환경이 다양해지면서, 여러 언어를 함께 사용하여 구현하는 다중언어 프로그램 개발이 보편화 되고있다. 다중언어 프로그램은 서로 다른 언어의 특성을 함께 사용할 수 있다는 장점을 갖는 동시에, 언어 사이의 상호 작용에서 예상치 못한 다양한 형태의 결함이 발생할 수 있고 이러한 결함이 보안 취약점으로 이어질 수 있다는 위험이 존재한다. 프로그램의 결함 및 보안 취약점을 사전에 탐지하는 정적분석 기법이 단일언어 프로그램을 대상으로 널리 사용되어 왔으나, 정적분석 기법은 프로그램의 구현 언어에 의존적이기 때문에 다중언어 프로그램을 대상으로는 적용이 제한된다. 본 세미나에서는 다중언어 프로그램, 특히 널리 활용되는 다중언어 안드로이드 앱에서 발생 가능한 결함 및 보안 취약점을 소개하고, 이를 탐지하는 정적분석 연구에 대해 소개한다.

- 약력:
  - 2020. 09. ~ present: 충남대학교 컴퓨터융합학부 조교수
  - 2020. 02. ~ 2020. 08.: 구글 Visiting Faculty Researcher
  - 2014. 03. ~ 2020. 02.: 한국과학기술원 박사과정 (지도교수: 류석영)
  - 2012. 02. ~ 2014. 02.: 한국과학기술원 석사과정 (지도교수: 한태숙)
  - 2006. 03. ~ 2012. 02.: 아주대학교 학사

신진 연구자 초청세미나 2: 모델-검증기법을 적용한 자가-적응 소프트웨어

- 일시: 2월 1일(월) 13:10~14:00
- 장소: Session 2
- 제목: 모델-검증기법을 적용한 자가-적응 소프트웨어 프레임워크
- 연사: 이의종 (충북대학교)
- 세미나 초록:

  자가-적응 소프트웨어(Self-adaptive software)는 요구사항에 따라 소프트웨어의 구조나 행동을 실행시간에 변화시키며 상황에 맞게 적응(adapt)하는 소프트웨어를 말한다. 본 연구는 자가-적응 소프트웨어의 프레임워크에 대한 연구로 크게 두 가지 특징을 갖고 있다. 1) 모델-검증 기반의 소프트웨어 모델링 및 검증 기법: 자가-적응 소프트웨어의 연구 분야 중, 실행시간 검증기법은 동적환경에서 소프트웨어가 적응을 수행하도록 하는 중요한 연구 분야 중 하나이다. 더불어 자가-적응 소프트웨어에 기존의 다양한 검증 방법을 적용시키는 연구들이 다방면으로 진행되고 있다. 모델-검증(Model checking)방법은 효과적인 모델검증 방법 중 하나이지만 상태 폭발(state explosion)과 같은 고질적인 문제점으로 인해 실행시간 검증에 활용되기는 어려운 측면이 있다. 본 연구의 프레임워크는 모델-검증 방법을 활용해 자가-적응 소프트웨어를 모델링하고 이를 실행시간에 검증하기 위한 방법을 기반으로 한다. 2) 게임이론 기반의 적응-전략 추출 기법: 자가-적응 소프트웨어는 환경변화에 따른 올바른 적응을 위해 가장 최적의 전략을 선택해야 한다. 이를 위해 게임이론에 기반한 적응전략(adaptive strategy) 추출기법이 적용되었다. 제안된 프레임워크는 사물인터넷(Internet of Things) 환경에 적용되어 실효성을 보였다

◆ 약력:
- 2020.9 ~ 현재 - 조교수, 소프트웨어학과, 충북대학교
- 2018.9 ~ 2020.8 - 박사후연구원, 정보보호학과, 세종대학교
- 2017.9 ~ 2020.8 - 외래교수, 컴퓨터공학과, 서울사이버대학교
- 2012.3 ~ 2018.8 – 박사, 컴퓨터학과, 고려대학교
- 2006.3 ~ 2012.2 – 학사, 컴퓨터정보학과, 고려대학교

신진 연구자 초청세미나 3: Better, Cheaper, and Faster Than Humans: Automated Test Generation for Improving Software Quality

◆ 일시: 2월 1일(월) 14:10~15:00
◆ 장소: Session 2
◆ 제목: Better, Cheaper, and Faster Than Humans: Automated Test Generation for Improving Software Quality
◆ 연사: 김윤호 (한양대학교)
◆ 초록:

소프트웨어 테스팅은 소프트웨어 품질을 향상시키는데 아주 중요한 역할을 하고 있다. 하지만 소프트웨어 크기와 복잡도가 기하급수적으로 증가하고 있기 때문에, 개발자가 버그를 찾기 위한 효과적인 소프트웨어 테스트를 작성하는데 어려움을 겪고 있다. 반면, 컴퓨팅 파워는 하드웨어 성능과 클라우드 컴퓨팅 기술의 발전에 힘입어 날이 갈수록 비용이 저렴해지고 있다. 따라서, 컴퓨팅 파워를 활용하여 소프트웨어를 과학적이고 체계적인 방법으로 자동으로 테스트하기 위한 연구가 필요하다.

본 발표에서는 최신 소프트웨어 테스트 자동 생성 기술인 CONBRIO와 FOCAL을 소개한다. CONBRIO는 크고 복잡한 소프트웨어의 테스트를 자동 생성하기 위해 소프트웨어를 구성하는 각 함수의 유닛 테스트를 자동으로 생성한다. 하지만 각 유닛은 전체 프로그램의 문맥 정보를 잃기 때문에 문맥 정보 부족으로 인한 거짓 경보가 발생할 수 있다. 이런 거짓 경보를 제거하기 위해 FOCAL은 유닛 테스트 실행 정보를 활용하여 시스템 레벨에서 버그를 재현할 수 있는 시스템 테스트를 자동으로 생성한다. CONBRIO와 FOCAL은 기존 벤치마크에서 효과적으로 버그를 찾았을 뿐 아니라 libpcre, libxml2와 같은 오픈 소스 프로젝트의 새로운 버그도 효과적으로 찾을 수 있었다.

◆ 약력:
- 2020.9 ~ 현재 - 조교수, 소프트웨어학과, 충북대학교
- 2018.9 ~ 2020.8 - 박사후연구원, 정보보호학과, 세종대학교
- 2017.9 ~ 2020.8 - 외래교수, 컴퓨터공학과, 서울사이버대학교
- 2012.3 ~ 2018.8 – 박사, 컴퓨터학과, 고려대학교
- 2006.3 ~ 2012.2 – 학사, 컴퓨터정보학과, 고려대학교

신진 연구자 초청세미나 4: Automated Mobile Security Solutions for End-users

- 일시: 2월 2일(화) 10:00~10:50
- 장소: Session 2
- 제목: Automated Mobile Security Solution for End-users
- 연사: 이윤규 (서울여자대학교)
- 초록:

With the recent development of mobile devices, end-users' dependency on mobile devices has increased. At the same time, security threats are becoming more prevalent, and the importance of end-user security is increasing. In this talk. I will introduce my representative research related to 'Automated Mobile Security Solutions for End-users'. SEALANT (ICSE 2017) is a technique that combines static analysis of app code, which infers vulnerable communication channels, with runtime monitoring of inter-app communication through those channels, which helps to prevent attacks. SCUTUM is a technique that automatically detects event-based MOM systems' vulnerabilities that expose the system to event attacks. Fusion Learning for Fingerprint Anti-Spoofing is a robust fingerprint anti-spoofing technique that supports high detection accuracy, generalization performance, and lightweight models.

- 약력:
  - Assistant Professor, Department of Information Security, Seoul Women's University
  - Staff Researcher, AI&SW Center, Samsung Advanced Institute of Technology
  - Ph.D., Computer Science, University of Southern California
  - B.S. & M.E., Computer Science, Korea University

KCSE 2021 Invited Talk

우수 국제 저널 출판 논문 초청

- MAESTRO: Automated test generation framework for high test coverage and reduced human effort in automotive industry (Journal of Information and Software Technology, 2020) – 김윤호 (한양대학교), 이동주 (현대모비스), 백준기 (현대모비스), 김문주 (KAIST)
- Predicting just-in-time software defects to reduce post-release quality costs in the maritime industry (Journal of Software Practice and Experience, 2020) – 강종구 (KAIST), 류덕산 (전북대학교), 백종문 (KAIST)
- The effectiveness of context-based change application on automatic program repair (Journal of Empirical Software engineering, 2020) – 김진대 (서울과학기술대학교), 김정호 (성균관대학교), 이은석 (성균관대학교), 김성훈 (홍콩과학기술대학교)

우수 국제 학술 대회 초청 발표

- Property-based Testing for LG Home Appliances using Accelerated Software-in-the-Loop Simulation (International Conference on Software Engineering, SE in Practice, 2020) – 박민규 (LG전자), 장훈 (현대자동차), 변태준 (University of Minnesota), 최윤자 (경북대학교)
- SAVER: Scalable, Precise, and Safe Memory-Error Repair (International Conference on Software Engineering, 2020) – 홍성준 (고려대학교), 이준희 (고려대학교), 이정수 (고려대학교), 오학주 (고려대학교)

MEMO

KCSE 2021 논문 목차

일반 논문

단편 논문

학부생 논문

Access log data 기반의 IT 인프라 이상징후 감지 모델

김정원*, 최호진
한국과학기술원 전산학부 소프트웨어대학원
e-mail : jaywon.kim@kaist.ac.kr, hojinc@kaist.ac.kr

IT infrastructure abnormal detection model based on the access log data of web server

Jungwon Kim* and Ho-Jin Choi
School of Computing, Korea Advanced Institute of Science and Technology

요 약

IT서비스를 운영하다 보면 시스템 장애를 완전히 피할 수 어렵고, 이를 예견하는 것은 더더욱 어려운 일이다. 더구나 시스템 장애는 그 시스템의 신뢰도 하락과 함께 이용자들이 서비스를 제공받지 못함에 따라 발생하는 회사나 조직 간에 손해배상을 청구하기도 할 정도로 매우 민감한 사항이다. 수년간 현장에서 겪었던 경험에 따르면 장애가 발생하기 전, 소위 말하는 '전조현상'이 있는데, 본 연구에서는 이러한 전조현상의 특성을 파악하면 시스템 이상현상을 감지할 수 있다는 가정에서부터 시작한다. 이미 이상징후 탐지 모델에 대해서는 과거부터 꾸준히 연구되어왔고 지금도 진행중이지만, 본 실험에서는 이상징후 탐지 모델을 구축하는데 사용될 데이터로써 웹서버의 access log를 선택하였고, 기존에 소개된 스펙트럼 잔차 모델을 실시간 IT 서비스 관제에 적용했을 때의 문제점 파악하여 개선된 모델을 제시하는 데에 그 목적이 있다.

1. 서 론

IT 산업의 장애상황에서 '이 상황을 어떻게 빨리 해결하는가'의 열쇠는 유능한 엔지니어의 역할도 있겠지만, 무엇보다 가장 중요한 것은 '얼마나 빨리 장애를 인지했는가'라고 할 수 있다.

일반적인 웹 서비스의 경우, 웹 서버를 통하여 back-end 비즈니스 서버로 트래픽이 흘러가는 모습인데, 이 때 웹서버에 들어오는 모든 트래픽 정보들이 access log에 기록된다. 이 access log는 특정 웹서비스에 요청되는 모든 트래픽을 기록하고 있기 때문에 이상징후를 판단 하기위한 훌륭한 학습 데이터로써 가치가 있기도 하다. 따라서 이 정보들을 수집한 뒤, 각 서비스의 패턴과 특성을 시계열 관점으로 분석이 가능하므로, 장애상황이 발생하기 전에 이상징후를 감지할 수 있는 모델을 구현할 수 있다

물론 이상징후 탐지에 대한 아이디어는 그 접근방법과 적용할 수 있는 분야가 다양하고, 많은 연구결과들이 있지만, 본 연구에서는 기존 방법론을 이용해서 웹 서비스의 access log 도메인에 적용했을 때 드러날 수 있는 한계점을 도출해내고, 이를 해결하기 위한 아이디어를 제시한다.

2. 관련 연구

일반적이지 않은 것을 찾는 것을 이상감지라고 하는데, 이에 대한 연구는 오래전부터 존재해왔지만, 최근 기계학습과 빅데이터 분야의 성장과 함께 다시 관심과 재조명을 받고 있다.

몇몇 연구들을 보면 이상징후 판별 모델에 시계열 도메인을 적용하여 ARIMA, LSTM 이 많이 언급[1][2]되고 있다. 그러나 본 연구에서는 시계열 데이터를 주파수 영역으로 변환한 후 특이 값을 판단하는 Saliency map 을 이용해서 이상데이터를 추출하는 알고리즘[3][4]을 활용하였는데, 이 알고리즘 역시 연산량에 대한 이슈가 존재하므로 실시간 도메인에 적용하기 위해서는 해결책이 필요하다. 본 연구에서는 이러한 이슈를 해결하기 위해 기존에 연구되었던 알고리즘을 활용[3]하여 최소한의 데이터 특성을 이용해서 해결할 수 있는 방법을 모색할 것이다.

3. 이상징후 판단 알고리즘

이미지로부터 중요 객체를 추출할 때 Saliency Map 을 사용[4]하는데, 시계열로 구성된 access log 데이터에 이 알고리즘을 적용하면 다른 영역에 비해 변화가

급격한 부분을 검출할 수 있다는 점에서 이상징후를 판별이 가능하다. Spectral Residuals 를 이용한 이상징후 검출방법은 관련 연구를 통해 많이 확인할 수 있고 그 검출 성능에 대해서도 검증되었지만 매분마다 이상징후를 판별해야 하는 본 연구의 목표에 부합하려면 Spectral Residuals 단독 모델이 아닌, 이를 보완할 수 있는 추가 학습모델을 고려해야 한다. 본 연구에서는 Spectral Residuals 알고리즘의 한정적인 데이터 수집상태에서 발생할 수 있는 탐지오류를 제어할 수 있도록, Polynomial Regression 알고리즘을 이용하여 상호보완하도록 하였다. 수많은 과거데이터를 기반으로, Polynomial Regression 을 적용하면, R^2(다중결정계수)값이 가장 큰 기준선을 결정하기 위해, 수많은 데이터를 이용할 수 있고, 시스템 및 일자별 패턴 파악 가능하기 때문에 Spectral Residuals 알고리즘이 해결하기 어려운 이슈를 보완할 수 있다.

이와 반대로, Polynomial Regression 만 사용했을 때에도 탐지오류가 발생할 수 있는 경우가 있다. 예를 들어, 정상범위와 비정상범위 경계에 있는 값이 연속적으로 존재한다고 가정했을 때, 미세하게 경계면을 오르내리는 패턴이 발생한다면 빈번한 alert 이 발생할 가능성이 있다.

따라서 이 두 알고리즘을 앙상블한 모델에서는 이상징후 상황을 모두 충족하는 경우만 인지하도록 할 수 있다.

4. 실험결과

본 연구의 목표인 실시간 모니터링을 위해, 비교적 데이터 수집이 적은 상황에서 이상징후를 탐지하게 되면, 비슷한 응답시간의 데이터들이 주를 이루고 있기 때문에, 정상적인 미세한 변화에도 임계치(이상징후 기준)를 초과하는 결과를 확인할 수 있었다.

반대로 관찰된 데이터가 충분할 경우, 이상징후가 시작되는 시점을 기준으로 충분히 학습된 이전과 다른 패턴이 검출됨에 따라 해당 데이터를 이상 값으로 판단하고 높은 outlier score 를 반환한다는 것을 확인하였다.

따라서 Spectral Residuals 의 결과는 실시간으로 입력되는 데이터를 계속 적재하면서 급변하는 이상징후 판단에는 효과적일 수 있으나, 데이터가 충분하지 않은 짧은 기간에서는 탐지오류를 발생시킬 수 있기 때문에 이를 보완하기 위해서는 Polynomial Regression 알고리즘을 결합한 의사결정이 필요하다.

아래 [그림 1]은 Polynomial Regression 모델이 판단한 정상범위를 참조하도록 하고 Spectral Residuals 결과와 앙상블하여 이상징후를 판단하는 것을 나타낸다. 실제 장애 데이터를 이용해서 이 모델에 적용해보면 서비스장애가 발생하기 18 분전에 이상징후를 감지하였음을 확인할 수 있었다. 즉, 두 알고리즘이 모두 이상징후라고 인정한 경우에만 이상징후로 판단하는 것이다. 따라서 Spectral Residuals 과 Polynomial Regression 을 상호보완한 앙상블 모델을 적용하면 실제로 시스템 장애가 발생하기전에 사전 대응을 할 수 있는 시간 확보가 가능하다는 것을 보여준다.

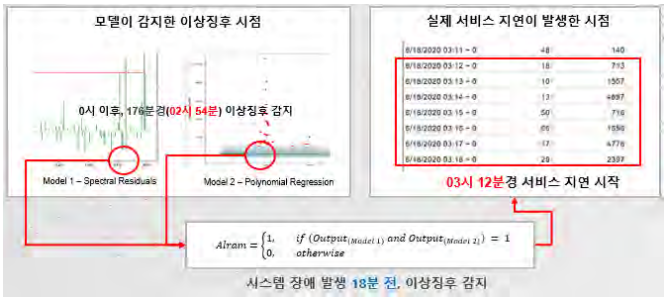


그림 1 : 상호보완 모델에서의 이상징후 감지

5. 결론

그 동안 이상징후 탐지분야는 LSTM, ARIMA, S-H-ESD 등 다양한 방법으로 연구되어왔고, 본 연구에서도 참고하였던 Spectral Residual 알고리즘 역시 이미지 객체 탐지에서 사용되던 것을, 이상탐지 모델의 한 방법으로써 많이 사용되고 있다.

따라서 본 연구에서는 단순 Spectral Residuals 알고리즘이 아닌, Polynomial Regression 알고리즘과의 앙상블 기법을 통하여 이 한계를 극복하는 실험을 진행하였고, 그 결과 실제 이상징후를 예측하여 장애상황에 대비할 수 있음을 확인하였다.

물론 이 방법이 최선의 방법이라고 단정할 수는 없으므로, 더 다양한 장애 데이터로 테스트하고 최적화하여, 범용적인 상황에서도 최적의 성능에 수렴할 수 있도록 시행착오가 필요할 것이다.

참고문헌

[1] GP Zhang, Time series forecasting using a hybrid ARIMA and neural network model - Neurocomputing 50, 159-175, 2003

[2] ZC Lipton, DC Kale, C Elkan, R Wetzel, Learning to diagnose with LSTM recurrent neural networks, arXiv preprint arXiv:1511.03677, 2015

[3] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang, Time-Series Anomaly Detection Service at Microsoft. In KDD. 3009-3017 July 2019

[4] X. Hou and L. Zhang. Saliency detection: A spectral residual approach. IEEE Conference on Computer Vision and Pattern Recognition, 2007

BERT 기반 자연어 처리 딥러닝 모델을 활용한 효과적인 버그 배정

최예람 O 백종문
한국과학기술원
yeram.choi@kaist.ac.kr, jbaik@kaist.ac.kr

An Effective Bug Triage using BERT based NLP Deep Learning Model

Yeram Choi O Jongmoon Baik
KAIST

요 약

본 연구에서는 산업계의 실제 버그 할당 프로세스 분석과 문제점 자동 할당에 관한 연구들을 분석함으로서 얻은 통찰을 토대로 최신 딥러닝 기술의 적용과 실제 산업계의 데이터를 활용을 통해 문제점 할당에 투입되는 자원을 절감하고자 한다. 이를 위해 산업계 데이터를 분석하고, 데이터 형태에 적합한 모델을 선정하고, 모델들을 기반으로 본 문제를 해결하기 위한 최적의 모델을 개발한 뒤, 기존 문제점 분류 프로세스 상 적절한 위치에 가장 효과적인 형태로 모델을 적용하고, 실제로 개선이 되는지 평가해보고자 한다. 본 연구를 통해 실제 버그의 재할당 횟수를 줄이거나 재할당 과정을 자동화 할 수 있다면 버그 배정(Bug Triage)에 투입되는 비용을 절감할 수 있을 것이다.

1. 서 론

소프트웨어 프로젝트에서 버그를 발견하고 개선하는 과정은 필연적이면서 비용이 많이 투입되는 부분이다. 특히 산업계에서는 경영 환경상 다양한 제약으로 인해 충분하지 못한 개발 기간이나 리소스 부족으로 인해 품질 문제가 발생하는 경우가 많고, 이로 인해 개발에 투입되는 리소스보다 품질을 확보하기 위해 투입되는 리소스가 많아지기 경우가 비일비재하다. 충분한 개발 일정이나 더 많은 리소스를 확보하는 것은 산업계에서 현실적으로 쉽지 않은 경우가 많기 때문에 품질 리스크를 줄이기 위해서는 결국 해당 부분은 제약사항으로 보고, 다른 부분에서 생산성을 향상시킬 수 밖에 없는 상황이다. 따라서, 산업계에서는 생산성 향상에 대한 니즈가 늘 존재해왔고, 이에 맞춰 소프트웨어 공학 학계에서도 생산성 향상에 관한 다양한 연구들이 진행되어 왔다. 그 중 본 연구에서는 품질 확보에 많은 리소스가 투입되고 있는 현실에 착안하여 버그 배정 과정에서의 생산성을 개선하는 방법을 찾아보고자 한다. 기존에도 버그 배정에 대한 다양한 연구들이 존재하지만 성능이 부족하거나 프로젝트의 특성이 너무 다른 탓에 실제 산업계에서 성공적으로 적용한 사례는 아직까지 보이지 않는다. 그러나 최근 머신 러닝 분야에 대한 연구가 활발히 진행되어 왔고, 머신 러닝을 생산성과 관련된 문제들에 적용할 때 효과적일 수 있다는 결과들이 나오고 있으며, 결과적으로 이러한 기존 연구들은 본 연구가 시작될 수 있도록 영감을 주었다.

본 연구에서는 산업계의 실제 버그 할당 프로세스 분석과 기존의 버그 자동 할당에 관한 연구들을 분석함으로서 얻은 통찰을 토대로 최신의 딥러닝 기술과 실제 산업계의 데이터를 활용하여 버그 할당 과정에서의 재할당 횟수를 줄일 수 있는지 검증하고자 한다. 또한, 기존 버그 할당 프로세스에서는 개발팀 할당이 정확하지 않은 경우 버그에 기술된 정보를 토대로 수동으로 분석하여 다른 개발팀으로 재할당해야 하는데, 버그 분석과 분석된 내용을 토대로 적합한 개발팀을 찾는 과정을 자동화하여 재할당 효율 자체를 높이고자 한다. 재할당 횟수를 줄일 수 있다면 버그 할당에 투입되는 전반적인 비용을 절감할 수 있을 것이고, 재할당 과정을 자동화한다면 버그를 수동으로 분석하고, 재할당하는 과정에 드는 비용을 개선할 수 있을 것이다. 또한, 본 연구에서는 실제 산업계 데이터를 활용하였고, 널리 사용중인 모델을 토대로 Practical하게 적용하는 것을 목표로 하기 때문에 기존의 연구들에서 제안된 모델들에 비해 산업계에 빠르고 쉽게 이식하기에 용이할 것이라고 기대한다.

2. 관련 연구

2.1 이력 기반 그래프 모델

그래프 모델은 버그 A를 개선하기 위해 수정 된 요소(component) B의 이력, 요소 B를 수정한 개발자 C의 이력 등 기존의 버그 검토 이력 데이터를 기반으로 구축되는 모델[1]이다. 그래프의 각 노드는 개발자, 컴포넌트, 버그로 구성되어 노드 간의 연결의 횟수에 따라 가중치(weight)를 부여한다. 이 모델을 활용하면 버그와 모듈, 모듈과 개발자 간의 연결 관계를 파악할 수 있고, 나아가서는 기존에 개선되었던 문제와 유사한 문제라고 식별이 되면, 본 그래프를 통해 최종적으로 어떤 개발자가 이슈를 개선하게 될지 추론할 수 있다.

또 다른 연구에서는 개발자 A에게 버그가 할당되었을 때, 개발자 B, 개발자 C 등으로 이동되는 이력 데이터를 기반으로 그래프를 구성하는 모델[2]이 존재하는데, 이를 통해 특정 개발자가 지정되었을 때, 최종적으로 어떤 개발자가 버그를 개선하게 될지 추론이 가능하여, 개발자간 이동하는 경로를 줄일 수 있고, 개발자들간의 연관성을 통해 개발자 그룹을 시각화 할 수 있다.

그래프 모델은 개발이 간단하고, 컴퓨터 자원이 많이 필요치 않다는 장점이 있지만 시작 노드의 정확도가 최종 결과에 큰 영향을 끼친다는 단점도 존재한다. 본 연구에서 해결하고자 하는 문제는 첫 번째 이슈 할당의 정확도를 신뢰하지 못하는 상황이기 때문에 그래프 모델은 적합하지 않다.

2.2 머신러닝 모델

딥러닝이 본격적으로 연구되기 전엔 문서 유사도를 머신러닝을 통해 분석한 뒤 유사한 문제가 할당되었던 이력 기반으로 버그 담당자를 찾는 연구들이 다수 존재하였다. Bug Fixer[1]는 버그에 기술된 소스 코드 정보, 버그 제목, 버그 설명 데이터를 활용하여 단어 단위로 토큰화(tokenization) 후 벡터 공간 모델(vector space model)을 활용하여 기존에 학습된 버그와 새로운 버그간의 유사도를 계산하는데, 문장 안에 특정 단어가 얼마나 등장하는지에 따라 가중치가 달라지는 단어 사용 빈도(occurrence)를 기반으로 계산하기 때문에 어떤 키워드에 의해 결과가 도출되었는지 추론 과정을 설명하기 쉬운 장점이 있으나, 한편으로는 단어의 연결로 이루어진 문장에 대한 context를 이해하지는 못한다는 단점이 있다. 실제 버그 데이터에는 기존에 발생하지 않았던 새로운 유형이 많이 발생하는 특성을 보이므로 유사도 기반의 모델보다는 맥락을 이해할 수 있고, 새로운 형태의 데이터에 강인한(robust) 딥러닝 모델이 더 적합하다고 볼 수 있다.

2.3 딥러닝 모델

딥러닝 모델은 버그에 기술된 내용과 실제로 문제를 개선한 개발자에 대한 데이터를 여러 개의 신경망으로 구성된 모델에 학습시켜 새로운 버그가 등록되었을 때, 학습시킨 모델을 통해 버그 개선을 할 확률이 높은 개발자를 예측하는 모델이다. 딥러닝 모델의 특성상 데이터의 맥락을 이해하거나 비선형적인 데이터에 강인한 특성을 보이기 때문에 데이터만 충분하다면 분야를 가리지 않고 우수한 성능을 나타내고 있다. 기존의 딥러닝 기반 버그 배정에 관한 연구들은 주로 버그 제목과 기술된 상세 설명 데이터를 위주로 학습에 적합한 형태로 전처리(tokenization, embedding) 후 최종 담당자를 결론으로 지도 학습을 진행하는 형태의 모델을 제안하고 있다.

기존에 제안된 자연어 처리 딥러닝 모델들은 크게 합성곱 신경망(convolutional neural network)[3], 순환 신경망(recurrent neural network)[4] 기반의 모델들로 나뉘는데, 문제의 형태에 따라 효과적인 모델이 나뉠 뿐 어느 한쪽이 압도하지는 않았으나, 최근에 등장한 트랜스포머 기반 모델은 자연어 처리 분야에서 의미있는 성과들을 내며 두각[5]을 드러내고 있다. 딥러닝 모델은 다양한 domain에 적용할 수 있는 이식성과 모델 적용이 쉽다는 장점이 있지만, 학습을 위해 컴퓨터 자원이 많이 필요하고, 모델의 추론 과정을 쉽게 설명하기 어렵다는 단점이 있다. 본 연구에서 해결하고자 하는 문제는 학습을 위한 데이터가 충분하고, 추론 과정보다는 실질적인 할당 수치 개선이 목적이므로 딥러닝 모델, 특히 최근에 좋은 성과를 내고 있는 트랜스포머 기반의 모델을 사용하기로 하였다.

3. 본 론

본 연구에서는 비교적 최근에 등장한 고성능의 트랜스포머 기반 딥러닝 모델을 활용하여 자연어 처리 부분을 담당하고, 자연어가 아닌 데이터는 일반적인 형태의 딥러닝 모델(fully connected neural network)을 활용한 뒤 두 모델의 결과를 합성하여 최종 결과를 출력하는 형태의 모델을 제안한다. 입력은 버그에 기입된 다양한 정보들을 사용하고, 출력으로는 해당 버그를 개선할 확률이 가장 높은 개발팀을 나타내게 된다.

기존의 버그 배정 프로세스와 다른 부분은 [그림 1]과 같이 버그를 등록한 사람이 경험적/테스트 케이스 기반으로 기입한 “기능 라벨”을 토대로 사전에 정의된 “라벨 – 개발팀 테이블”을 통한 단순 할당을 하는 프로세스에서 버그의 재현 경로, 증상, 재현 빈도 등 다양한 정보들을 토대로 개발팀을 추론하도록 변경하는

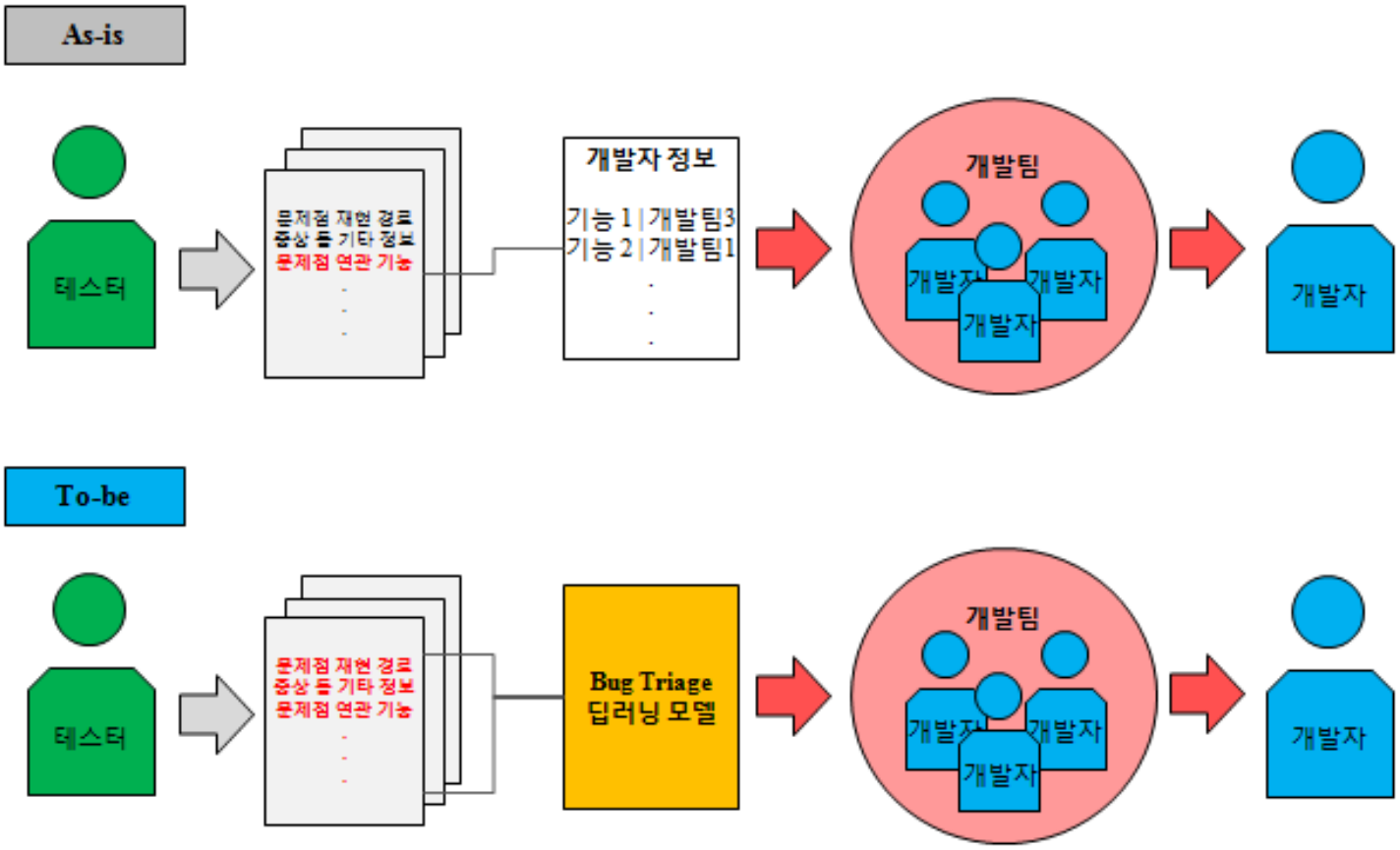


그림 1. 문제점 배정 프로세스

부분이다. 최종 개발자까지 추론해내지 않는 이유는 본 연구를 통해 개선하려고 하는 문제에서는 개발팀의 리더가 개발자를 할당하였을 때의 정확도가 매우 높으며, 문제가 개발팀의 리더를 거치지 않고 바로 할당될 경우, 할당된 개발자가 많은 버그를 갖게 되는 경우가 발생하여 병목으로 인해 버그 배정의 시간이 늘어나거나, 여유가 있는 다른 개발자에게 재할당하는 등의 overhead가 발생하는 경우가 빈번하기 때문에 이러한 제약사항들을 고려하였을 때, 가장 효과가 좋을 것으로 판단하였기 때문이다.

따라서 본 연구의 주요 기여(contribution)를 요약하면 다음과 같다.

- BERT 기반 자연어 처리 딥러닝 기술을 활용한 버그 배정 모델(A.I. Bug Triager) 개발
- 산업계에 효과적인 버그 할당 시스템 모델링
- 평가를 통해 산업계에 적용 시 효용성 여부 확인

3.1 문장형 데이터 처리 모델

3.1.1 합성곱 신경망 기반 모델

성능 비교를 위한 베이스라인 모델로 기존 연구 중 가장 유사한 연구에서 제안된 모델을 선택하였다. 일반적으로 자연어 처리 문제는 순환 신경망 기반 모델이 효과가 뛰어난 것으로 알려져 있으나 절대적이진 않으며, CNN 기반 모델을 제안한 연구에서 본 연구와 거의 유사한 형태의 데이터를 활용하였기 때문에 비교의 대상으로 적합하다고 판단하였다. 다만, 해당 연구에서 구현된 모델을 구할 수 없었기 때문에 유사한 파이프라인으로 토대로 모델을 구현하였다.

파이프라인의 구성은 문장형 데이터가 입력으로 전달되면 토크나이저를 통해 토큰으로 나눈 뒤 Word2Vec[6]을 활용하여 벡터 형태로 임베딩을 진행한다. 이후 합성곱 레이어(convolutional layer)를 통해 여러개의 필터(filter)로 추출(extract)한 뒤 특정 크기의 보폭(stride) 내에서 가장 큰 값을 선택(max pooling)한 뒤 하나의 긴 데이터로 잇는다(concatenation). 이후 소프트맥스 레이어(softmax layer)를 통해 다중 클래스 분류(multi class classification)을 진행한다.

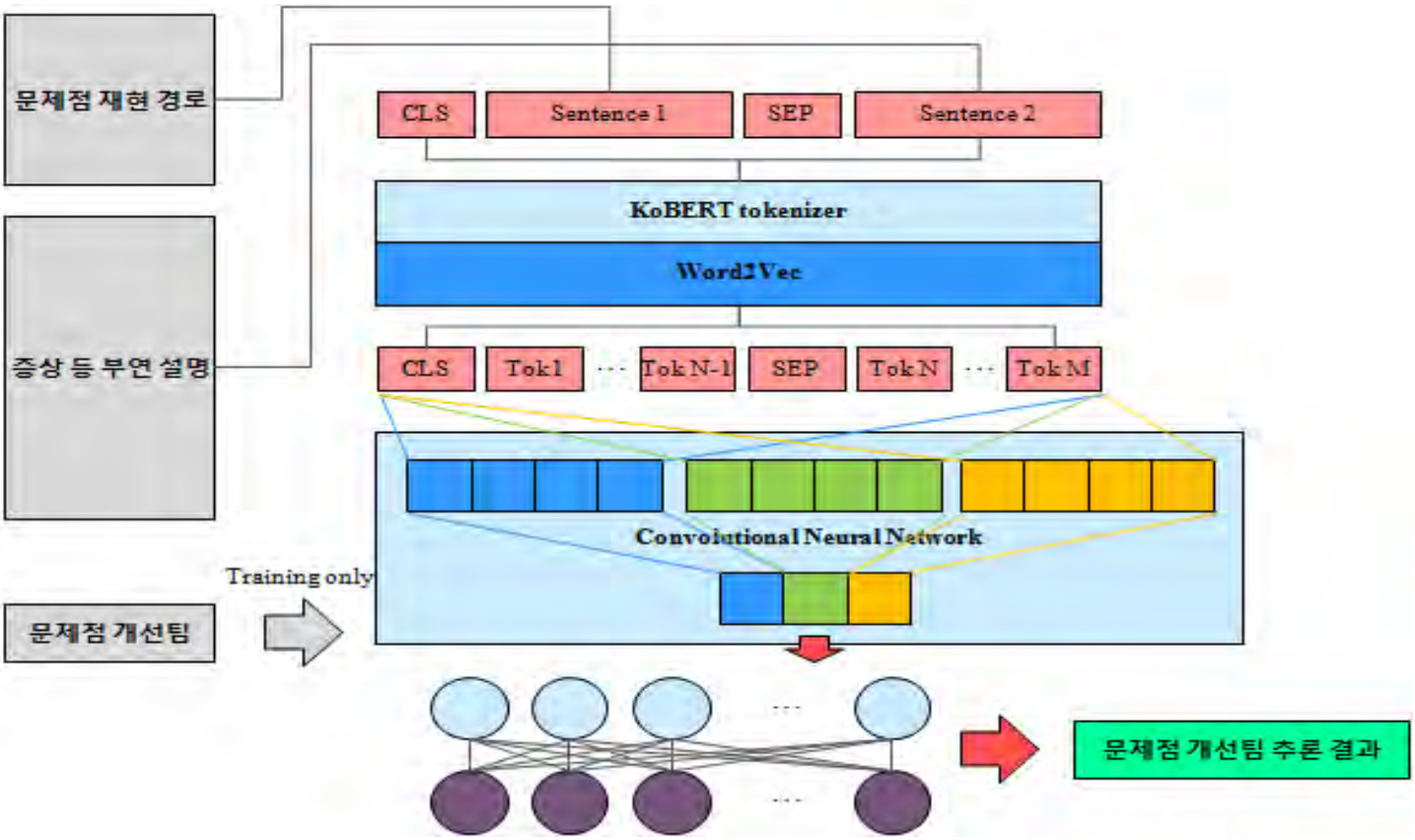


그림 2. Convolutional Neural Network 기반 모델

본 모델 구현을 위하여 토크나이저 성능 편차를 두지 않기 위해 KoBERT 토크나이저를 동일하게 활용하였고, 모델[7]을 활용하였다. 이후 각각 한 개씩의 단 차원 합성곱 레이어, 단차원 전역 최대 풀링 레이어(global max pooling layer), 소프트맥스 레이어로 구성하여 모델 개발을 완료하였다.

3.1.2 트랜스포머 기반 모델 (A.I. Bug Triager)

구글에서 발표한 BERT[5]를 기반으로 다양한 파생 모델이 발표되었는데, 이 모델들은 자연어 처리 분야의 권위 있는 대회(예: SQuAD, GLUE 등)에서 상위 순위를 독식하며 성능을 입증하였다. 또한, 본 모델은 학습을 위한 완성도 높은 참고 문헌들이 개발 커뮤니티 등에 퍼져있고, 인기 있는 모델인 만큼 다양한 국가에서 다양한 언어들에 특화된 파생 모델을 공개하고 있다. 한글이 문장 데이터의 주를 이루는 본 연구의 특성상 한글을 지원하는 BERT 모델을 탐색하였으며, 제한된 자원과 개발 난이도를 고려하여 한글 사전 학습 모델인 DistilKoBERT[8]라는 모델을 선택하였다. 본 모델의 특징은 한글에 특화된 BERT 모델인 KoBERT[9] 대비 경량화된 모델로 적은 컴퓨터 자원으로도 활용이 가능하며, 한글 데이터 해석에 적합하도록 사전 학습이 되어 있어서 별도의 학습이 필요치 않고, 모델에 입력으로 사용될 데이터의 형태가 단순하며 적합한 데이터 형태로 만들기 위한 전 처리기를 내장하고 있어서 사용이 쉽다는 점이다. KoBERT 토크나이저의 경우 한글 위키 + 뉴스 텍스트 기반으로 학습된 구글의 Sentence Piece[10] 기반 토크나이저이며, DistilKoBERT는 한국어 위키, 나무위키, 뉴스 등의 데이터를 토대로 사전학습 되었다. 본 연구에서는 문장에서 한글만을 추출한 뒤 [그림 3]과 같이 KoBERT 토크나이저를 활용하여 토큰화 및 임베딩을 진행하고, DistilKoBERT를 통해 대표 속성(representation)을 추출한 뒤, 해당 속성들을 회귀(regression)을 통해 다중 클래스 분류 문제를 해결하는 모델을 구현하였다. 한글만을 추출한 이유는 토크나이저가 특정 언어에만 특화되어 여러 언어가 섞인 본 문제에서 제대로 동작하지 않는 문제를 회피하고, 또한 데이터 특성상 한글이 주로 문장을 구성하고 있어서 본 연구에서는 과감하게 한글 데이터만 활용하기로 결정하였다.

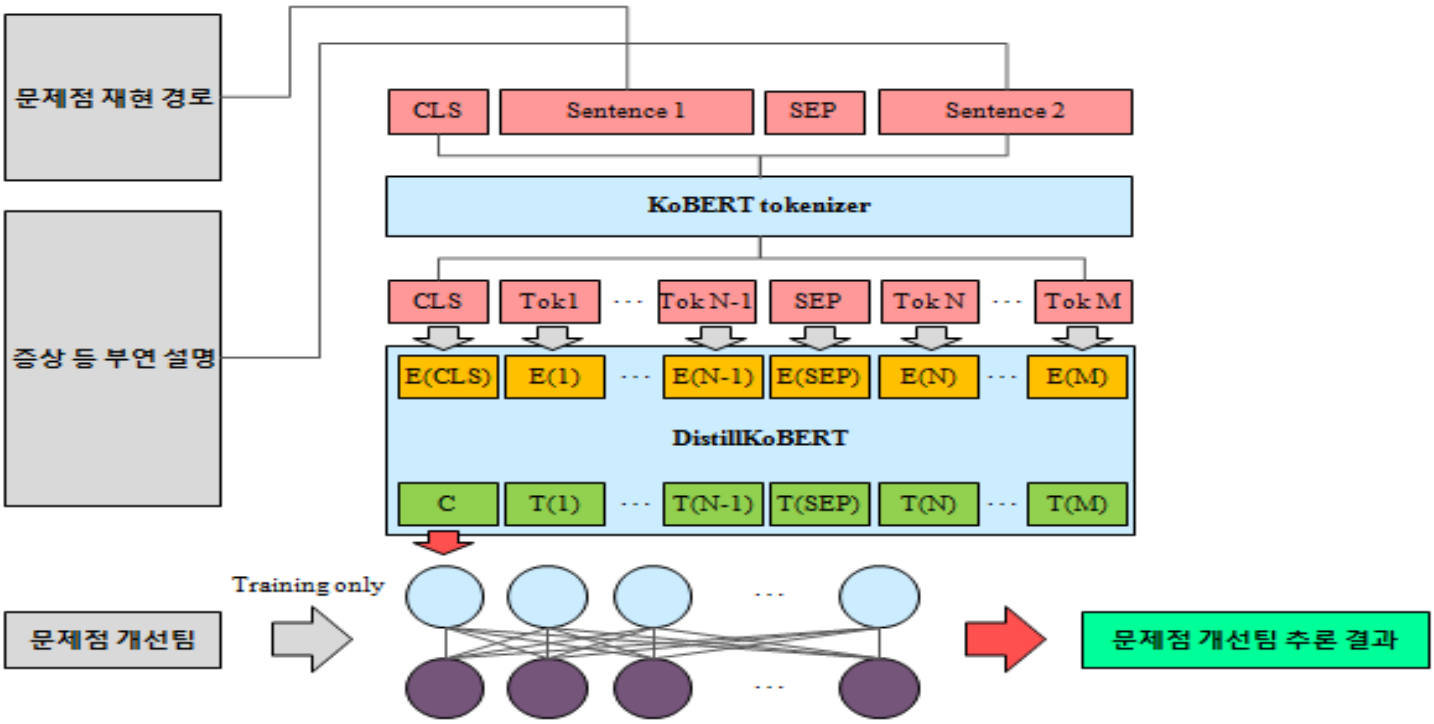


그림 3. BERT 기반 모델(A.I. Bug Triager)

입력 데이터는 “문제점 재현 경로”, “증상 등 부연 설명” 문장을 연결한 뒤 토큰화 및 임베딩하였으며, 문장 사이에는 [SEP] 토큰을 삽입하여 모델에서 구분할 수 있도록 처리하였다. BERT의 특성상 모든 입력 데이터의 차원을 맞춰야 하므로 토큰화 후 가장 긴 데이터를 기준으로 길이를 설정하였다. 길이가 짧은 문장들에는 빈공간을 채우는 패딩(padding)을 적용하고, 패딩 영역이 추론에 영향을 주지 않도록 어텐션(attention)을 적용하였다.

최종적으로 입력은 t(토큰의 수) × n(데이터의 수)의 차원으로 구성되고, BERT의 [CLS] 토큰의 트랜스포머 출력을 다시 입력으로 받아 완전 연결 레이어 및 소프트맥스 레이어를 통해 개발팀의 확률을 나타내는 c(클래스 수) × n(데이터 수)의 차원 데이터를 출력하는 모델로 구현하였다.

본 모델의 데이터 흐름을 다시 요약하면 문장 데이터를 입력으로 받아서 전처리(한글 추출, tokenization, embedding, padding, attention masking) 후 사전 학습된 BERT를 통해 대표 속성을 추출하고, 대표 속성과 기존 버그 개선 결과를 학습하여 개발팀별 버그를 할당 받게 될 확률을 출력한다.

3.2 비 문장형 데이터 처리 모델

비 문장형 데이터를 학습하기 위하여 일반적인 형태의 완전 연결 신경망 모델을 구현하였다. [그림 4]와 같이 5개의 레이어로 구성하였으며, 복수 개의 피쳐(feature)를 입력으로 받고 최종 출력은 자연어 처리 모델과 동일하게 개발팀별 확률을 출력하도록 구현하였다. 입력 데이터의 각 피쳐는 라벨 인코딩(label encoding)을 진행하여 f(피쳐 수) × n(데이터 수)의 차원으로 구성되고, 출력 데이터는 소프트맥스를 통해 개발팀의 확률을 나타내는 c(클래스 수) × n(데이터 수)의 차원으로 구성된다.

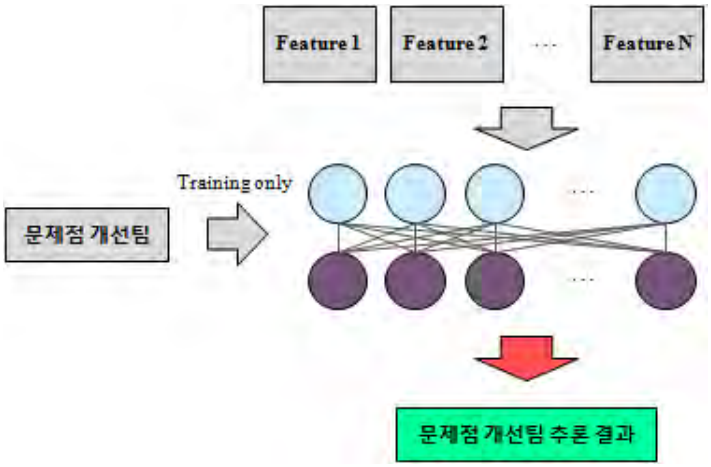


그림 4. 비문장형 데이터 처리 모델

3.3 합성 모델

앞서 설명한 두 개의 모델들의 결과를 합성하여 최종 결과를 출력하는 모델이다. 본 연구에서는 가중치

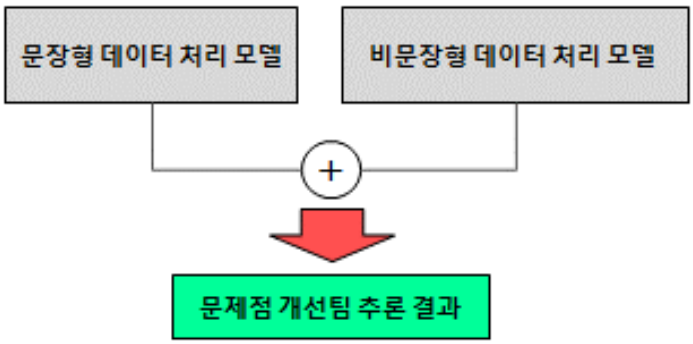


그림 5. 합성 모델

합(weighted sum) 방식으로 구현하였으며, 각 하위 모델에 적용할 가중치 값은 실험을 통해 더 좋은 결과가 나오는 값으로 선택하였다. 합성 모델의 [그림 5]와 같이 입력은 두 서브 모델의 출력 값을 토대로 개발팀별 최종 확률을 출력하도록 구현하였다. 다양한 조합으로 구성한 결과 문장형 데이터 처리 모델에 0.6, 비 문장형 데이터 모델에 0.4의 가중치를 주었을 때 가장 결과가 좋았고, 해당 값으로 선택하였다.

4. 결 과

4.1 실험 환경

본 연구에서는 제안된 모델의 성능을 측정하기 위해 실제 임베디드 소프트웨어를 개발하는 회사에서 특정 제품을 개발하는 과정에서 발생한 버그(약 2만개)을 활용하였으며, 문장형 데이터는 “문제점 재현 경로”, “증상 등 부연 설명”에 대한 기술 데이터를 입력 데이터로 활용하였고, 비 문장형 데이터는 “문제 심각도”, “재현 빈도”, “칩 종류”, “기능 대분류”, “기능 소분류”를 사용하였다. 최종 모델의 산출물은 문제를 개선할 것으로 예상되는 “개발팀”별 확률 값으로, 본 문제에서 사용한 데이터 기준으로 13개의 개발팀의 확률이 출력되게 된다. 학습과 평가를 위해서 데이터를 80:20의 비율로 나누어 진행하였으며, 10겹 교차 검증(10 fold cross validation)을 적용하였다.

각 모델의 컴파일 옵션으로 최적화 알고리즘(optimizer)은 adam을 활용하였고, 손실 함수는 categorical crossentropy, 그리고 지표(metric)는 정확도(accuracy)를 활용하였으며, 완전 연결 레이어(fully connected layer)의 활성화 함수(activation function)은 ReLU를 사용하였다. 또한 각 모델별 하이퍼 파라메터는 모델의 기본값을 활용하였으며 일반화(regularization)을 위한 dropout 등은 적용하지 않았다. 하이퍼 파라메터 튜닝은 본 연구에서 중요한 부분은 아니기 때문에 파인 튜닝을 통한 추가적인 성능 개선은 이후 연구로 남겨두었다. 추가적으로 구글의 콜랩(colab)을 활용하여 실험을 진행하였기 때문에 성능의 제약으로 인해 BERT 모델 활용 시 batch size는 64로 설정하였다.

4.2 데이터 분석

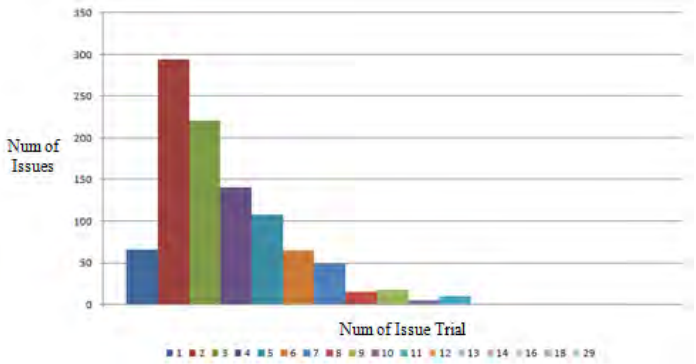


그림 6. 재할당 횟수에 따른 문제점 분포

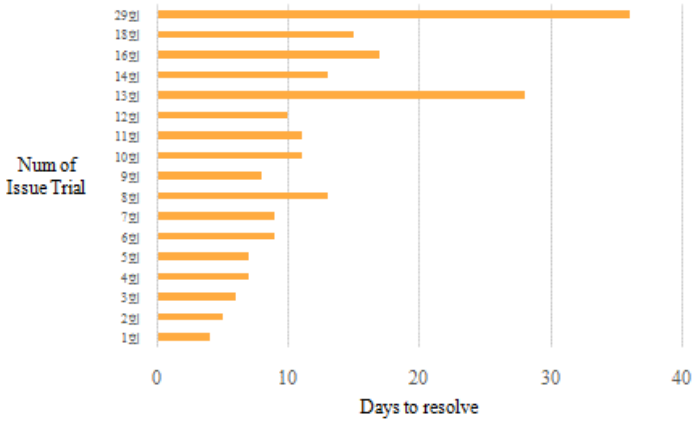


그림 7. 재할당 횟수에 따른 문제점 개선 소요 시간

실험에 앞서 의미있는 측정 지표(evaluation metric)를 설정하기 위하여 데이터 분석을 수행하였다. 재할당 횟수를 기준으로 데이터를 분석한 결과 [그림 6]과 같이 대부분(80% 이상)의 문제는 5번의 재할당 과정 안에 해결이 되었으며, 한번의 할당으로 개선되는 버그는 6% 수준에 불과하였다. 또한, [그림 7]과 같이 선형적으로 증가하지는 않았지만 대체적으로 재할당 횟수가 높으면 버그를 개선하는데 투입되는 시간도 늘어나는 추세도 확인할 수 있다.

이를 토대로 측정 지표는 Top-N 정확도(accuracy)로

정하였다. Top-N 정확도는 N번째 순위 까지의 결과 값 안에 참인 값이 나타난 비율을 나타내는 지표로 이를 식으로 표현하면 다음과 같다.

$$\text{Top} - \text{N accuracy} = \frac{1}{|Q|}\sum_{i=1}^{Q} assess\left(t_{\mathbb{I}(i)}, R_i^N\right) \quad (1)$$

$$\text{assess}\left(\text{t}_\text{i}, \text{R}_\text{i}^\text{N}\right) = \begin{cases} 1 & \text{if } \text{t}_\text{i} \in \text{R}_\text{i}^\text{N} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

수식 1의 Q는 테스트 수를 의미하며, R는 N번째까지 높은 순위의 결과 값들을 모아놓은 집합이다.

4.3 실험 결과

	Top-1	Top-2	Top-3	Top-4	Top-5
Manual	6.61%	36.14%	58.11%	72.29%	**83%**
Baseline Model	**31.01%**	**54.18%**	66.92%	75.20%	81.28%
A.I. Bug Triager	28.79%	53.59%	**67.62%**	**76.60%**	**82.84%**

표 1. 모델간 성능 비교

앞서 설명한 내용과 같이 본 연구의 목표는 재할당 횟수를 줄일 수 있는지에 있기 때문에 성능 측정은 Top-1~5 정확도를 기준으로 실험을 진행하였다. Top-1~4 정확도를 평가하는 이유는 재할당 횟수를 얼마나 줄였는지 평가하기에 가장 적절한 지표이기 때문이다. Top-5 정확도의 경우, 기존 데이터 분석에서도 알 수 있듯이 1~5회의 재할당이 발생한 버그가 대부분이기 때문에 Top-5 정확도가 높거나 동등하다면 자동 할당의 도구로서 사용이 가능한지를 가늠할 수 있는 중요한 지표로 판단할 수 있다.

[표 1]을 보면 Top-1 정확도에서 딥러닝을 활용한 모델을 통한 할당의 정확도가 수동 할당 대비 매우 높은 것을 알 수 있다. 다만, 본 연구에서 제안한 BERT 기반 앙상블 모델 대비 베이스라인 모델인 CNN 기반 앙상블 모델이 조금 더 나은 성능을 보여주고 있다. Top-2 정확도에서도 유사한 추이를 보이다가, Top-3~4 정확도에 이르러서는 수동 할당 대비 성능 편차는 줄어들었지만, BERT 기반 앙상블 모델이 가장 우수한 성능을 나타내고 있다. 이를 토대로 딥러닝 기반 버그 자동 할당 모델이 수동 할당 모델 대비 재할당 횟수를 줄임에 효과적임을 알 수 있다.

Top-5 정확도 결과를 보면 BERT 기반 앙상블 모델의 성능이 수동 버그 할당과 동등한 성능을 보여주고 있으며, 이를 토대로 보면 기존 수동 할당 시스템을 자동 할당 시스템으로 교체한다면 수동 할당에 투입되는 자원을 낭비 없이 온전히 효율화할 수 있음을 알 수 있고, 추가적인 모델 성능 개선을 통해서 그 이상의 효과를 기대할 수 있다.

	Top-1	Top-2	Top-3	Top-4	Top-5
문장형	25.24%	51.72%	62.26%	71.91%	79.24%
문장형 + 비문장형	**28.79%**	**53.59%**	**67.62%**	**76.60%**	**82.84%**

표 2. 비문장형 데이터 조합 성능 비교

추가적으로 문장형 데이터만 활용했을 때의 실험 결과를 확인해보았다. [표 2]를 보면 비문자형 데이터를 같이 활용한 모델 대비 평균적으로 모든 지표에서 조금씩 열위를 보이고 있음을 보여준다. 이를 통해 버그에 기술된 다른 데이터를 추가로 활용한 모델이 좀 더 나은 성능을 낸다고 볼 수 있으며, 기존 연구들에서 주로 사용하였던 오픈 소스 버그 데이터 대비 실제 산업계 데이터에는 중요한 정보들이 많기 때문에 기존 연구들을 기반으로 산업계 데이터에 맞게 개량한다면 충분히 의미 있는 결과를 낼 수 있음을 보여준다.

5. 결 론

본 연구에서는 기존 버그 할당, 개선 이력을 학습하여 최종적으로 버그를 개선할 것으로 예상되는 개발팀을 추론하는 딥러닝 모델을 제안하였다. 딥러닝 모델은 최근 자연어 처리 분야에서 두각을 드러내고 있는 트랜스포머 기반의 모델을 활용하였으며, 실제 산업계에서 수집된 데이터를 활용하여 좀 더 현실적인 연구를 진행해볼 수 있었다. 실험 결과를 통해 알 수 있듯이 본 연구에서 제안한 딥러닝을 통한 버그 자동 할당 모델은 기존 버그 할당에서 발생한 재할당 횟수를 줄일 수 있으며, 전체적인 정확도 관점에서 기존 시스템과 유사한 성능을 보여주었기 때문에 시스템 대체 시 자동화를 통한 효율성 개선의 효과를 온전히 볼 수 있음을 실험을 통해 증명할 수 있었다. 또한, 산업계 데이터, 프로세스의 특성을 고려하여 실질적인 효과가 있으면서도 유지보수도 용이한 모델을 제안하였고, 실제 산업계 데이터에서 얻을 수 있는 부가적인 정보가 실제 모델 성능 개선에 의미가 있음을 밝혀 내었다. 이를 통해 성능이 충분치 않아서 산업계에 적용할 수 없었던 기존의 유사 연구들도 기존의 모델 구조는 유지하면서도 산업계 데이터의 부가 정보를 활용한다면 추가적으로 성능 개선이 가능할 수도 있다는 가능성을 엿볼 수 있었다. 또한, 실험 여건상 모델 자체의 성능 개선을 위한 하이퍼 파라메터 튜닝을 진행하지 못하였음에도 실제로 사용할 수 있는 수준의 성능 결과가 나왔기 때문에 향후 튜닝을 통한 추가 개선에 대한 가능성도 열어두었다.

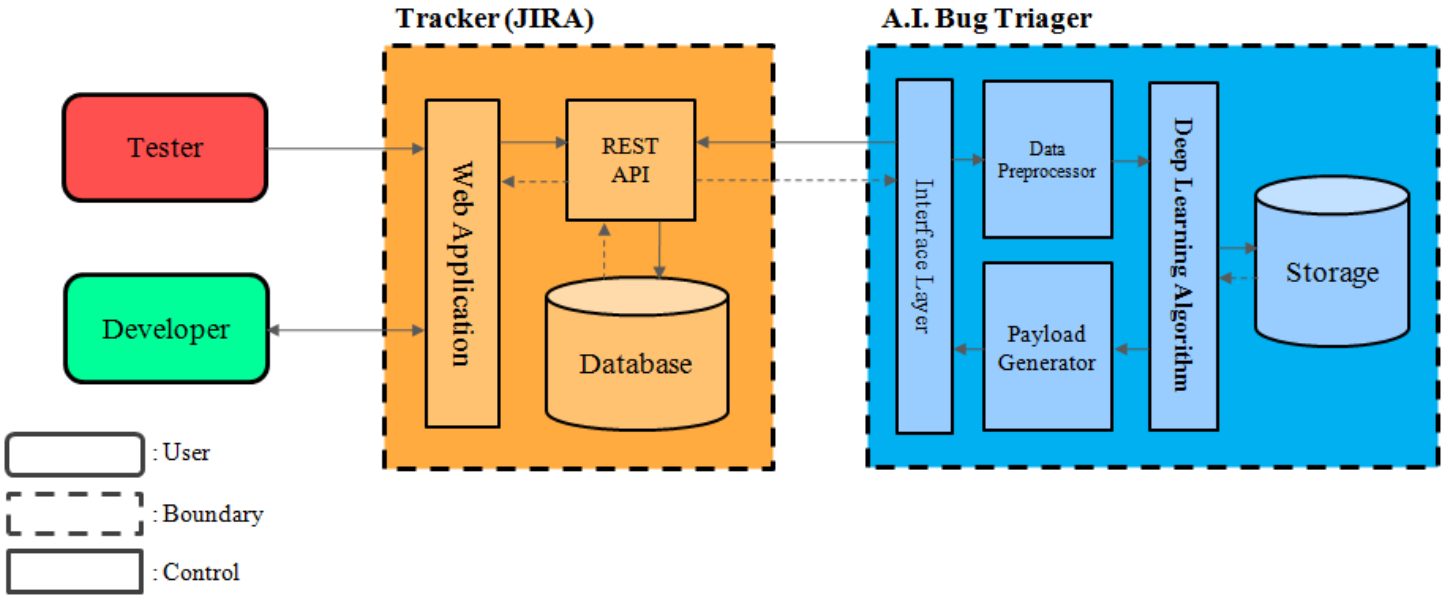


그림 8. 통합 이슈 관리 시스템

이후에는 언급한 하이퍼 파라메터 파인 튜닝, 한글 외 기타 언어로 이루어진 데이터 활용, 자연어 처리 모델 사용 시 사전 학습 모델이 아닌 소프트웨어 개발 도메인의 데이터 기반 학습, 합성 모델의 앙상블 학습 등을 통해 모델 자체의 성능 개선하고, 기존 버그 할당 시스템에 적용하여 버그 자동 할당 시스템을 구축한 뒤, 추론 과정에 대한 시각화를 통해 개발자나 매니저들이 결과에 납득하고, 더 나아가 검토에 잘 활용할 수 있도록 하며, 개발팀 할당뿐만 아니라 개선 예상 일정, 예상 버그 발생량 등 다양한 형태로 결과를 활용할 수 있도록 연구를 확장시켜 나갈 예정이다.

추가 연구를 통해 최종적으로 개발된 모델을 버그 배정 프로세스에 적용하였을 때 기대되는 모습은 [그림 8]과 같이 새로운 버그가 등록되면, 버그를 모니터링하고 있던 버그 자동 할당 모델이 이를 즉시 분석하여 버그를 개선할 것으로 예상(Top-1)되는 개발팀 리더에게 문제를 할당하면서 동시에 Top-2~5 개발팀 리더를 구독자(watcher)로 등록하고 코멘트로 알림을 보내도록 한다. 또한, 추론의 근거를 코멘트로 기입하여 개발팀 리더 혹은 개발자가 힌트로 활용할 수 있도록 한다. 동시에 예상되는 문제 개선일로 목표 일정(due date)가 설정되며, 프로그램 매니저에게 계획한 개발 계획에 위험성(risk)이 있는지에 대해 시각화하여 리포트한다. 마지막으로 실제로 버그가 개선된 후에는 개선된 버그를 새로운 입력 데이터로 학습하여 지속적으로 모델의 성능을 개선해나간다. 현업 종사자로서 현재 수준의 연구 결과를 동료들에게 소개하였을 때 매우 좋은 피드백을 받았으며, 시스템의 최종 형태에 대해 높은 기대감을 갖고 있음을 확인하였다. 따라서 추가 연구를 통해 통합 이슈 관리 시스템이 개발된다면 실제 산업계에서의 활용도는 매우 높을 것이라 예상하며 이 연구를 마친다.

참고문헌

[1] Hau Hu, Effective Bug Triage on Historical Bug-Fix Information, ISSRE, 2014

[2] Gaeul Jeong, Improving Bug Triage with Bug Tossing Graphs, ACM SIGSOFT, 2009

[3] Sun-Ro Lee, Applying Deep Learning Based Automatic Bug Triager to Industrial Projects, ACM SIGSOFT, 2017

[4] Senthil Mani, Exploring the Effectiveness of Deep Learning for Bug Triaging, ACM India, 2018

[5] Jacob Devlin, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, NAACL, 2019

[6] Tomas Mikolov, Efficient Estimation of Word Representations in Vector Space, ICLR, 2013

[7] Kyubyoung Park, Pre-trained word vectors of 30+ languages, https://github.com/Kyubyong/wordvectors, 2016

[8] Jangwon Park, DistilKoBERT: Distillation of KoBERT, https://github.com/monologg/DistilKoBERT, 2019

[9] SKT Brain, KoBERT: Korean BERT pre-trained cased, https://github.com/SKTBrain/KoBERT, 2018

[10] Taku Kudo, SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing, Conference on Empirical Methods in Natural Language, 2018

DNN 기반의 효율적 컴파일 에러 위치식별 방법

배민지 ○ 백종문
한국과학기술원
minjibae@kaist.ac.kr, jbaik@kaist.ac.kr

DNN Based Efficient Compilation Error Localization

Minji Bae ○ Jong-Moon Baik
KAIST

요 약

소프트웨어 개발 과정에서 컴파일 에러는 자주 발생하나, 컴파일러 에러 메시지만으로는 문제 해결이 어려운 경우도 있다. 에러 위치와 수정 방법을 제시하는 기존 연구들은 대부분 초보 프로그래머의 데이터를 활용하거나 특정 언어에만 적용이 가능한 제약 사항이 있어, 실무에서 수행하는 프로젝트에 실제로 제안을 바로 적용하기는 어렵다.

본 연구에서 제안하는 모델은 LSTM 모델을 기반으로 구성된 DNN을 이용하여 컴파일 에러 위치를 식별한다. 학습 데이터로 휴대폰 소프트웨어 개발 프로젝트의 안드로이드 빌드 에러 로그와 수정 코드 데이터를 사용하였으며, 실제 테스트 결과 52%의 정확도를 보여, 빌드 에러가 발생했을 때 빠르게 에러 위치를 식별할 수 있도록 돕는다.

1. 서 론

안드로이드 OS를 기반으로 한 휴대폰 소프트웨어 개발 프로젝트는 사용자의 다양한 요구사항을 반영하고 사업자의 새로운 서비스를 지원하기 위해 그 규모와 복잡도가 나날이 커지고 있다. 다양한 인원이 한 소스에서 동시에 개발을 진행하는 과정에서, 컴파일 에러는 피할 수 없는 문제이다.

프로젝트의 규모가 거대해짐에 따라 각 기능 소스를 수정 및 개발하는 개발자와 소스를 통합 및 빌드하는 개발자가 분리되어, 컴파일 에러 발생시 빌드 담당자가 여러 인원이 각자 수정한 소스 변경점들을 전체적으로 파악하고 분석해야 하는 상황을 초래하였다.

컴파일러에서 출력하는 에러 메시지가 항상 정확한 위치를 표시하지는 않기 때문에, 에러 메시지를 해석하고 에러 위치를 식별하여 최종적으로 해당 에러를 발생시킨 기능 및 개발자를 찾는 것에도 상당한 시간이 소요된다. 에러 위치를 식별한 후에도 에러 수정 시간과 컴파일 재작업 시간이 추가로 소요되기 때문에, 빠르면서도 정확하게 에러 위치를 식별하고 담당자를 찾는 것은 개발 일정 관리에 있어 중요하다.

2019년 한 해 동안 사내 빌드 시스템에 요청된 휴대폰 소프트웨어의 전체 빌드 개수와 그 결과를 취합한 결과, 월 평균 11934건의 빌드 중 2556건(요청 건의 약 21%)의 빌드가 실패하는 것을 알 수 있었다. 빌드 실패의 각 단계별로 에러 개수와 주요 원인을 살펴보니, 컴파일 단계의 에러가 그 중 847건(35%)으로 가장 많이 나타났다. 컴파일 단계에서 빌드 실패 후 그 지점부터 재빌드를 진행한 139건을 상세히 추적한 결과, 에러 없이 진행된 빌드 시간과 비교하였을 때 빌드 외에 1.8 ~ 3.8 시간이 추가로 소요됨을 알 수 있었다.

업무 구조상 수정 작업 없이 해당 에러 원인 코드 변경점만 제외하고 바로 재작업 후 개발자에게 이미지를 배포하는 경우가 대부분이기에, 재빌드 과정에서 빌드 외에 소요된 시간을 컴파일 에러 위치식별에 소요된 시간으로 판별하고, 이를 딥러닝으로 개선하기 위해 본 프로젝트를 시작하였다.

2. 제안 방법

본 연구에서 제안하는 DeepErrorFinder는 LSTM을 기반으로 하여, 실무 프로젝트에서 실제로 발생한 에러 로그와 수정 코드를 각각 추출하여 학습을 진행한다. 학습이 완료되면, 이후 발생하는 새로운 컴파일 에러에 대해, 해당 컴파일시 반영된 여러 개의 변경점 중 해당 에러를 발생시킨 변경점이 무엇인지 예측하여, 담당자가 그 에러 위치를 빠르게 찾도록 하는 모델이다.

2.1. Dataset

2017년 1월부터 2020년 8월까지의 빌드 에러와 수정 데이터를 취합 후, 그 중에서도 commit 정보가 있는 pair만 선별하여 총 1517 쌍의 데이터를 수집하였다. 이 중, 칩셋 의존성이 큰 modem 부분은 추후 과제로 넘기고 이번 프로젝트에서는 android 부분의 데이터 976쌍에 먼저 집중하여, 학습 및 평가에 사용하였다.

2.2. 전처리: 에러 로그 파싱

학습에 사용할 빌드 로그는 학습 과정에서의 불필요한 로드를 줄이기 위해 에러와 관계없는 빌드 진행 관련 로그 부분을 먼저 제거하였다. 추출한 로그에서 이후

상세 에러 내용을 파싱할 때는 아래 사항들을 고려하였다.

- 에러 로그에 포함된 경로 정보
- 에러 발생 파일명과 파일 별 에러 라인 정보
- 경로, 파일명, 라인 정보를 제외한 메시지 부분

2.3. 전처리: 코드 변경점 추출

본 논문에서는 코드의 변경 사항을 확인하기 위해 사내 휴대폰 소프트웨어 개발에 사용 중인 git과 gerrit 시스템을 이용해서 하기의 정보를 추출하였다.

- 파일의 경로 정보
- 변경 파일명 리스트
- 파일 별 변경 라인 리스트
- 코드 변경점

코드 변경점의 경우, 텍스트 변경과 구조적 변경을 같이 추적하기 위해 AST(Abstract Syntax Tree)를 활용하였다. 현업에서 C와 C++, JAVA 파일을 모두 사용하여 개발을 진행하기 때문에, 오픈소스 AST 툴 중 가장 많은 언어를 지원하는 GumTree 툴을 git diff 기능과 함께 사용하였다 [1].

2.4. 학습 모델

빌드 로그에서 코드 변경점을 예측해내기 위해, 번역기 모델에 주로 쓰이는 NMT(Neural Machine Translation) 모델을 사용하였다.

DeepErrorFinder 모델은 NMT 중에서도 tensorflow 기반의 LSTM-RNNs을 사용하였다. 3개의 LSTM encoder와 1개의 LSTM decoder로 구성되어 있으며, attention은 Bahdanau attention [2], optimizer는 Adam을 사용하였고, optimizer의 learning rate는 0.001로 설정하였다. 또한 과적합을 막기 위한 dropout은 0.2로 설정하여 진행하였다.

3. 실험 및 평가

이 프로젝트에서는 3겹 교차 검증으로 학습을 진행하였다. 학습 결과 정확히 오류가 발생한 경로와 파일을 맞추는 정확도는 52.4%로 나타났으며 (표 1 참조), 파일을 맞췄을 때 코드 변경 위치까지 정확히 맞추는 경우는 그 중에서도 평균 35.3%였다 (표 2 참조).

한 빌드 에러당 하나의 코드 변경점을 예측하고 있기 때문에 정밀도는 정확도와 동일하다. 재현율의 경우, 실제로 빌드 에러를 일으킨 코드 변경점이 1개가 아닌 경우에 예측하지 못한 변경점을 오답으로 판단하여 계산할 수 있다. 하나 이상의 정답이 있는 경우가 있음에도 제안 모델 구조상 하나의 변경점만 출력하기 때문에 F1 점수는 정확도 대비 낮은 수치인 0.46이다 (표 3 참조).

예측에 소요된 전체 시간은 평균 20건의 변경점 기준 전처리를 진행하기 위해 git 시스템에서 로컬로 코드를 다운로드하는 시간과 예측에 걸리는 시간을 합하여 약 32분, 즉 0.5 시간 정도이다.

표 1. 정확도 (파일 단위)

회차	평가 데이터수	맞춘 에러 파일	정확도
1-fold	325	171	52.6%
2-fold	325	173	53.2%
3-fold	325	167	51.3%
평균	325	170	52.4%

표 2. 정확도 (코드 단위)

회차	맞춘 에러 파일	코드 라인 위치	정확도
1-fold	171	76	44.7%
2-fold	173	49	28.3%
3-fold	167	55	32.9%
평균	170	60	35.3%

표 3. 정밀도와 재현율

회차	평가 데이터수	실제 파일	맞춘 파일	정밀도	재현율
1-fold	325	382	171	52.6%	44.8%
2-fold	325	460	173	53.2%	37.6%
3-fold	325	401	167	51.3%	41.6%
평균	325	414	170	52.4%	41%

6. 결론 및 향후 연구

이 연구에서는 실무에서 발생하는 실제 컴파일 에러 데이터를 기반으로 함으로써, 실무 활용이 가능한 모델을 제안한다.

수동으로 진행하는 위치식별과 비교하였을 때, 딥러닝을 통해 컴파일 에러 위치식별 수행시 최대 7.6배까지 시간 단축이 가능하다. 또한 파일 기준 정확도의 경우 파일 단위 예측을 기준으로 52%를 달성하여, 시간과 인력의 낭비를 줄이고 최종적으로 개발 시간 단축에 이바지할 수 있을 것이다.

향후에는 제안 모델을 실무에 적용하면서 추가 학습 데이터 확보 및 지속적인 학습을 통해 정확도를 높여 나갈 계획이다.

참고문헌

[1] FALLERI, Jean-Rémy, et al. Fine-grained and accurate source code differencing. In: Proceedings of the 29th ACM/IEEE international conference on Automated software engineering. p. 313-324. 2014.

[2] BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

ML을 활용한 Adaptive Streaming 재생 동기화

송기혁 ○ 백종문
한국과학기술원
Kihyouk.song@kaist.ac.kr, jbaik@kaist.ac.kr

Synchronization Playback of Adaptive Streaming on Multiple Clients using ML

Kihyouk Song○ Jongmoon Baik
KAIST

요 약

디스플레이 산업 중 비디오월 등의 디지털 사이니즈 제품들에서는 복수의 디스플레이에서 영상을 실시간으로 동시 재생하는 “동기화 재생” 기능이 사용되고 있다. 최근들어 미디어 스트리밍 방식의 동기화 재생에 대한 요구가 늘고 있으며, 이에 대한 솔루션으로는 Server-Side 기반의 IP Multicast 방식이 주로 사용된다. 하지만 IP Multicast 방식은 강력한 서버 성능, 안정적인 네트워크 대역폭 등 충분한 수준의 네트워크 인프라가 요구되며, 해당 스트리밍 프로토콜에 대한 이해를 바탕으로 한 복잡한 구현 난이도로 인해 주로 상용 미디어 솔루션 업체에 의해 제공이 되고 있다. 이에 본 연구는 현재 스트리밍 분야에서 널리 사용되고 있는 “HTTP 기반 적응형 스트리밍(HTTP Adaptive Streaming, HAS)” 즉 Client-Side 기반의 동기화 재생 솔루션을 제안한다. 본 연구에서는 오픈소스 기반의 HTML5 미디어 Player 형태로 동기화 재생 솔루션을 구현하며, 동기화 재생에 반드시 요구되는 통일성, 실시간성을 확보하기 위해 해결해야만하는 난제들을 휴리스틱과 머신러닝 두 가지 방식으로 접근하여 그 성능을 비교한다. 해당 솔루션은 표준 웹 기술을 사용하기 때문에 응용이 쉽고, 클라이언트 구현을 중심으로 일반적인 네트워크 환경에서도 서비스가 가능하며, 기존 HAS의 장점을 그대로 가져와 변화하는 네트워크 환경에 맞춰 동기화 재생을 할 수 있다.

1. 서 론

동기화 재생이란 두 개 이상의 디스플레이 디바이스에서 재생되는 미디어의 재생 시간을 동기화하여 매 시점마다 동일한 렌더링 결과를 표시하도록 하는 것을 의미한다. 이러한 미디어 동기화 재생은 디지털 사이니즈와 같은 디스플레이 제품군에서 주로 사용된다. 일반적으로는 비디오월, LED 전광판과 같은 대형 디스플레이 집합에 사용되지만 3~5대 정도의 소규모 구성 또는 단독 디스플레이 여러 대가 공공장소 등지에 산재하면서도 동일한 미디어를 재생하는 경우에도 사용되기도 한다. (그림 1)

그림 1. 동기화 재생 사용 예시

현재까지는 별도의 미디어 플레이어를 이용하여 영상을 HDMI/DP와 같은 HW 인터페이스를 통해 디스플레이에 분배하거나, 각 디스플레이에서 직접 로컬 파일을 재생하면서 네트워크를 통해 실시간으로 재생을 제어하는 방식 등으로 동기화 재생을 구현하고 있다. 하지만 전자의 경우 미디어 포맷이 HW 인터페이스 규격에 제한되고, 미디어 플레이어에서 출력하는 1개의 소스만 사용할 수 있으며, 미디어 플레이어 자체도 매우 고가의 외부 장비라는 단점이 있다. 후자의 경우는 별도의 외부 장비는 필요하지 않지만, 다수의 대용량 미디어 재생 목록을 사용하는 경우 저장 장치의 제약이 따르고, 각 디바이스에서 개별적으로 파일을 재생하는 만큼 완벽한 동기화 재생을 구현하기가 어렵다. 이에 최근 들어 미디어 스트리밍 기반의 동기화 재생에 대한 요구가 늘고 있다. 스트리밍 방식은 앞서 소개한 방식들이 가지는 단점들로부터는 어느 정도 자유롭지만 네트워크 환경에 영향을 많이 받는다는 점에서 위험 요소가 존재한다. 현재까지 동기화 재생에 주로 사용되는 스트리밍 프로토콜은 UDP나 RTSP와 같은 IP 멀티캐스트 방식이다. 이는 멀티캐스트 전용 IP 대역을 통해 서버가 미디어 스트림을 클라이언트로 밀어 넣는

방식인데, 공중파 방송 등에 버금가는 뛰어난 실시간성으로 어떤 인위적인 추가 구현 없이도 동기화 재생이 가능하다. 하지만 보장된 성능을 위해서는 요약에서도 언급했듯이 안정적으로 스트리밍 서비스가 가능한 네트워크 인프라를 위한 비용이 필요하다. 그렇지 않을 경우, 랜덤 패킷 로스와 같은 문제에 의해 영상의 프레임이 훼손되는 신뢰성 저하나 서버와 클라이언트의 동기화가 틀어져서 실시간성이 깨지는 문제가 발생하기도 한다. 그리고 서버 중심의 방식이기 때문에 IP 멀티캐스트에 대한 기술 이해를 바탕으로 서버와 클라이언트가 함께 구현되는 미디어 솔루션 형태로만 서비스가 가능하다. 그래서 최근 일반 스트리밍 분야에서는 Youtube, Netflix 서비스와 같은 HTTP 기반의 적응형 스트리밍(HAS) 방식이 각광받고 있다.

그림 2. HTTP Adaptive Streaming(HAS) 개요

HAS는 ABR 알고리즘을 통해 스트리밍 중인 미디어의 비트레이트를 변화하는 네트워크 환경에 맞춰 가변한다. (그림 2) 네트워크 트래픽이 원활한 경우 높은 비트레이트의 미디어 청크를 전송하여 고해상도의 고품질 영상을 재생하고, 반대의 경우에는 낮은 비트레이트의 청크를 전송하여 비교적 저품질의 영상을 재생하여 버퍼에 스트림이 고갈되어 발생하는 리버퍼링 현상을 최대한 회피하며 사용자에게 서비스한다. 또한 서버는 HAS를 위해 비트레이트 별로 인코딩 된 미디어에 HTTP로 접근만 할 수 있으면 표준 웹 기술을 사용한 클라이언트 구현만으로 다양한 서비스를 개발할 수 있다. 그리고 IP 멀티캐스트와 달리 일반적인 네트워크 환경에서도 스트리밍이 가능하다는 장점이 있다.

두 개의 클라이언트 및 Youtube의 라이브 방송으로 동기화 재생을 시도할 경우, 라이브 영상임에도 두 클라이언트의 시간 차이가 현저하게 발생하며, 클라이언트마다 네트워크 처리량이 다르기 때문에 ABR 동작에 의한 비트레이트 가변으로 클라이언트가 서로 다른 품질로 영상이 재생되고 있음을 알 수 있었다.

그림 3. 재생 시간 및 비트레이트 차이

이렇게 현재 HAS로 다중 클라이언트에서 스트리밍을 수행하는 경우 클라이언트 간 재생 시간 편차가 발생하고 서로 다른 비트레이트로 스트리밍이 되면서 전체적인 통일까지 저하되는 현상이 나타나고 있다. 그래서 동기화 재생을 HAS로 구현하기 위해서는 3가지 목표를 정의한다.

- HAS 동작 방식을 이해하고 그에 맞는 동기화 재생 스킴을 구현
- 모든 클라이언트의 스트리밍 비트레이트를 통일하면서도 적절한 스트리밍을 확보
- 동기화 재생의 가장 기본이 되는 요소인 동시 재생을 구현

이에 본 연구는 HAS 동기화 재생 클라이언트를 직접 구현하고, 위 비트레이트 결정 로직과 동시 재생 로직을 구현하기 위해 머신러닝을 사용한 방식과 머신러닝을 사용하지 않고 단순하면서도 직관으로 구현한 방식, 이렇게 두 가지 방식으로 접근하여 그 성능을 비교하고자 한다.

2. 관련 연구

관련 연구를 참조하고자 하는 노력은 연구 전반에 걸친 방법론에 대한 아이디어를 얻거나 비교 분석을 위해서가 아닌 서론에서 언급한 주요 목표들을 해결하기 위해 부분별로 이루어졌다. 먼저 HAS로 동기화 재생을 구현하는 시도는 [1]에서 찾아볼 수 있었다. 해당 특허에서는 서버가 미디어 세그먼트의 인덱스를 직접 지정하여 모든 클라이언트가 해당 세그먼트를 일괄적으로 받아가는 방식이다. 또한 미디어 세그먼트의 크기가 크면 동일한 클라이언트의 버퍼 내에서 재생하는 시간이 길어져 클라이언트 간 재생 시차가 발생할 수 있기 때문에 미디어를 더 작은 크기의 청크로 분할 인코딩한다. 이로 인해 더 자주 서버에 엑세스하게 되며 이때마다 서버에 의한 동기화가 수행된다. 해당 특허는 서버가 직접 중심이 되어 동기화 재생을 제어하는 명확하고 심플한 방법이다. 하지만 여기서는 각 클라이언트의 네트워크 상황이 다르기 때문에 비트레이트 별로 각각 인덱스가 유지 및 관리되어 클라이언트마다 스트리밍되는 인덱스는 매 순간 동일하지만 비트레이트는 상이해져 재생 품질이 달라질 수 있다. 또한 각 클라이언트의 버퍼에 누적된 스트림이 실제 렌더링 되는 시점에

발생할 수 있는 미세한 오차를 조정할 수 있는 방법은 나와있지 않았다.

두번째는 비트레이트 최적화에 관련한 연구이다. 리버퍼링 없이 높은 비트레이트로 스트리밍할수록 사용자가 체감하는 품질은(QoE) 높아진다. [2]에서는 강화학습을 사용하여 클라이언트의 네트워크 상황에 가장 적합한 비트레이트로 스트리밍을 할 수 있는 방안을 소개한다. [3]에서 제시한 A3C 기반 뉴럴 네트워크를 사용하여, 스트리밍 중인 클라이언트의 버퍼 상태나 네트워크 처리량을 입력 State로 받아서 1-depth CNN에 입력하면 현재 스트리밍 환경에 가장 적합한 다음 청크의 비트레이트를 액션으로 도출하게 된다. 높은 비트레이트의 세그먼트를 서비스할수록 높은 보상을 받고, 선택한 비트레이트가 클라이언트의 네트워크 처리량을 초과하여 리버퍼링이 발생한 경우 페널티를 받는 방식으로 학습과 액션을 동시에 수행한다. 해당 연구는 기존 규칙 기반 ABR 알고리즘 대비 12~25% 가량 향상된 성능을 보여준다고 밝히고 있으며, 이 모델을 적용할 경우 동기화 재생 시 발생할 수 있는 잠재적인 리버퍼링 위험성, 혹은 이를 회피하기 위해 너무 낮은 비트레이트만을 지향하게 되는 문제를 효과적으로 개선하여 스트리밍 품질을 높일 수 있다는 판단이다. 하지만 해당 연구는 동기화 재생이 아닌 개별적인 스트리밍 시나리오가 타겟이기 때문에 본 연구에 적용 시에도 동일한 효과가 얻을 수 있을지는 확실하지 않았다.

마지막으로 [4]의 경우 어떤 연구는 아니지만 오픈소스 미디어 프레임워크인 GStreamer에서 실시간 동기화 재생을 수행하는 방식에 대한 내용이다. Gstreamer는 Demuxer와 직접 연결되는 미디어 파이프라인을 다루는데, 먼저 각 파이프라인의 내부 클락을 동기화하기 위해 주 클락의 시간 값을 샘플링한 후 최소제곱법 선형 회귀를 수행한다. 예를 들면 AV Sync를 위해 비디오 스트림과 오디오 스트림 각 파이프라인의 PCR을 선형 회귀 모델을 통해 기준이 되는 주 클락에 근사하도록 조정하는 경우이다. 이러한 디바이스 내부적인 동작 외에도 NetClock이라는 모듈을 사용하여 기준이 되는 주 클락을 네트워크 상의 다른 디바이스의 것으로 참조하여 복수의 디바이스에서 클락을 동기화 하는 것이 가능하며 이를 통해 동기화 재생이 가능하다. (디바이스 간 클락 샘플링은 UDP 패킷을 주고 받으며 이루어진다) 이와 같은 동작성은 실시간 동기화 재생의 중요한 아이디어지만 미디어의 PTS가 보정된 PCR을 스스로 따라가게 되는, 즉 Demuxer와 직결되는 Backend 프레임워크여서 가능한 방식일 수 있어 본 연구에서 구현하고자 하는 Frontend 웹 어플리케이션에 직접적으로 적용하기에는 제한이 따를 것으로 예상되었다.

3. 구 현

구현은 관련 연구들을 적절하게 응용하여 앞서 언급한 세가지 목표들을 해결하는 방향으로 진행했다. 먼저, 현재 개별 재생에 포커스되어 있는 기존 HAS 클라이언트에 어떤 동기화 재생 스킴을 적용했는지 소개한다.

크게 셋업 페이즈, 스트리밍 페이즈, 플레이백 페이즈로 구분할 수 있다. 셋업 페이즈에서는 클라이언트 들의 마스터, 슬레이브 역할을 설정하고 클라이언트 간에 P2P 통신을 위한 연결을 수립한다. 스트리밍 페이즈에서는 모든 클라이언트가 매번 동일한 비트레이트의 청크를 가져오는 동작을 통해 각 클라이언트의 버퍼에 동일한 스트림을 구성하는 것이 목표이다. 스트리밍과 동시에 플레이백 페이즈도 함께 진행되는데, 여기에서는 클락 동기화를 통해 실시간 동시 재생을 수행한다. (그림 4)

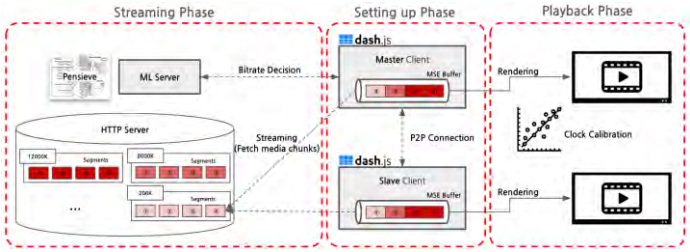


그림 4. HAS 동기화 재생 클라이언트 개요

구현은 오픈소스인 dash.js[5]를 기반으로 레퍼런스 클라이언트를 수정하는 방식으로 진행했고, 앞서 언급한 바와 같이 스트리밍 페이즈와 플레이백 페이즈의 핵심 로직들은 머신러닝을 사용한 방식과 머신러닝을 사용하지 않은 방식으로 구분해서 구현했다.

3.1 Setting up Phase

[1]에서는 서버가 동기화 재생을 제어했다면 본 연구에서는 클라이언트 중 마스터 역할을 갖는 1개의 클라이언트가 제어하도록 했다. 핵심은 그 외의 슬레이브 클라이언트들이 마스터 클라이언트와 직접 P2P로 연결되는 점이다. 이를 위해서는 사전에 모든 클라이언트들이 동기화 재생을 위해 구현된 WebSocket 서버에 접속하는 과정이 필요하다. 가장 먼저 접속하는 클라이언트가 해당 서버로부터 마스터 역할을 부여 받게 된다. (그림 5 좌측) 이후에 접속하는 클라이언트들은 슬레이브로써 서버를 통해 마스터에 P2P 연결 요청을 하게 되고, 마스터가 그에 응답하게 되면 해당 Slave와 페어로 직접 연결이 수립된다. (그림 5 우측) 이 P2P 연결에는 webRTC 프로토콜이 사용되며, 데이터 채널을 통해 별도의 서버 없이 클라이언트 간 직접 메세지를 주고 받게 된다. 동기화 재생에 참여하는 모든 클라이언트가 마스터(1):슬레이브(N) 관계로 P2P 연결이 완료되면 재생이 시작되며 이후의 동기화를 위한 동작은 데이터

채널을 통해 마스터를 중심으로 제어된다.

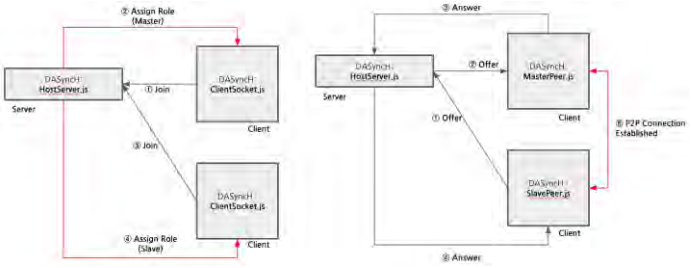


그림 5. 클라이언트 간 역할 설정 및 P2P 연결 설정

3.2 Streaming Phase

다음은 미디어 서버 또는 CDN으로부터 미디어 세그먼트를 받아와 버퍼에 저장하는 스트리밍 단계이다. 이 단계의 동기화 재생 목표는 모든 클라이언트가 동일한 재생 품질로 재생되도록 하는 것이다. 이를 위해서는 모든 클라이언트들이 매순간 동일한 인덱스, 동일한 비트레이트의 미디어 세그먼트를 요청하는 동작이 필요하다. 그리고 그 비트레이트는 모든 클라이언트들의 네트워크 처리량을 고려하여 한 클라이언트라도 리버퍼링이 발생하지 않는 수준으로 결정되어야 한다. 여기서부터는 휴리스틱과 머신러닝 두 가지 방식으로 구현한다. 휴리스틱 메소드는 클라이언트들이 각자의 ABR 알고리즘으로 자신의 네트워크 상황에 적합하다고 결정한 비트레이트를 마스터가 취합하고, 이 중 가장 낮은 비트레이트를 선택하여 슬레이브에 전달하면 (그림 6) 모든 클라이언트가 해당 비트레이트로 세그먼트를 요청한다. 머신러닝 메소드를 사용하는 방식은 [2]에서 제안된 강화학습 모델 “Pensieve”를 응용하여 모든 클라이언트의 네트워크 처리량 데이터를 마스터를 통해 전달 받고 이를 통해 비트레이트를 결정하는 방식이다(그림 7).

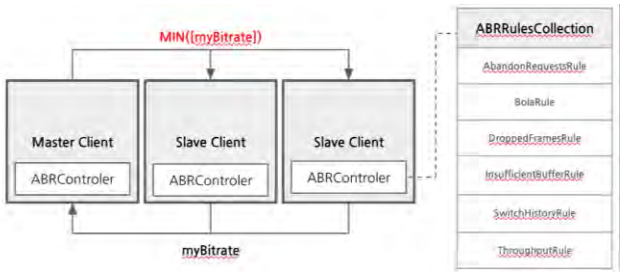


그림 6. 비트레이트 결정(1): 최소값

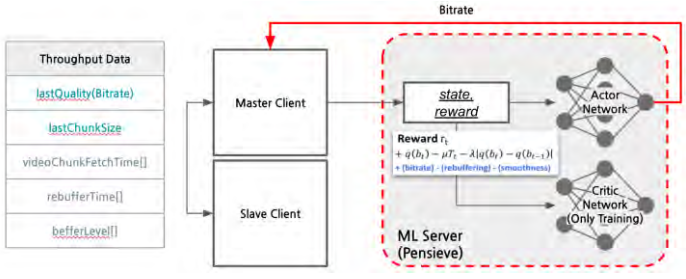


그림 7. 비트레이트 결정(2): 강화학습 모델에서 판단

3.3 Playback Phase

버퍼에 저장되는 스트림 데이터를 동기화했다면, 다음은 버퍼의 스트림을 렌더링하는 단계이다. 여기서는 동시 재생 즉, 동시 렌더링을 위해서 클라이언트들이 기준이 되는 마스터의 클락으로 동기화하는 동작을 하게 된다. 여기서는 모든 디바이스의 클락이 일치하지 않고 조금씩 틀어져 있다는 사실을 전제한다. 이 시간 차이가 100밀리초만 되도 미디어 재생 시 3~6프레임 가량 차이가 발생한다. 그렇기 때문에 슬레이브는 마스터의 시간을 알기 위해서 마스터에게 주기적으로 시간을 물어보고, 마스터와 자신의 시간이 얼마나 틀어져있는지 오프셋을 구하는 동작이 필요한데, 이상적인 상황에서는 이 물어보고 받는 라운드 트립 시간이 항상 균일하고 일정하기 때문에 이 라운드 트립의 중간 지점을 기준으로 마스터 시간과의 오프셋을 구하면 정확하게 마스터의 클락과 동기화를 할 수 있다. 하지만 현실의 네트워크에서는 이 물어보고 받는 시간이 항상 일정하지 않는 지터(Jitter)라고 하는 지연변이 현상이 존재하기 때문에, 마스터의 시간과 자신의 시간이 정확히 얼마가 차이가 나는지 알 수가 없다. (그림 8 좌측) 그래서 휴리스틱 메소드의 경우는 일단 이상적인 상황을 가정해서 500ms마다 마스터에게 물어보고, 라운드 트립의 중간 값을 기준으로 오프셋을 구하고 이 오프셋으로 슬레이브 시간을 보정하게 된다. 머신러닝 메소드의 경우 바로 오프셋을 구하는 게 아니고, 먼저 마스터가 알려주는 마스터의 시간과 라운드 트립에 걸린 시간의 중간 값을 윈도우 사이즈 32개로 샘플링을 하면서 선형회귀 모델을 만든다. 이 모델을 가지고 켈리브레이션이 필요한 시점의 슬레이브 시간 값을 x값으로 넣었을 때의 마스터 시간 추정 값 y를 도출해서 이 두 값으로 오프셋을 구하는 방식이다. (그림 8 우측)

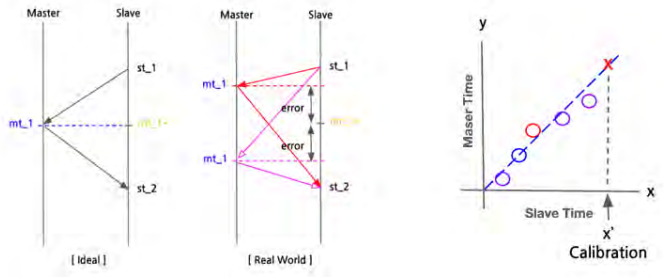


그림 8. 샘플링 과정에서의 지연 변이와 선형회귀 분석

4. 성능 평가

앞서 두 페이즈에서 구현한 휴리스틱 메소드와 머신러닝 메소드를 비교하는 형식으로 실험을 진행했다.

4.1 클락 동기화 성능

먼저 클락 동기화 평가를 위한 환경이다. 하나의

디바이스에서 마스터와 슬레이브를 함께 실행한 루프백 환경에서 테스트를 했다. 이렇게 할 경우 마스터와 슬레이브의 클락이 동일하기 때문에, 레이블을 알 수 있게 된다. 레이블이 있기에 위 메소드들로 마스터 클락과 동기화를 했을 때 실제 마스터의 시간과 얼마나 차이가 발생했는지 확인이 가능해진다. 다만 루프백 환경에서는 이 라운드 트립에 걸리는 시간이 거의 없다시피 할 정도이기 때문에 P2P 메세지를 주고 받는 부분에 딜레이를 랜덤하게 추가해서 현실에서의 네트워크 트래픽과 지터 현상을 재현했다. 이렇게 딜레이가 없는 <Case 1>, 50~200ms의 딜레이를 추가한 <Case 2-a>, 50~300ms의 랜덤 딜레이를 추가한 <Case 2-b> 이렇게 총 세 가지 케이스로 나눠서 각각 1000회씩 테스트를 진행했고, 실제 마스터 클락과의 오차 수준을 MSE와 MAE로 평가했다.

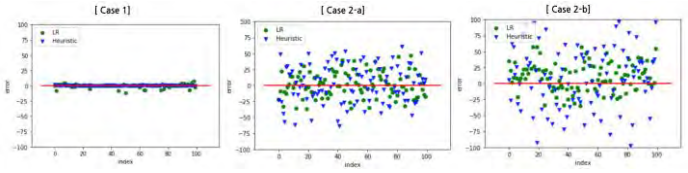


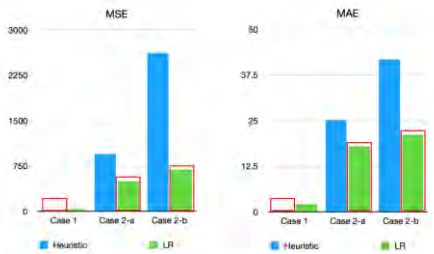
그림 9. 케이스 별 각 메소드의 오차 분포

각 케이스 별 오차 분포를 살펴볼 경우, 케이스 1의 경우 Round Trip 딜레이가 거의 없는 만큼 오차도 굉장히 작게 나오는 수준이나, 케이스 2와 함께 전체적으로 살펴보면 딜레이가 증가할수록 오차 분포가 증가하는 모습을 볼 수 있다. (그림 9)

표 1. 케이스 별 각 메소드의 MSE, MAE 지표

MSE	Heuristic	LR
Case 1	0.48316	33.77557
Case 2-a	945.90851	494.99169
Case 2-b	2,618.71452	689.90157

MAE	Heuristic	LR
Case 1	0.35418	2.16773
Case 2-a	25.12051	17.91329
Case 2-b	41.71996	21.15539



전체 테스트에 대한 MSE와 MAE 계산 결과는 케이스 1에서는 휴리스틱 메소드가, 케이스 2에서는 머신러닝 메소드가 더 작은 오차 수준을 나타낸다. (표 1) 특이점은 딜레이가 증가할수록 휴리스틱 메소드는 오차 수준도 큰 폭으로 증가하는 반면에 머신러닝 메소드의 오차 증가폭은 상대적으로 좀 더 로버스트한 것을 볼 수 있고, 케이스 1보다는 케이스 2가 현실적인 환경과 유사하기 때문에 선형회귀 분석을 사용한 머신러닝 메소드가 성능 면에서 우위에 있다고 평가할 수 있다.

4.2 비트레이트 결정 성능

다음은 비트레이트 결정 관련 평가를 위한 환경이다. 기본적으로는 [2]의 평가 환경과 동일하지만, 클라이언트 하나가 아닌 동기화 재생을 위한 두 개의 멀티 클라이언트로 실험을 진행한 차이점이 있다. 테스트 데이터 역시 논문과 동일한 FCC 브로드밴드, 노르웨이 HSDPA 데이터셋을 동일하게 사용했다. 여기서 말하는 테스트 데이터는 네트워크 처리량이 타임라인으로 기록된 네트워크 트레이스 형태인데, 동일한 네트워크 환경에서 어떤 비트레이트 결정 방식이 더 높은 품질로 스트리밍을 서비스하는지 평가하기 위함이다. Mahimahi[6]라는 네트워크 에뮬레이션 툴을 사용해서 이 데이터셋의 네트워크 환경을 그대로 재현을 하고, 휴리스틱 메소드와 머신러닝 메소드로 각각 스트리밍 동기화 재생을 수행했다. (그림 10)

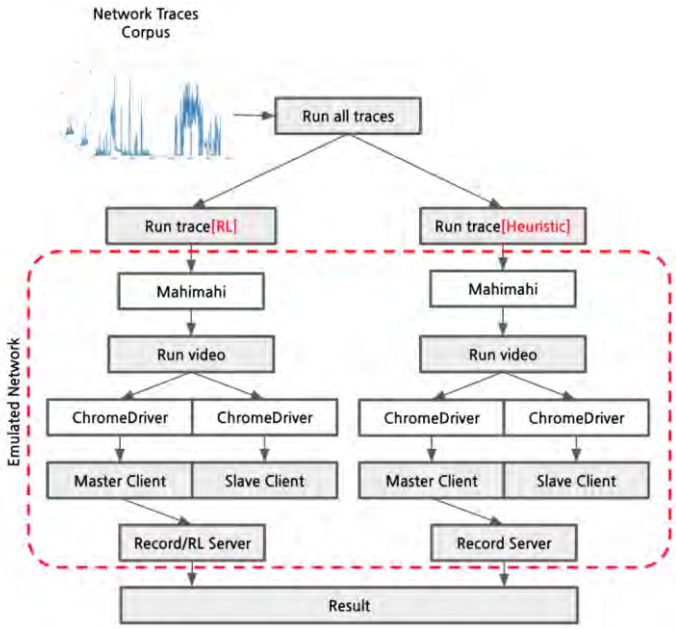


그림 10. 동일 네트워크 환경 재현 및 검증 실험

평가 지표는 [2]의 리워드 산출 방식과 동일한 스트리밍 퀄리티이며, 청크별로 평가한 리워드를 모두 서메이션 해서 스트리밍 1회에 해당하는 스코어를 산출했다. 여기서 말씀드리는 스트리밍 퀄리티는 얼마나 높은 비트레이트의 청크를 딜리버리 했는지, 버퍼링이 얼마나 발생했는지, 비트레이트가 너무 자주 가변이 되는건 아닌지를 나타낸 스코어이다. (1) QoE = SUM(Bitrate - Rebuffering Time - Smoothness)

$$QoE = \sum_{n=1}^{N} q(R_n) - \mu \sum_{n=1}^{N} T_n - \sum_{n=1}^{N-1} \left| q(R_{n+1}) - q(R_n) \right| \quad (1)$$

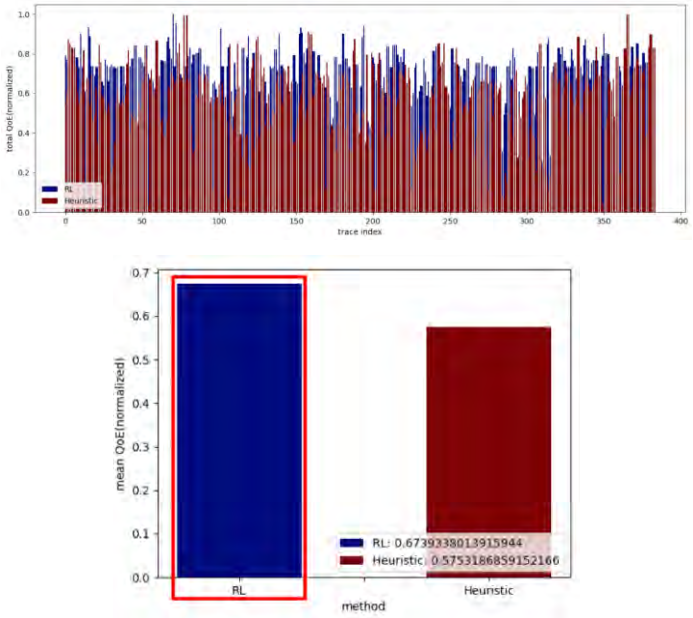


그림 11. 전체 결과(Normalized QoE)

총 380개 정도의 네트워크 환경에서 스트리밍을 진행했다. 대체적으로 파란색의 머신러닝 메소드가 높은 퀄리티를 기록하는 것을 볼 수 있고, 전체 평균치 역시 머신러닝 메소드가 높게 나타났다. (그림 11) 이 데이터셋으로 재현한 네트워크 환경이 그렇게 좋지 못한 상황에서 클라이언트 두 개로 열악한 대역폭을 나눠서 스트리밍을 하다 보니 평균 RAW 스코어는 전반적으로 낮았다. (휴리스틱 메소드: -51.3478, 머신러닝 메소드: -28.8740)

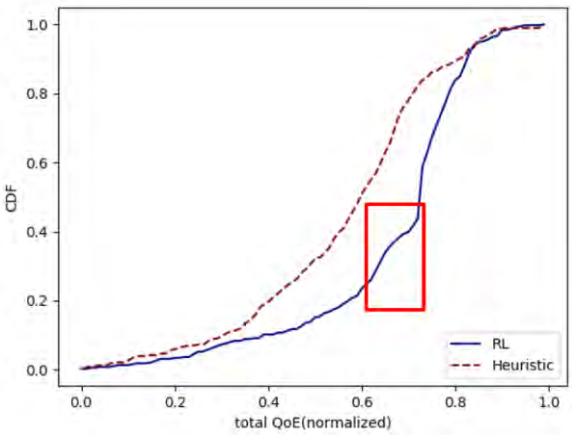


그림 12. CDF 그래프

그리고 이 결과를 누적확률분포(CDF) 그래프로 나타냈을 때, 특정 퀄리티에 대해 해당 퀄리티보다 낮게 나올 확률을 나타내는 그래프이기 때문에, 전반적으로 우측 하단에 자리잡은 강화학습 모델을 사용한 머신러닝 메소드가 상대적으로 스트리밍 퀄리티 면에서 더 우위에 있다고 판단할 수 있다. (그림 12)

5. 요약 및 고찰

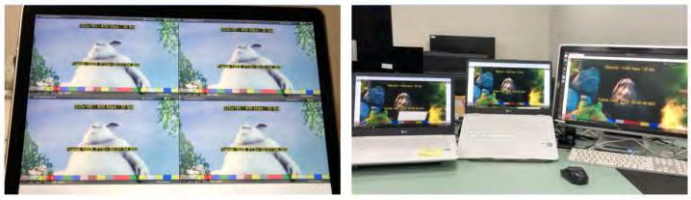


그림 13. HAS 동기화 재생 시연 장면

다중 HAS 클라이언트의 동기화 재생을 위해 각 클라이언트들을 마스터, 슬레이브로 역할을 구분하고, 서로 간의 직접적인 통신을 위해 webRTC 데이터 채널을 활용한 P2P 연결을 수립하였다. 이 연결을 통해 비트레이 통일과 클락 동기화에 필요한 데이터와 메세지를 주고 받는다. Bitrate 통일을 위해 각 클라이언트들의 ABR 컨트롤러를 통해 결정된 비트레이트 중 최소값, 또는 강화학습 모델을 통해 결정된 비트레이트로 컨센서스 후 미디어 청크를 요청하는 방식으로 구현되었다. 동일한 네트워크 환경에서 이 두 방식의 성능 평가를 진행한 결과는 강화학습 모델을 사용한 방식이 더 좋은 품질의 스트리밍을 서비스할 수 있는 것으로 나타났다. 클락 동기화는 동시 재생, 즉 동시 렌더링을 위해 필요하며 주기적으로 슬레이브가 마스터에게 시간 정보를 요청하는 것을 기본 동작으로 한다. 첫번째 방식은 슬레이브가 마스터의 시간을 요청하고 응답을 받는 라운드 트립 시간의 중간 지점을 기준으로 마스터 시간과의 차이를 오프셋으로 슬레이브의 클락을 보정한다. 두번째 방법은 이 라운드 트립 시간과 마스터의 시간을 32개의 윈도우 사이즈로 샘플링하여 이를 x와 y로 선형회귀 모델을 만든다. 이 모델을 통해 특정 시점의 슬레이브 시간에서의 마스터 시간을 추정하여 이를 통해 오프셋을 구하고 시간을 보정하는 방식이다. 성능 비교 시 마스터와의 클락 오차가 더 적은 방식은 역시 선형회귀 분석을 사용한 방식이었다.

본 연구의 결과는 위의 내용처럼 전반적인 HAS 동기화 재생 클라이언트를 구현한 것이며 네트워크 환경 등에 따라 발생하는 난제 두 가지를 휴리스틱 메소드와 머신러닝 메소드로 각각 해결하여 그 둘의 성능을 비교했으며, 머신러닝 메소드의 성능이 상대적으로 더 좋다는 사실을 나타내고 있다. 하지만 머신러닝 메소드의 성능이 절대적으로 우위에 있다는 결론을 내리기는 어렵다. 클락 동기화의 경우, 라운드 트립 딜레이가 거의 없는 루프백 환경에서는 휴리스틱 메소드의 성능이 좀 더 좋은 결과를 내는 모습을 볼 수 있었다. 물론 이러한 조건은 실제 환경에서 보기드문 경우이지만, 네트워크 환경의 상태에 따라 휴리스틱 메소드가 머신러닝 메소드에 비해 우위를 가지는 라운드 트립 딜레이 구간은 일정 부분 존재한다는 것을

짐작할 수 있다. 비트레이트 결정 부분에서도, [2]의 연구와 동일한 환경을 바탕으로 구현 및 검증을 진행했기 때문에 해당 연구와 마찬가지로 강화학습 모델의 스트리밍 품질이 더 나은 수준으로 나온 측면이 없지 않다. 하지만 [2]에서 검증에 사용된 두 개의 네트워크 트레이스의 경우, 실제 환경이나 다른 데이터셋에 비해 열악한 수준의 네트워크 환경을 재현하고 있었다. 하지만 실제 환경이나 벨기에 LTE/4G와 같은 다른 네트워크 트레이스를 통해 정성적 평가를 수행했을때, 최대치의 스트리밍 품질로 스트리밍 서비스가 가능할 정도로 네트워크 상태가 좋았으며 그 때의 각 메소드의 성능은 큰 차이는 없었지만 휴리스틱 메소드가 우위에 있었다. 즉, 네트워크 환경이 열악한 상황에서는 강화학습 모델의 비트레이트 결정 성능이 빛을 발하지만, 오히려 네트워크 상황이 좋은 상태에서는 버퍼링을 회피하고자 하는 에이전트의 성향으로 인해 최대치의 비트레이트를 선택하는 횟수가 휴리스틱 메소드에 비해 적었던 것이다. 위 두 가지 사례로 인해, 선뜻 머신러닝 메소드가 절대적 우위에 있다는 결론을 내리기는 힘들다. 그러나 HAS가 수행될 수 있는 무대는 인터넷까지 확장을 할 수 있으며, 로컬 네트워크라고 하더라도 네트워크 처리량은 언제나 변화무쌍하며 특히 동기화 재생의 경우 여러 개의 클라이언트가 정해져 있는 네트워크 대역폭을 나눠서 스트리밍 재생을 해야하는 조건이다. 이에 좀 더 낮은 수준의 네트워크 환경에서 라운드 트립 지연 시간이 길어지고, 비트레이트 결정이 더욱 힘든 상황에서 머신러닝 메소드가 상대적 우위를 가지는 것은 충분히 의미가 있는 포인트이다. 게다가 자체적으로 구현한 휴리스틱 메소드 역시 반대의 경우에 더 나은 성능을 보여주고 있기 때문에, 네트워크 처리량에 따라 적절한 메소드를 선택하여 사용하는 하이브리드 접근법까지 생각해 볼 수 있다.

이 외에 다른 제약사항이라고 할 만한 부분은 클락 동기화 이후 동시 렌더링을 구현하는 방법론에 관한 것이다. [4]에 언급하는 GStreamer의 경우, Demuxer와 직접 연결되는 파이프라인을 다루는 Backend 프레임워크이기 때문에 PCR을 동기화하면 재생되는 미디어의 PTS가 자동적으로 동기화되는 효과를 얻을 수 있다. 하지만 본 연구에서 구현한 HTML5 기반의 동기화 재생 클라이언트의 경우, 재생 중 실시간 보정을 통해 동시 렌더링을 구현하는데 어려움이 있다. 동기화된 클락을 기준으로 일정 수준 이상의 오차가 발생하게 되면 재생 중 바로 SEEK 동작을 통해 재생 시간을 조정해야하는데, 웹 어플리케이션과 같은 Frontend에서 이 동작을 요청하고 실제 수행되고 그 결과를 받게 되는데까지 100ms 전후의 지연이 발생하게 된다. 즉 실시간 재생 중 SEEK 동작을 통해 동시 렌더링을 구현하는 것보다는 최초 재생 시작 시 또는 부득이하게 특정 클라이언트에서 버퍼링이 발생하여 렌더링되는 프레임에 큰 편차가 생겼을때, 모든 클라이언트가 재생을 멈추고, 동기화된 클락을 사용하여 모두 동일 지점으로 SEEK를 수행하고 일정 시간 이후에 재생을 다같이 시작하도록 제어하는 “재생 시작 동기화” 방법으로 구현이 이루어졌다. 이렇게 Frontend에서 발생하는 타이밍적인 제약은 동기화 재생을 위한 또다른 과제이기도 하나, 이 역시 마찬가지로 Backend 미디어 프레임워크와의 하이브리드 구현을 통해 극복할 수 있을 것으로 판단된다. Fontend는 스트리밍 동기화를, Backend는 렌더링 동기화를 각각 담당하도록 하는 것이다. 물론 Backend 변경 사항이 발생하는 경우 웹엔진이나 그 하위의 미디어 프레임워크까지 함께 묶여서 일반적인 웹어플리케이션 배포 방식을 사용할 수는 없게 되지만 동기화 재생이 반드시 필요한 디지털 사이니즈 제품에 특정하여 구현하는 것은 충분히 가능한 일이다.

6. 결 론

본 연구를 진행하며 얻은 성과들을 응용하고, 부수적으로 발생한 제약 사항들을 개선하면 동기화 재생뿐만 아닌 다른 분야로까지 확장을 할 수 있을 것으로 보인다. 예를들면 HAS 재생을 위해 모든 클라이언트가 직접 미디어 청크를 Fetch하지 않고서도, webRTC를 통해 클라이언트끼리 미디어 청크를 주고 받는 P2P 스트리밍 방식을 생각해볼 수 있다. 여기서 강화학습 모델을 다시 적용하되 비트레이트 결정이 아닌 가장 좋은 네트워크 처리량을 갖는 Peer를 선택하는데 사용할 수 있을 것이다. 이런 방식은 네트워크 트래픽 자체를 줄일 수 있어 클라우드 비용을 절감할 수 있고, 더 높은 품질의 스트리밍을 서비스 할 수 있게 될 것이다. 이처럼 HAS가 여러 형태로 발전하는데 있어 본 연구가 기여할 수 있기를 바라며 동시에 개인적으로도 다음 연구로의 발판 역할을 할 것으로 기대하는 바이다.

6. 참고 문헌

[1] PLAYBACK SYNCHRONIZATION AMONG ADAPTIVE BITRATE STREAMING CLIENTS, *United States Patent Application Publication(US2020/0021868 A1)* (2020)

[2] Neural Adaptive Video Streaming with Pensieve, *Hongzi Mao, SIGCOMM '17: Proceedings of the Conference of the ACM Special Interest Group on Data Communication August(197-210)* (2017)

[3] Asynchronous Methods for Deep Reinforcement learning, *V. Mnih et al, International Conference on Machine Learning(1928-1937)* (2016)

[4] Synchronized Multi-Device Media Playback with GStreamer, *Luis de Bethencourt, Samsung Open Source Group*

[5] dash.js, Akami, https://github.com/Dash-

Industry-Forum/dash.js(2016)
[6] R. Netravali et al. 2015. Mahimahi: Accurate Record-and-Replay for HTTP. In Proceedings of USENIX ATC.

결함 치명도 분석을 통한 협업 사이버물리시스템의 안전성 확보

만주르 후세인 • 나자캇 알리 • 홍장의
충북대학교 컴퓨터과학과
{hussain | nazakatali}@selab.cbnu.ac.kr jehong@chungbuk.ac.kr

Safety Assurance in Collaborative Cyber-Physical Systems through Fault Criticality Analysis

Manzoor Hussain • Nazakat Ali • Jang-Eui Hong
Department of Computer Science, Chungbuk National University

요 약

Collaborative Cyber-Physical Systems (CCPS) are those systems containing massively interconnected and tightly coupled physical and cyber components. However, this tight coupling of components creates safety issues. Due to the tight coupling of CCPS's components, a single fault in CCPS can trigger many other faults. A fault criticality analysis approach based on composite hazard analysis technique and content relationships among hazard analysis artifacts is presented in this paper. We present a Fault Criticality Matrix (FCM) to perform fault criticality analysis at design time so that the safe development of CCPS can be ensured.

1. Introduction

Collaborative cyber-physical systems are those systems containing tightly coupled cyber and physical components, massively interconnected, and collaborates by sharing information and resources to achieve common goals. The tight coupling of cyber and physical components has many advantages, but it also creates safety issues. The safety of a single cyber-physical system (CPS) can be achieved by following the safety standards such as ISO 26262 and IEC 61508 or by applying the hazard analysis techniques. However, in collaborative CPSs the components are massively interconnected, and tightly coupled therefore it creates safety issues because a minor fault in one system can activate many other faults in other collaborating systems. Hazard analysis helps safety engineers to identify faults and it also provides information about safety guards as mentioned in [1]. Collaborative CPSs are safety-critical therefore, fault traceability is very important to determine the fault routes, their origin, to determine that the system must fulfill the safety goal, and all identified hazards were illuminated [2]. An approach was proposed by [3] that provides safety in platooning CPSs within the safeCOP project. A framework called SafeTrace was proposed by [4] that supports traceability among safety requirements, design, and safety analysis artifacts. A tool called NuDE 2.0 was developed by the authors [5] for safety analysis and verification for safety-critical systems.

In our previous work, we proposed a composite hazard analysis technique for collaborative CPSs based on content relationships among Fault Tree Analysis (FTA), Failure Mode Effect Analysis (FMEA), and Event Tree Analysis (ETA).

This paper is based on our previous work [6] in which we presented a conceptual fault traceability matrix. Now we present FCM to perform fault criticality analysis in CCPS. The focus of this work is to analyze collaborative nature CPSs, analyze the safety issues, performing composite hazard analysis, and fault criticality analysis of CCPS in detail. Figure 1 shows the working of our proposed approach.

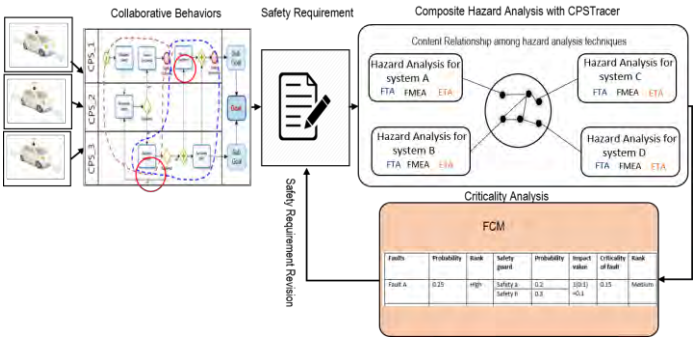


Figure 1: A Proposed Approach for Safety Assurance through Fault Criticality Analysis in Collaborative CPS.

We first analyze the collaborative behavior of CPSs and then extract safety requirements. The safety requirements are defined to reduce the risks in the system and then we perform composite hazard analysis using the defined safety requirements of collaborative CPSs. In our composite hazard analysis, we perform FTA, FMEA, and ETA for CCPS. We use our previously defined content relationships [6] such *as Inheritance, Influence, Overlap, and Supplement* relationships to establish content relationships among the ETA, FMEA, and ETA. We developed a tool called CPSTracer to perform composite hazard analysis of CCPS. After performing the composite hazard analysis of CCPS in our CPSTracer tool, the FCM algorithm

detects the content relationships among hazard analysis artifacts (FTA, FMEA, and ETA), and based on the content relationships among FTA, FMEA and ETA the algorithm generates FCM which enables to perform fault criticality analysis of CCPS.

2. Fault Criticality Matrix

This tight coupling of CCPS's components and the collaborative nature of CCPS has many advantages as well as creates safety issues. We present a matrix-based fault criticality analysis approach to perform criticality analysis of faults in CCPS. In the proposed FCM, we organize the identified faults in the first column i.e., the "*Fault*" column. In column "*P*", we organized the probability of occurrence of each fault. The third column "C" presents the criticality of fault without considering the safety guard to determine the actual fault's criticality. The ranking of the faults is based on the criticality of the fault which is organized in column "*Rank*". The safety guard column (i.e., "*SGs*") contains all safety guards provided to each fault, and the probability of occurrence of safety guard is organized in column "*P(SGs)*". In column "*IV*", the impact values for each fault are organized. Impact value is the number of other potential faults that can be activated by a particular fault. This value is determined by the number of influences, overlap, and inheritance relationships that a fault makes with other faults. One influence relationship or one inheritance relationship or one overlap relationship with other faults have a 0.1 impact value. For example, if a fault is making influence relationships with two other faults then the impact value (*IV*) will be 0.2. The column "*FC*" represents the criticality of each fault after supplying the safety guards. This column is introduced to show the effect of the safety guards on fault criticality. The "*Rank*" column ranks the criticality of faults after supplying a safety guard to show the effect of the safety guard on the criticality of each fault. Table 1 shows the template of the proposed FCM.

Table 1: Template for Fault Criticality Matrix

Fault	P	C	Rank	SGs	P(SGs)	IV	FC	Rank
k	X_k	X_k+N	High	i	S_i	N	C	Medium
k	X_k	X_k+N	Medium	i	S_i	N	C	Low
k	X_k	X_k+N	Low	i	S_i	N	C	Negligible

P:Probability, C: Criticality, SGs: Safety Guards, P(SGs): Probability of Safety Guards, IV: Impact Value, FC: Fault Criticality(after safety guard)

The ranking criteria for determining the criticality of faults are shown in Table 2.

Table 2: Ranking Criteria of the Criticality of faults.

Fault Criticality Value	Ranking
C<=0.0	No effect
0.0 ~0.005	Negligible
0.005 ~ 0.01	Low
0.01 ~ 0.15	Medium
0.15 ~ 0.4	High
0.4 ~0.6	Very high
0.6 ~ 1.0	Catastrophic

3. Fault Criticality Calculation

The criticality of fault is represented by FC. Let us consider S (s_1, s_2, s_3…S_i) is the probability of safety guards where i=1,2,3... and X (x_1, x_2, x_3…X_k) is the probability of faults where k=1,2,3, … The safety guard S_i is supplied to mitigate the impact of fault X_k from the systems where i is the number of safety guards and k is the number of faults. The impact value of fault is represented by N. The value N is defined to be 0.1 if the fault has an impact on only one fault. If the faults impact multiple faults, then each impact would have a 0.1 value and the final impact value will be the sum of all values. The probability of occurrence of each fault and safety guard is taken from the respective hazard analysis technique and the impact value is taken from the number of influences, overlap, and inheritance relationship that a fault is making with other faults. Now the value of fault criticality is determined by using equation (1).

$$FC = (X_k + N)\forall k - \sum_{i \in I} Si \qquad (1)$$

The minus or zero value of criticality of any fault indicates that the faults do not affect the system as the fault's criticality is mitigated by supplying a safety guard. Reduction in criticality indicates that the criticality is reduced however, for the failure-free system, all potential faults must be mitigated.

4. Conclusion

The tight coupling of CCPS components creates safety issues. A single fault in CCPS may trigger many other faults due to interconnected components and their collaboration. Therefore, it is a challenging task to determine which faults are more critical to the system′s safety. We proposed a matrix-based fault criticality analysis approach called FCM to perform the fault criticality analysis in CCPS.

We are working on designing a deep neural network-based safety verification approach for collaborative CPSs at run time.

5. Acknowledgment

This work is supported by the Korea National Research Foundation of the Ministry of ICT and Science (NRF-2017M3C4A7066479, NRF-2020RIA2C1007571).

6. References

[1] Clifton A. Ericsson, Hazard Analysis Techniques for System Safety, 2016.

[2] H Cleland, S Rayadurgam, P Mäder, and W Schafer, "Software and Systems Traceability for Safety-Critical Projects", Technique Report from Dagstuhl Seminar, 2015.

[3] S. Medawar, D. Scholle, and I. Sljivo, "Cooperative safety critical CPS platooning in SafeCOP", Proc. 6th Medit. Conf. Embedded Comput. (MECO), pp. 1-5, Jun. 2017.

[4] A. Y.-Z. Ou, M. Rahmaniheris, Y. Jiang, L. Sha, Z. Fu and S. Ren, "Safetrace: A safety-driven requirement traceability framework on device interaction hazards for MD PnP", Proc. 33rd Annu. ACM Symp. Appl. Comput. (SAC), pp. 1282-1291, 2018.

[5] E.-S. Kim, D. Lee, J. Sejin, J. Yoo, J. Choi and J.-S. Lee, "NuDE 2.0: A formal method-based software development verification and safety analysis environment for digital I&Cs in NPPs", J. Comput. Sci. Eng., vol. 11, no. 1, pp. 9-23, 2017.

[6] Ali, N., Hong, J.: Failure detection and prevention for cyber-physical systems using ontology-based knowledge base. Computers 7(4), 68 (2018)

교차 버전 결함 예측을 위한 베이지안 최적화 프레임워크

최정환[1], 류덕산[2]
[1]연세대학교 인공지능학과, [2]전북대학교 소프트웨어공학과
jeognwhan.choi@yonsei.ac.kr, duksan.ryu@jbnu.ac.kr

Bayesian Optimization Framework for Cross-Version Defect Prediction

Jeongwhan Choi[1], Duksan Ryu[2]
[1]Yonsei University, [2]Jeonbuk National University

요 약

최근 소프트웨어 결함 예측 연구는 교차 프로젝트 간의 결함 예측 뿐만 아니라 교차 버전 프로젝트 간의 결함 예측 또한 이루어지고 있다. 종래의 교차 버전 결함 예측 연구들은 WP(Within-Project)로 가정한다. 하지만, CV(Cross-Version) 환경에서는 프로젝트 버전 간의 분포 차이 또한 중요하다. 본 연구에서는 다른 버전 간의 분포 차이까지 고려하는 자동화된 베이지안 최적화 프레임워크를 제안한다. 이를 통해 분포 차이에 따라 전이 학습(Transfer Learning) 수행 여부를 자동으로 선택하여 준다. 해당 프레임워크는 버전 간의 분포 차이, 전이 학습과 분류기(Classifier)의 하이퍼파라미터를 최적화하는 기법이다. 실험을 통해 전이 학습 수행 여부를 분포차 기준으로 자동으로 선택하는 방법이 효과적이라는 것을 알 수 있다. 그리고 최적화를 이용하는 것이 성능 향상에 효과가 있으며 이러한 결과 소프트웨어 인스펙션 노력을 감소할 수 있다는 것을 확인할 수 있다. 이를 통해 교차 버전 프로젝트 환경에서 신규 버전 프로젝트에 대하여 효과적인 품질 보증 활동 수행을 지원할 것으로 기대된다.

1. 서론

소프트웨어 결함 예측은 소프트웨어 데이터를 기계학습 기법에 적용하여 결함이 발생하기 쉬운 소프트웨어 모듈들을 효과적으로 찾는다. 이를 통해 소프트웨어의 품질 보장과 코드 리뷰 혹은 테스팅에 대한 리소스 할당을 효과적으로 할 수 있다.

신규 소프트웨어 프로젝트의 경우 데이터가 부족하기 때문에 다른 프로젝트의 충분한 데이터를 사용한다. 이러한 CP(Cross-Project) 환경에서는 프로젝트 간의 분포 차이를 줄여주는 전이 학습(Transfer Learning)이 주로 사용된다.

최근에는 프로젝트의 이전 버전들을 사용하여 신규 버전 프로젝트의 결함을 예측하는 연구가 대두된다. 이를 CV(Cross-Version) 환경이라 한다. CV 환경에서는 다른 버전 간의 분포 차이가 있을 수 있지만, 종래의 연구에서는 버전 간의 분포 차이를 고려하지 않는다.

본 연구에서는 다른 버전 간의 분포 차이까지 고려하는 CVDP(Cross-Version Defect Prediction)용 베이지안 최적화 프레임워크를 제안한다. 이를 통해 분포 차이에 따라 전이 학습 수행 여부를 자동으로 선택하여 준다. 해당 프레임워크는 버전 간의 분포 차이, 전이 학습과 분류기(Classifier)들을 모두 베이지안 최적화(Bayesian optimization)를 통해 성능을 높인다.

본 연구에서 제안하는 CVDP를 위한 베이지안 최적화 프레임워크는 모든 데이터셋에 대해 성능 향상 효과가 있고 소프트웨어 인스펙션 노력 비용을 감소할 수 있음을 보여준다.

2. 관련 연구

CVDP에 대한 최근 연구로는 S.Amasaki의 연구가 있다[1][2]. 이 연구는 전이 학습이 CP 환경이 아닌 CV 환경에서도 적용되는지를 연구 하였다. 하지만, 우리의 연구와는 다르게 이 연구는 버전 간의 분포 차이에 대한 분석이 없으며 이를 고려하지 않았다.

다른 CVDP 연구들에 대해서도 버전 간의 분포 차이에 대한 고려가 이루어지지 않다. 특히 X.Yang et al. 의 연구에서는 버전 간의 분포를 확인하지 않은 채, CV 환경이 WP 환경과 유사하다고 가정한다[3].

3. 연구 방법

본 연구가 제안하는 베이지안 최적화 프레임워크는 CVDP 단계가 포함되어 있다. CVDP 단계에서 최신 버전을 타겟 버전 프로젝트, 이전 버전을 소스 버전

프로젝트로 설정한다. 두 버전에 대한 분포 차이를 KS(Kolmogorov-Smirnov) 테스트를 사용하여 확인한다. 분포가 다른 칼럼이 우리가 설정한 임계값보다 크면 전이 학습을 진행하고 작으면 진행하지 않는다.

선택적 전이 학습을 통해 타겟 버전 프로젝트의 분포와 유사한 인스턴스들로 필터링된 데이터들이 CVDP 모델의 트레이닝셋으로 사용된다. 타겟 버전에 대한 예측 성능 평가를 AUC(Area Under Curve)로 진행하고 소프트웨어 인스펙션 감소 비율에 대해서 FIR(File Inspection Reduction)로 분석한다. FIR은 랜덤 선택과 비교하여 예측 모델을 사용하여 인스펙션 할 때, 동일한 PD(Probability of Detection)를 달성하기 위해 줄어든 인스펙션 할 파일 수의 비율이다. CVDP 모델의 분류기로는 BRF(Balanced Random Forest)를 사용하고 전이학습은 TCA(Transfer Component Analysis)와 NNFilter(Nearest Neighbor Filter), DSNF(Data Selection and Nearest Neighbor Filter) 를 사용한다. 베이지안 최적화 프레임워크의 최적화 단계 부분에서는 하이퍼파라미터들의 탐색 공간을 정의하고 목적함수에 따른 최적의 값을 찾는다.

4. 실험 설계

데이터셋은 버전 정보가 있는 MORPH 데이터셋을 사용한다[4]. 그리고 실험 설계를 위한 연구질문과 가설은 아래와 같다.

RQ1. CV 환경에서 버전 간 프로젝트들의 분포가 다른가?

RQ2. CV 환경에서 분포 차이에 따른 선택적 전이 학습을 적용할 경우 성능 향상이 있는가?

RQ3. 베이지안 하이퍼파라미터 최적화를 적용한 프레임워크의 성능 향상 효과가 얼마나 있는가?

RQ4. CVDP 모델에서 우리의 접근 방법이 최적화를 하지 않은 기본 모델과 대비하여 소프트웨어 인스펙션 노력을 감소시키는가?

5. 실험 결과

RQ1. 프로젝트의 버전 간 데이터 분포 차이

모든 버전 조합 쌍에서 평균 2.37개의 피쳐의 분포가 다르다는 것을 확인할 수 있다. 이는 CV 환경에서 프로젝트마다 버전간 분포 차이가 다르다는 것을 의미하며 분포가 다른 피쳐의 개수는 [0, 9]의 범위를 가진다. 중간값인 5를 RQ3에서 임계값 범위의 최대값으로 설정하여 사용한다.

RQ2. 분포 차이에 따른 선택적 전이 학습의 효과

분포 차이에 따른 선택적 전이 학습의 성능 향상에 대해서 실험한 결과이며 효과크기는 모두 매우 크기에 선택적 전이학습을 적용하는 것의 효과가 크다.

RQ3. 베이지안 최적화 프레임워크의 효과

베이지안 최적화를 적용한 프레임워크가 최적화를 하지 않은 모델보다 성능 효과가 있는지 실험한 결과이다. 전이 학습들의 실험 결과에 대한 효과크기는 크기에 본 연구의 최적화 프레임워크가 효과적이다.

RQ4. Effort-aware 예측 분석

Effort-aware 예측 분석 관점에서 본 연구에서 제안하는 프레임워크를 FIR(File Inspection Reduction Ratio) 기준으로 평가한다. 전이 학습들의 FIR 값에 대한 Cohen's D 효과 크기는 TCA와 NNFilter의 경우에 중간 크기이다. 본 논문이 제안하는 CVDP를 위한 베이지안 최적화 프레임워크는 소프트웨어 인스펙션 노력 비용을 감소시킬 수 있음을 확인하였다.

6. 결론

본 연구는 CP 환경에서의 전이 학습이 CV 환경에서 어떻게 적용되어야 할지 보인다. CVDP의 관련연구들은 분포 차이를 고려하지 않은 채 CV 환경을 다루지만 우리가 제안하는 프레임워크는 분포 차이를 고려하고 전이 학습 수행 여부를 선택한다.

본 연구는 다른 다양한 분류기와 전이 학습 기법들을 본 연구의 베이지안 최적화 프레임워크에 확장하여 적용 가능하다. 향후 연구로 이 프레임워크를 확장할 계획이며 베이지안 최적화 뿐만 아니라 다른 최적화 알고리즘 또한 적용할 것이다.

Acknowledgement

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

참고 문헌

[1] S. Amasaki, "Cross-version defect prediction: use historical data, cross-project data, or both?," *Empir. Softw. Eng.*, pp. 1573-1595, Mar. 2020.

[2] S. Amasaki, "Cross-Version Defect Prediction using Cross-Project Defect Prediction Approaches," in *Proceedings of International Conference on Predictive Models and Data Analytics in Software Engineering*, 2018.

[3] X. Yang and W. Wen, "Ridge and Lasso Regression Models for Cross-Version Defect Prediction," *IEEE Trans. Reliab.*, pp. 885-896, Sep. 2018.

[4] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering - PROMISE '10*, 2010, p. 1.

군집주행을 위한 ISO/PAS 21448 표준의 확장

김 영 재○, 홍 장 의
충북대학교 컴퓨터과학과
youngjae@cbnu.ac.kr, jehong@cbnu.ac.kr

The Extension of ISO/PAS 21448 Standard for Platoon Driving

Youngjae Kim, and Jang-Eui Hong
Dept. of Computer Science, Chungbuk National University, South Korea

요 약

매우 복잡한 도로 환경으로 인하여 안전하고 신뢰성 있는 자율주행차량을 개발하는 것은 매우 어려운 일이다. 이러한 자율주행차량의 안전을 확보하기 위한 연구가 다방면으로 수행되고 있다. 지난 2019년 제정된 ISO/PAS 21448 표준은 이러한 자율주행차량의 의도된 기능에 의하여 발생 가능한 위험을 줄이고자 하는 목적을 가진다. 한편, 여러 대의 자율주행차량이 하나의 군집을 형성하여 주행하는 군집주행 기술도 개발되고 있다. V2V 통신을 통하여 차량 간의 정보를 교환하며 주행하는 군집주행 기술은 기민한 반응으로 차량 간의 매우 좁은 안전거리를 가능하게 함으로써 도로의 통행처리량 증대, 에너지 소비의 감소, 공해물질 배출의 감소 등 다양한 장점을 지닌다. 그러나 군집주행은 여러 대의 자율주행차량이 상호작용을 하므로 복잡성이 더욱 증가하여 그 안전을 확보하기가 더 어렵다. 이를 위해 본 연구에서는 ISO/PAS 21448 표준을 이러한 군집주행 기술에 적절히 적용할 수 있도록 수정 및 확대하는 방안을 제시한다. 제시된 방안을 군집주행의 차선변경 기능에 적용하는 사례 연구를 통하여 제안하는 확장 방안의 유용성을 확인하였다.

1. 서 론

1885년 세계 최초의 가솔린 자동차가 등장한 이후, 자동차는 화석연료를 사용하는 내연 기관을 사용하여 사람이 운전하는 것이 당연한 패러다임으로 우리 생활에 자리 잡았다. 하지만 급격한 기술의 발달로 화석연료 대신 전기자동차, 수소연료전지 자동차와 같이 다양한 에너지원으로 주행하는 차량이 등장하였고, 이제는 사람이 운전한다는 패러다임마저 바꿀 자율주행차량의 등장을 목전에 두고 있다.

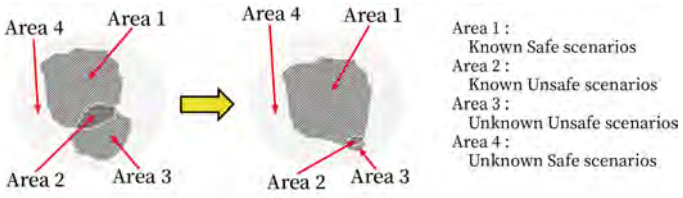
하지만 자율주행차량의 완성을 가로막는 것은 차량을 주행시키는 기계적 기술이 아니라 충분한 안전을 확보하는 것의 어려움에 있다. 자율주행차량은 대표적인 안전필수 시스템(Safety-Critical System)의 사례로서 사고 발생 시 탑승객과 주변 사람들에게 부상을 입히거나, 심지어 생명을 앗아갈 수도 있는 위험성을 가진다. 그러나 다양한 변수를 고려해야 하는 도로의 복잡한 환경은 자율주행차량의 충분한 안전을 확보하는 것을 어렵게 만들고 있다.

그럼에도 불구하고 교통사고 발생원인의 90% 이상이 운전자의 운전미숙, 운전자의 피로 및 과실 등으로 인해 발생하기 때문에 미국 도로교통안전청(National Highway Traffic Safety Administration, NHTSA)에서는 자율주행차량의 상용화가 운전자의 문제로 발생하는 교통사고의 94%를 예방하고, 이로 인해 발생하는 사회적, 경제적 이익이 미국에서만 연간 1900억 달러에 달할 것으로 예측[1]하고 있다.

따라서 안전한 자율주행차량을 개발하기 위한 다양한 노력이 전 세계적으로 진행되고 있다. 특히 차량에 도입되는 전장 장비들의 기능 고장을 폭넓게 다루기 위하여 지난 2011년 처음 발표된 ISO 26262 표준[2]은 자동차 산업 전반에 폭넓게 활용되고 있다. 2018년에 한차례 개정되었고 'Road Vehicles – Functional Safety'라 명명된 ISO 26262 표준은, V모델 기반의 표준 프로세스를 통하여 모든 생명주기에서 발생 가능한 고장 상황에 대하여 차량의 안전을 확보하기 위한 노력을 담고 있다.

하지만 자율주행차량에서 기능의 고장 없이도 발생하는 사고가 보고되고 있다. 지난 2016년, 테슬라의 모델 S 자율주행차량은 대형 트럭에 흰색으로 도장된 색상을 밝은 하늘의 색으로 혼동하여 충돌 사고를 발생시켰고, 운전자가 사망하는 사고가 발생하였다. 이렇듯 차량의 기능 고장 없이도 발생 가능한 사고를 방지하기 위한 새로운 표준의 필요성이 대두되었고, ISO 26262 표준을 보완할 수 있는 ISO/PAS 21448 표준[3]이 지난 2019년 제정되었다.

ISO/PAS 21448 표준은 강한 빛, 음영 지대 등으로 인해 발생하는 센서의 성능 제약, 예측 가능한 운전자의 실수, 여러 주변 환경과의 상호작용 등으로 인해 기능의 고장 없이 발생하는 위험을 줄이고자 제정된 표준으로, 〈그림 1〉과 같이 알려지지 않은 위험(Unknown Unsafe) 시나리오와 알려진 위험(Known Unsafe) 시나리오를 알려진 안전(Known Safe) 시나리오로 전



〈그림 1〉 ISO/PAS 21448 표준의 목적

환함으로써 자율주행차량의 안전한 주행을 확보하는데 그 목적을 둔다.

한편, 자율주행차량의 발전과 함께 군집주행에 대한 연구가 가속화되고 있다. 군집주행은 차량 서로 간의 V2V(Vehicle to Vehicle) 통신을 통하여 정보를 주고받는 여러 대의 자율주행차량이 하나의 군집을 형성하여 주행하는 기술이다. 군집주행은 차량 간의 기민한 반응을 통해 사람이 달성하기 힘든 좁은 안전거리를 갖는 주행을 가능하게 한다. 이렇게 형성된 좁은 차량 간의 간격은 도로의 통행처리량을 증대시키고, 공기저항의 감소를 일으킴으로써 에너지 소비의 감소, 공해물질 배출의 감소 등 다양한 장점[4]을 얻는다.

하지만 이러한 군집주행은 여러 대의 자율주행차량이 상호작용을 하는 만큼 시스템이 더 큰 복잡성을 갖게 되고, 안전에 대한 고려사항이 더욱 증가하므로 그 안전을 확보하기가 한 대의 자율주행차량에 비해 더 어렵다.

본 논문에서는 이와 같은 군집주행에서 발생 가능한 위험을 최소화하고 안전한 군집주행이 가능한 기술을 확보하기 위하여 ISO/PAS 21448 표준의 범위를 군집주행으로 확장하기 위한 방법을 제시한다. 이를 위하여 ISO/PAS 21448 표준에서 다루는 상황을 개별 자율주행차량, 군집 내부에서의 상호 작용, 군집 외부와의 상호작용 등 세 가지의 범주로 분류하고, 각 범주에서 필요로 하는 ISO/PAS 21448 표준의 고려사항에 대하여 연구하였다. 이와 같은 연구를 군집주행의 차선 변경 기능에 대한 사례 연구를 통해 진행하였다. 이와 같은 사례 연구를 통해 군집주행의 안전을 확보하기 위하여 제시된 방법을 적용하였다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 ISO/PAS 21448 표준과 관련 연구를 알아보고, 3장에서는 본 논문에서 제시하는 군집주행을 위한 ISO/PAS 21448 표준의 확장을 설명한다. 4장에서는 사례 연구로서 제시된 방법을 통하여 군집주행의 차선 변경 기능에 적용하는 방법을 보인다. 5장에서는 결론 및 향후 연구에 대해 이야기하고자 한다.

2. 관련 연구

2.1 ISO/PAS 21448 표준

2019년, SOTIF(Safety Of The Intended Functionality)라 명명된 ISO/PAS 21448 표준이 발표되었다. '의도된 기능의 안전'이라는 의미의 SOTIF는 주행 중인 모든 차량의 컴포넌트들이 의도된 기능대로 작동하는 상황에서도 발생이 가능한 위험을 축소하여 충분히 안전한 상태를 확보하는 것을 뜻한다. ISO/PAS 21448 표준에서 다루는 위험의 요인은 다음과 같이 크게 세 가지로 분류할 수 있다.

(1) 차량의 성능 제약 또는 불충분한 상황 인식
(2) 합리적으로 예측 가능한 사용자의 오용 및 잘못된 HMI(Human Machine Interface)
(3) 다른 사용자, 주행에 적용되는 인프라, 날씨 등의 환경 조건, 전자파 간섭 등 차량 주변으로부터의 영향

ISO/PAS 21448 표준에서는 위와 같은 요인으로부터 발생 가능한 위험으로부터 SOTIF 안전을 달성하기 위한 방법으로 SOTIF 활동을 제시한다. SOTIF 활동을 지속적으로 충분히 수행함으로써 위와 같은 상황에 대한 위험을 발견하고, 발견된 위험을 축소시켜 안전을 최대한 확보할 수 있다.

ISO/PAS 21448 표준에서 제시하는 SOTIF 활동의 흐름도는 〈그림 2〉와 같다. 각 단계별 활동을 통해 지속적으로 기능에서 발생 가능한 위험을 식별하고 그 위험성을 평가한다. 높은 위험성으로 인해 대책이 필요한 경우 위험을 감소시키기 위해 기능을 수정하는 활동을 반복한다. 모든 위험이 충분히 허용 가능한 낮은 수준에 도달하였을 경우, 최종적으로 SOTIF 활동을 종료하고 SOTIF 안전을 달성하여 해당 기능에 대한 위험성 분석을 마치게 된다. 흐름도를 통해 확인할 수 있는 구체적인 SOTIF 활동의 단계는 〈표 1〉과 같다.

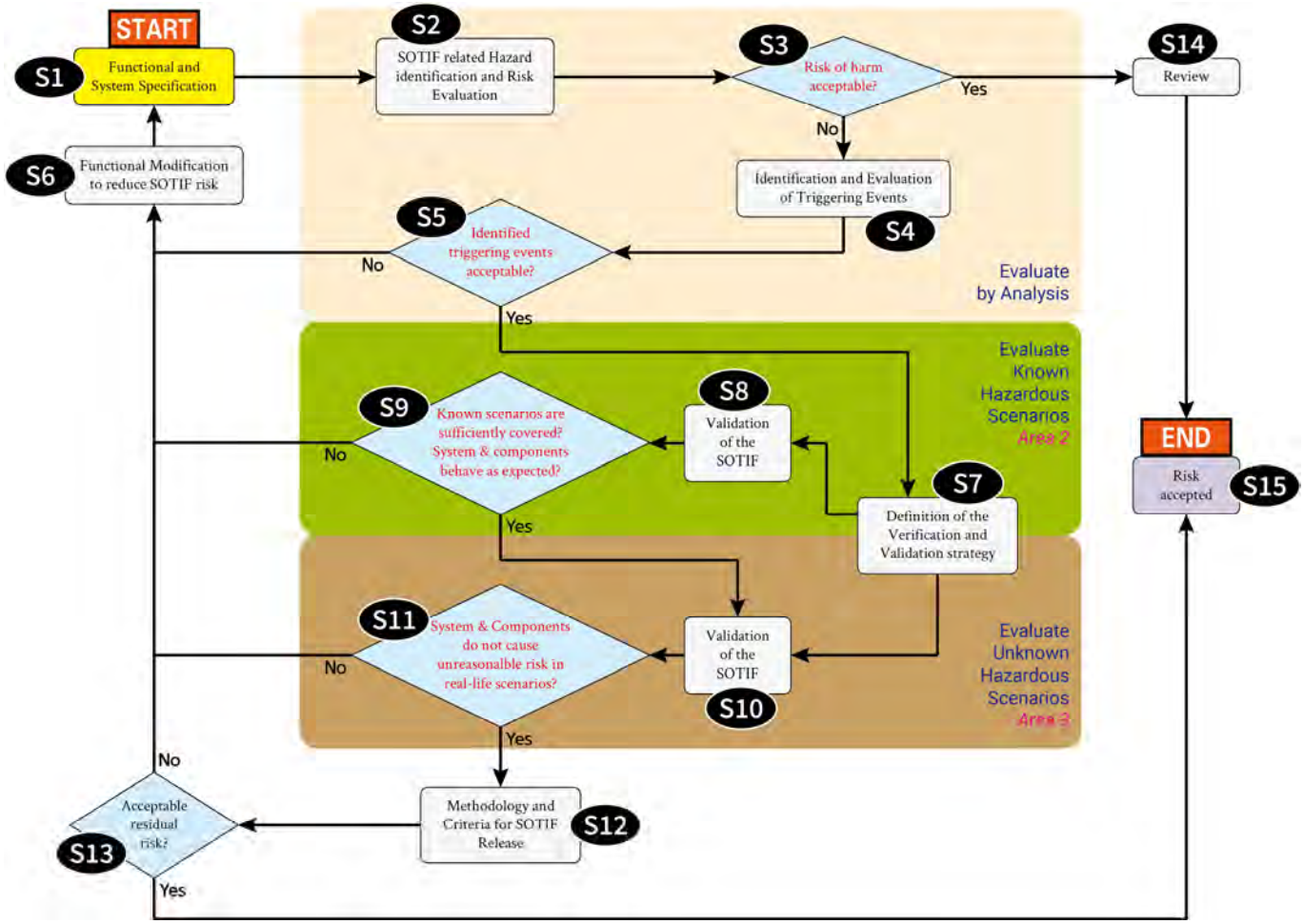
2.2 군집주행

로드 트레인(Road Train)이라고도 불리는 군집 주행은 여러 대의 자율주행차량이 좁은 차량 간 간격을 갖는 하나의 군집을 이루어 주행하는 기술이다. 지난 2009년, 유럽에서 군집주행 기술을 실증하기 위해 SARTRE(SAfe Road Trains for the Environment) 프로젝트[5]가 시작되었고, 사람이 주행하는 선두 차량을 세 대의 후속 차량이 완전한 자율주행으로 따라가는데 성공하였다.

이후 유럽을 중심으로 세계 각국에서 지속적으로 군집주행과 관련된 기술의 개발이 진행되고 있으며, 국내에서도 현대자동차가 2019년, 군집주행의 도로 테스트에 성공[6]하였다.

군집주행을 실현하기 위해서는 다양한 기술이 필요하다. ACC(Adaptive Cruise Control) 기술[7-8]은 각종 센서를 통해 앞 차량의 정보를 수집하여 상황을 판단하고, 자동으로 속도와 조향을 조정하며 앞 차량을 일정한 간격으로 따라가는 기술이다.

단순히 센서를 통하여 주변 상황을 파악하는 ACC 기술로부터 발전된 CACC(Cooperative Apaptive Cruise Control) 기술[7-9]은 주변의 여러 차량과 도로 곳곳의 인프라로부터



〈그림 2〉 SOTIF 활동의 흐름도[3]

V2X(Vehicle to Everything) 통신을 통하여 정보를 획득함으로써 더욱 기민하고 효율적인 주행을 가능하게 한다.

이러한 V2X 통신이 가능하도록 하는 차량 간 통신에 대한 기술도 발전하고 있다. 1999년, 미국 연방통신위원회(Federal Communications Commission, FCC)는 5.9GHz 대역의 주파수를 차량 간 통신을 위하여 할당하였다. 이어서 각국이 같은 주파수 대역을 차량 간 통신에 할당하였다.

2010년, 미국 전기전자학회(IEEE)에서는 차량 간 통신을 위하여 WAVE(Wireless Access in Vehicular Environments)라 불리는 IEEE 802.11 표준[10]을 제정하였다. 하지만 최근 와이파이 기술을 기반으로 하는 WAVE 표준의 단점이 부각되었다. 2017년, 3GPP(3rd Generation Partnership Project)에서 LTE 기술 기반의 C-V2X 통신 표준[11]을 발표하였다. 이 표준은 현재 5G 기술의 도입으로 훨씬 빠른 응답 성능을 제공하며 차세대 V2X 통신의 표준으로 떠오르고 있다.

이런 기술을 바탕으로 주행하는 군집은 효율적인 주행을 위하여 다양한 군집전략을 수행[12-13]하며 동적으로 그 크기와 구성을 변화시킨다. 대표적인 군집 전략에는 두 군집을 하나의 군집으로 합치는 합류(Merge) 전략, 하나의 군집을 두 개 이상의 군집으로 분리하는 분리(Split) 전략, 주행 중인 군집으로부터 빠져나가는 이탈(Leave) 전략 등이 있다.

2.3 문헌 연구

최근 자율주행차량 기술의 성숙에 힘입어 군집주행 기술과 그 안전을 확보하고자 하는 연구가 최근 활발히 수행되고 있다.

O. Kirovskii 등은 차량의 안전을 확보하기 위한 두 표준 ISO 26262와 ISO/PAS 21448의 통합을 통해 요구사항의 추적성을 확보하는 방법과 안전 사례의 생성에 대한 연구[14]를 수행하였다.

S. Santini 등은 군집주행에서 군집 리더를 두지 않고, 군집을 구성 차량들의 합의를 기반으로 군집 전략을 수행하는 방법에 대하여 연구[15]하였다. 이 방법은 더 나은 군집의 안정성과 안전성을 이끌어낼 수 있다.

J. Ploeg 등은 군집에 적용 가능한 계층적 제어 구조를 제시[16]하였다. 이 연구에 의하여 고안된 제어 구조는 SOTIF 관련 위험성을 감소시킬 수 있다.

H. E. Monkhouse 등은 자율주행 상황에서 차량에 영향을 주는 사람의 행동에 대하여 연구[17]하였다. 이 연구에서는 사

람에 의해 발생 가능한 잠재적인 위험을 인지할 수 있는 방법을 제시한다.

R. Horowitz 등은 자동화고속도로시스템(Automated Highway System, AHS)에서 수행 가능한 효과적인 군집의 차선변경 전략을 연구[18]하였다.

또한 H. C. Hsu는 군집의 차선변경 전략을 설계하고 차선변경 과정에서 발생하는 군집의 운동역학적인 영향을 분석[19]하였다.

3. ISO/PAS 21448 표준의 확장

본 논문에서는 단일 차량에 적용되는 ISO/PAS 21448 표준을 군집주행으로 확장하기 위한 방법을 제시한다. 먼저 본 논문의 연구에서는 다음과 같이 군집주행 중인 차량에 대하여 그 적용 범위에 따라 표준의 적용 범주를 세 가지로 분류하였다.

(A) 개별 자율주행차량에 대한 적용

(B) 군집 내부에서의 상호작용에 대한 적용

(C) 군집 외부와의 상호작용에 대한 적용

〈표 1〉 SOTIF 활동의 단계별 상세

단계 (Step)	SOTIF 활동 이름	SOTIF 활동 내용
S1	기능 및 시스템 명세	차량에 탑재된 모든 기능은 SOTIF 활동이 수행되어야 한다. 기능 및 시스템 명세로부터 SOTIF 활동의 대상이 될 기능을 선택한다.
S2	SOTIF 관련 위험의 식별 및 평가	Step 1에서 선택된 기능으로부터 발생할 수 있는 SOTIF 관련 위험 시나리오들을 식별하고, 그 위험성을 평가한다.
S3	식별된 위험이 허용 가능한 수준의 위험성을 갖는가?	Step 2에서 평가한 위험성이 충분히 허용 가능한 수준의 작은 위험인지, 아니면 대책을 필요로 할 정도의 위험성을 가지는지 판정한다.
S4	트리거링 이벤트의 식별 및 평가	Step 3에서 판정된 위험 시나리오가 어떤 상황에서 발생하는지 트리거링 이벤트를 식별하고, 발생 확률을 평가한다.
S5	식별된 트리거링 이벤트가 허용 가능한 빈도인가?	Step 4에서 평가된 트리거링 이벤트의 발생 확률이 충분히 허용 가능한 수준의 낮은 빈도인지, 아니면 대책을 필요로 할 정도의 빈도로 발생하는지 판정한다.
S6	SOTIF 위험을 감소시키기 위한 기능 수정	각 단계에서 파악된 위험성을 감소시키기 위한 안전대책을 세운다. 수립된 안전대책에 의해 시스템의 기능을 수정 및 보완하고 이를 기능 및 시스템 명세에 반영한다.
S7	검사 및 검증(Verification & Validation) 방안 정의	알려진 시나리오에 대해서는 다양한 상황의 해당 시나리오에서 시스템이 어떻게 동작하는 지를 확인하고, 알려지지 않은 시나리오에 대해서는 충분한 실제 상황에서의 주행 테스트를 수행하여 SOTIF 관련 위험이 충분히 축소되었음을 확인하기 위한 검사 및 검증 방안을 수립한다.
S8	SOTIF 검증 (Area 2)	Step 7에서 수립된 방안에 의하여 알려진 위험 시나리오(Area 2)에서의 검증을 수행한다.
S9	알려진 시나리오가 충분히 다루어졌는가? 또 시스템과 구성 요소들이 기대대로 작동하는가?	다양한 알려진 시나리오에서 검증이 수행되었는지, 또 해당 시나리오에서 트리거링 이벤트에 의하여 발생한 위험이 수립된 안전대책에 의하여 적절히 통제되어 시스템과 구성 요소들이 기대대로 작동하는지를 판정한다.
S10	SOTIF 검증 (Area 3)	Step 7에서 수립된 방안에 의하여 알려지지 않은 위험 시나리오(Area 3)에서의 검증을 수행한다.
S11	시스템과 구성 요소들이 실제 상황에서 비합리적인 위험을 발생시키지 않았는가?	실제 상황에서의 주행 테스트를 통해 미처 확인하지 못한 요인에 의해 비합리적인 위험 상황이 발생하지는 않았는지를 판정한다.
S12	SOTIF 달성을 위한 방법과 기준	Step 9, Step 11에서 수행된 검증 결과가 SOTIF 달성 기준을 만족하는지 평가한다.
S13	잔류 위험이 허용 가능한 수준의 위험성을 갖는가?	처리되지 않고 남아있는 위험들이 전부 허용 가능한 수준의 낮은 위험성을 갖는지 판정한다.
S14	리뷰	Step 3에서 충분히 허용 가능한 수준의 낮은 위험성을 가진다고 판정한 위험 시나리오를 재검토한다.
S15	허용된 위험	충분히 허용 가능한 수준의 낮은 위험성만을 갖는 위험 시나리오만을 남겨두고, 주어진 기능에 대한 모든 SOTIF 활동을 종료한다.

(A)는 군집을 구성하는 각각의 자율주행차량들에게 적용되어야 할 부분을, (B)는 군집을 구성하는 차량들끼리의 상호작용 상황에서 발생 가능한 부분을, (C)는 군집 외부의 다른 차량 및 환경적 요소 등에 고려되어야 하는 부분에 대한 SOTIF 활동을 의미한다. ISO/PAS 21448 표준에는 본 논문의 2.1절에서 소개한 SOTIF 활동의 각 절차에 대한 구체적인 수행 방법과 세부적인 고려 사항들이 나열되어 있다. 본 연구에서는 이러한 수행 방법과 고려 사항들을 군집주행에 적합하도록 위와 같이 분류된 범주에 따라 표준의 내용을 수정 및 확장하였다.

3.1 의도된 기능의 위험 식별 및 평가 단계

ISO/PAS 21448 표준의 6절에는 의도된 기능의 위험 식별 및 평가 단계에 대한 구체적인 활동 수행 방법이 설명되어있다.

본 연구에서는 위의 세 가지 분류에 따라 다음과 같이 위험 식별 및 평가 단계를 진행할 것을 제안한다.

3.1.1 (A) 개별 자율주행차량에 대한 적용

개별 자율주행차량은 다른 차량들과 군집주행이라는 협업을 하는 과정에서도 자신의 차량에 요구되는 기능을 잘 만족해야 한다. 또한 군집의 리더가 요구하는 주행 요구사항과 군집전략의 수행을 이행하는 것은 개별 차량의 책임이다. 따라서 개별 차량에는 충분하고 다양한 안전에 대한 고려를 필요로 하므로, 기존의 모든 활동을 수행할 것을 필요로 한다.

그러나 기존의 ISO/PAS 21448 표준은 차량의 V2X 통신 상황으로 인해 발생 가능한 위험에 대한 고려가 부족하다. 차량에 탑재된 통신 장비의 고장으로 인한 위험은 ISO 26262 표준에서 다루고 있다. 또한 차량 통신 장비의 성능 부족으로 인한 일시적인 통신 중단, 전파의 간섭, 장비의 열화와 같은 부분은 이미 ISO/PAS 21448 표준에서 다루는 범위이다. 하지만 악의적 공격을 이용하여 발생 가능한 위험은 ISO/PAS 21448 표준이 군집주행 등 통신이 가능한 차량에서 새롭게 고려되어야 할 요소이다.

차량의 V2X 통신을 통한 악성 공격은 사고 발생이 큰 피해로 이어지는 자율주행차량의 특성상 매우 큰 위험 요소가 된다. 보안 취약점을 이용한 보안 공격은 ISO/PAS 21448 표준에서 다루는 범위가 아니다. 이러한 사이버보안 공격을 방지하기 위한 ISO/SAE 21434[20]표준이 현재 개발 중에 있다. 그러나 Z. Khattak의 연구[21]는 보안 공격 없이도 발생 가능한 군집 내부에서의 악의적 공격의 위험성을 보여준다. 예를 들면 군집 리더가 자신의 군집 내부의 공격 대상에게 급정지를 지시하거나, 또는 바로 옆에 지나가는 차량이 있는 상황에도 차선 변경을 지시하는 경우 인명 손실을 야기할 수 있는 위험 요인이 된다. 이를 방지하기 위하여 블록체인을 기반으로 군집전략 수행하는 방법에 대한 연구[22] 등이 진행되고 있지만, 이 또한 블록체인의 특성상 군집 구성 차량의 과반수가 공격에 참여하는 경우 무용지물이 된다.

따라서 군집의 리더로부터 군집전략의 수행 지시를 받을 때, 이 지시가 올바른 지시인지, 내 차량의 안전에 위해가 되지 않는지를 확인하는 대책이 개별 차량에게 필요하다. 다른 차량의 행동을 모니터링하며 이상행동이 감지되지는 않는지를 확인하는 것은 좋은 대책이 될 수 있다. 이와 같은 대책은 ISO/PAS 21448 표준에서 다루는 '다른 구성요소가 의도한 기능을 어떻게 사용하는지에 대한 가정'에 해당하므로 반드시 적절한 대책이 (A)의 범주에서 고려되어야 한다.

3.1.2 (B) 군집 내부에서의 상호작용에 대한 적용

군집에 소속된 자율주행차량은 개별 주행과 관련된 기능 뿐 아니라 다양한 군집전략과 관련된 기능도 수행해야 한다. 그러나 전적으로 군집 주행은 자율주행차량으로만 구성될 것을 전제조건으로 한다. 사람이 운전하는 차량은 좁은 차량 간 거리를 유지하고, 다양한 군집 주행 전략을 수행하기에는 반응속도가 떨어지고 일탈적인 행동을 할 수 있기 때문에 군집 주행을 수행하기에 부적합하다. 따라서 ISO/PAS 21448 표준에서 사용자의 실수에 대하여 다루는 '합리적으로 예측 가능한 사용자의 오용' 항목에 대한 부분은 (B)의 범주에서는 고려하지 않아도 된다.

또한 차량 주변의 영향과 관하여 군집 내부의 모든 행동은 리더에 의하여 통제되고, 환경적 요인들 또한 주행 기능을 수행하는 (A)의 범주에서 고려될 사항이므로 (B)의 범주에서는 고려의 대상이 아니다.

따라서 (B)의 범주에 대해서는 '성능 제약 및 불충분한 상황 인식' 항목에 따른 위험 요인만 분석하면 충분히 의도된 기능 안전을 달성할 수 있다. 단, 성능 제약 부분에서 기존의 ISO/PAS 21448 표준과는 달리 통신 장비의 성능 제약에 대해 발생 가능한 위험에 대하여 다양한 고려를 필요로 한다.

3.1.3 (C) 군집 외부와의 상호작용에 대한 적용

3.1.2절과 같은 이유로, (C)의 범주에서도 모든 차량의 제어는 자율주행으로 이루어지기 때문에 '합리적으로 예측 가능한 사용자 오용' 항목에 대해서는 다룰 필요가 없다. 하지만 군집 외부의 차량 및 주변 환경에 대한 적용을 고려했을 때 '성능 제약 및 불충분한 상황 인식'에 대한 항목과, '차량 주변의 영향' 항목 중 '다른 사용자' 부분에 대한 영향은 절대적이다. 특히 다른 차량과의 영향은 발생 빈도가 굉장히 높은 시나리오로 다양한 상황이 충분히 고려되어야한다.

하지만 '차량 주변의 영향' 항목 중, 도로 인프라, 환경 요소, 전자기파 간섭 부분에 대한 위험 시나리오는 (A)의 범주에서 다루는 것만으로도 충분하다. 따라서 (C)의 범주에서는 고려하지 않아도 된다.

3.2 트리거링 이벤트의 식별 및 평가 단계

ISO/PAS 21448 표준의 7절에는 트리거링 이벤트의 식별 및 평가 단계에 대한 구체적인 방법과 세부적인 고려 사항들이 나열되어 있다. 트리거링 이벤트의 분석은 알고리즘과 관련된 트리거링 이벤트와 센서 및 액추에이터와 관련된 트리거링 이벤트로 나뉜다. 본 연구에서 제시된 범주에 따라 다음과 같은 부분을 분석 과정에서 고려할 것을 제시한다.

3.2.1 (A) 개별 자율주행차량에 대한 적용

개별 차량에는 기존의 ISO/PAS 21448 표준에서 제시하는 방법을 그대로 적용하는 것을 필요로 한다. 다만 추가적으로 3.1.1절에서 제시한바와 같이 군집 내의 악의적 공격에 대한 트리거링 이벤트를 고려해야한다. 대부분의 경우에는 군집 전략의 수행을 하기 위한 메시지의 수신이 트리거링 이벤트가 된다.

3.2.2 (B) 군집 내부에서의 상호작용에 대한 적용

본 연구에서는 군집을 구성하는 차량들 사이에서 발생 가능한 트리거링 이벤트의 분석 과정에서 다음 〈표 2〉와 같은 사항을 추가적으로 고려할 것을 제안한다.

〈표 2〉 군집 주행의 트리거링 이벤트

카테고리	트리거링 이벤트
알고리즘 관련 트리거링 이벤트	• 내 차량의 현재 위치 및 속도 • 다른 차량의 위치 및 속도 • 도로 인프라
통신 기능 관련 트리거링 이벤트	• 다른 통신 모듈의 기계적 장애 • 무선 신호 간섭 • 도달 거리 • 정확성 • 지연 시간 • 내구성

한편 다른 차량들로부터 제공된 정보를 분석하고 군집의 행동을 결정하는 군집 리더는 다음 〈표 3〉과 같은 내용에 대하여 추가적인 트리거링 이벤트 분석을 수행해야 한다.

〈표 3〉 군집 리더가 고려해야할 트리거링 이벤트

카테고리	트리거링 이벤트
군집 리더에게 발생 가능한 트리거링 이벤트	• 전송받은 정보의 불일치 • 군집 차량으로부터 신호 미수신 • 정보의 실시간 처리 불가능

3.2.3 (C) 군집 외부와의 상호작용에 대한 적용

군집 외부의 차량 및 환경과 상호작용할 때 발생 가능한 트리거링 이벤트를 분석할 때에는 다음 〈표4〉와 같은 사항에 대한 추가적인 고려를 필요로 한다.

〈표 4〉 군집 외부와 발생 가능한 트리거링 이벤트

카테고리	트리거링 이벤트
알고리즘 관련 트리거링 이벤트	• 환경과 위치 • 도로 조건 및 교통 법규 • 다른 운전자의 기대 행동
주변환경 관련 트리거링 이벤트	• 다른 차량 위치 및 속도 • 주변 환경 파악 가능 범위 • 다른 차량의 V2X통신 여부

3.3 테스트케이스의 작성 단계

ISO/PAS 21448 표준의 부록 F에는 안전성 분단계석을 위한 시나리오 구축 시 고려해야할 요인의 예를 담고 있다. 본 연구에서는 고속도로 환경에서 군집주행인 차량의 테스트를 할 때, 기존 요인 외에 추가적으로 다음 〈표 5〉와 같은 요인의 추가적인 고려를 제시한다.

〈표 5〉 군집주행의 시나리오 구축 시 고려 요인

요 인	상 황
군집 주행	• 군집의 리더일 때 • 군집의 중간 차량일 때 • 군집의 마지막 차량일 때 • 군집의 속도 • 군집의 차량 간격 • 군집의 통신 성능 • 군집 전략을 수행 중인지 여부 • 군집의 크기
주변 차량	• 주변 차량이 군집인 경우 - 주변 군집의 크기 - 주변 군집의 차량 간 간격 • 주변 차량이 단독주행 차량인 경우 - 군집과 단독주행 차량 사이의 간격 - 단독주행 차량의 자율주행 여부 • 주변 차량이 군집과 단독주행 차량이 혼재 - 차량 구성의 순서 - 군집과 단독주행 차량 사이의 간격

위와 같이 군집주행에 적합하도록 ISO/PAS 21448 표준을 확장한 내용을 군집주행 기술 개발에 적용함으로써 안전한 군집주행을 달성하고자 하는 목표에 도달할 수 있다.

4. 사례 연구

본 연구에서 제시한 군집주행을 위해 확장된 ISO/PAS

21448 표준을 실제 SOTIF 달성 활동에 적용하기 위하여 주행중인 군집의 차선 변경 기능의 사례를 연구하였다. 미국에서 제안된 AHS에서는 자율주행 차량과 일반 수동 운전 차량이 공존하는 기간 동안 고속도로를 주행하는 차량의 안전을 확보하기 위해 자율주행차량 및 군집주행차량 전용 차선을 고속도로에 설치할 것을 제안한다. AHS에서는 전용차선을 주행 중인 자율주행 또는 군집주행 차량이 수동 운전 차량이 주행하는 차선으로 차선을 변경하는 것을 금지한다. 하지만 실제 상황에서 자율주행 전용 차선에 고장 난 차량이 존재하거나, 서행하는 차량이 존재할 경우 군집주행의 안정성과 원활한 교통의 흐름을 위해 제한적인 군집의 차선 변경 기능을 필요로 할 수 있다. 또한 미래에 모든 차량이 자율주행차량으로 구성되는 경우에는 다양한 상황에서 군집의 유동적인 차선변경 기능을 요구할 것이다. 또한 사고 회피를 위한 알고리즘의 설계를 위해 군집의 차선변경 기능을 필요로 할 수도 있다. 따라서 본 연구에서는 군집의 차선 변경 기능을 설계 및 구현하고, 해당 기능으로부터 발생 가능한 위험 시나리오로부터 SOTIF 활동을 진행하였다.

4.1 군집 차선 변경 기능의 설계

군집주행 전략은 다양한 방법으로 개발 및 수행이 가능하다. 본 연구에서는 미국 UC Davis 대학의 연구팀이 개발한 오픈소스 군집주행 시뮬레이터인 VENTOS[23]의 아키텍처를 기반으로 군집의 차선 변경 기능을 설계하였다. VENTOS 시뮬레이터는 오픈소스 도로 교통 시뮬레이터인 SUMO(Simulation of Urban MObility)[24]와 오픈소스 네트워크 시뮬레이터인 OMNET++(Objective Modular NEtwork Testbed in C++)[25]를 결합하여 개발되었다. VENTOS 시뮬레이터는 합류, 분리, 이탈 등 다양한 군집 전략을 지원하지만, 군집 전체가 주행 중 차선을 변경을 하는 기능은 지원하지 않는다.

군집주행 기술은 충분히 성숙된 자율주행기술을 바탕으로 개발된다. 자율주행차량은 주행 중 각 차량이 차선 변경을 필요로 하는 상황을 가진다. VENTOS 시뮬레이터에서 사용하는 개별 차량의 차선변경 모델에서는 다음과 같은 상황에서 차선변경을 시도하도록 정의하고 있다.

(1) 전략적(Strategic) : 자신의 경로 진행을 위함
(2) 협력적(Cooperative) : 다른 차선의 변경을 돕기 위함
(3) 속도 이득(Speed gain) : 더 빠른 주행이 가능할 경우
(4) 법규 준수(Compliance) : 교통 법규에 따르기 위함

또한 각각의 자율주행차량은 차선 변경을 하고자 하는 의도를 가진 상태에서 차선변경이 가능한 상황에 도달할 경우 차선변경을 시도한다. 위와 같이 VENTOS 시뮬레이터에서 사용하는 개별 자율주행차량의 차선변경 모델을 이용하여 다음과 같은 알고리즘으로 군집 전체의 차선변경 프로토콜을 설계하였다.

군집 전체의 차선 변경 기능을 수행하기 위하여 군집 리더는 군집 구성 차량들로부터 V2V 통신을 통해 각 차량의 센서를 통해 수집된 주변의 교통 상황과 환경을 평소에 지속적으로 전송받는다. 전송받은 정보에는 도로의 양 옆이 비어있어 차선 변경이 가능한 상태인지, 아니면 다른 차량이 존재하거나 접근하는 차량이 존재하여 차선 변경이 불가능한 상황인지에 대한 정보를 포함학 있다. 군집의 선두에서 주행하는 군집리더 차량이 차선의 변경을 필요로 할 경우, V2V 통신을 통해 전송받은 값을 토대로 군집 전체의 차선 변경 가능성을 확인한다. 차선 변경이 가능할 경우 통신을 통해 모든 군집 구성 차량에게 차선을 변경을 지시하고, 자신도 차선을 변경한다. 차선 변경 지시를 받은 군집 구성 차량들은 차선을 변경하고, 차선 변경이 완수된 후, 차선변경을 완료하였다는 확인 메시지를 군집 리더 차량에게 송신한다.

이와 같이 설계된 차선변경 기능에서 군집 리더가 수행하는 행동의 정의는 다음 〈알고리즘 1〉과 같다.

〈알고리즘 1〉 군집차선변경 기능에서 Leader의 행동

```
01 if (wantLaneChange)
02   Start LaneChangeManeuver
03   if (allFollowersLaneChangePossible)
04     Send LaneChangeMsg to followers with Dir
05     Start AckTimer
06     Perform LaneChange to target Dir
07     if (Receive AckMsg from all follwers)
08       Cancel AckTimer
09       End LaneChangeManeuver
10     else if (AckTimerExpires)
11       while (True)
12         Send ConfirmLaneChangeMsg to followes
13         Start AckTimer
14         Wait AckTimerExpires
15         if (Receive AckMsg from all followers)
16           break
17       End LaneChangeManeuver
18   else
19     Cancel LaneChangeManeuver
```

01행에서 차선 변경을 필요로 하는 경우, 군집 차선 변경을 시작한다. 03행에서 군집 리더가 변경하고자 하는 차선으로 모든 군집 구성 차량이 차선 변경이 가능한 상태이면, 04행에서 V2V 통신을 통해 차선변경 지시를 전송하고 차선을 변경한다. 05행에서 다른 군집 구성 차량들의 차선 변경을 확인하기 위한 타이머를 켜고, 모든 차량들로부터 차선 변경을 마쳤다는 메시지를 전송받으면 타이머를 끄고 차선변경 전략을 종료한다. 하지만 모든 차량들로부터 메시지를 전송받지 못할 경우, 11행~16행의 과정을 통해 차선변경의 수행 여부를 지속적으로 확인한다.

군집에서 리더가 아닌 군집 구성 차량이 차선변경 기능을 수행할 때의 행동은 다음 〈알고리즘 2〉와 같이 정의하였다.

〈알고리즘 2〉 군집차선변경 기능에서 Follower의 행동

```
01 if (Receive LaneChangeMsg from Leader with Direction)
02    Start LaneChangeManeuver
03    Perform LaneChange to target Direction
04    Send AckMsg to Leader
05 if (Receive LaneChangeMsg from Leader)
06    if (Finish LaneChange)
07       Send AckMsg to Leader
```

01행에서 군집 리더로부터 차선변경 메시지를 수신한 경우, 지시받은 방향으로 차선 변경을 수행한다. 차선 변경을 마친 후 확인 메시지를 군집 리더에게 전송한다. 한편 발송한 메시지가 군집 리더에게 전달이 되지 않아 05행에서 차선 변경을 마친 이후에 리더로부터 차선변경 확인 메시지를 받은 경우, 확인 메시지를 다시 군집 리더에게 전송한다.

위와 같이 설계된 군집주행에서의 차선변경 기능을 VENTOS 시뮬레이터의 소스코드를 수정하여 구현하였다. 〈그림 3〉은 VENTOS 시뮬레이터에서 차선변경 기능이 수행되는 시뮬레이션 장면이다. 녹색 차량은 단독주행 중인 차량이고, 빨간색과 파란색 차량은 군집주행 중인 차량이다. 이때 빨간색 차량은 군집의 리더임을 의미한다.

〈그림 3〉 군집주행의 차선변경 기능 시뮬레이션 장면

〈그림 3〉의 각 장면별 설명은 다음과 같다. 1차선에는 녹색의 단독 주행차량이 있고 2차선에는 세 대의 차량으로 구성된 군집이 주행 중이다. (a)에서 선두 차량이 왼쪽으로 차선 변경을 희망하고 있다. 하지만 녹색 차량으로 인해 전체 군집의 차선 변경이 불가능한 상황이다. 하지만 녹색 차량의 양보로 (b)와 같이 군집의 차선 변경이 가능한 상황에 도달하였고, (c)에서 전체 군집이 차선 변경을 수행하고 있다. (d)는 군집이 차선 변경을 마친 장면이다.

4.2 SOTIF 활동

4.1절에서 설계한 차선 변경 기능을 바탕으로 해당 기능으로부터 발생 가능한 위험을 최소화하기 위해 ISO/PAS 21448 표준에서 정의한 SOTIF 활동을 제한적으로 수행하였다. SOTIF 활동의 수행 과정에서 군집주행에 적합하도록 본 연구에서 제시한 수정 및 확장 내용을 고려하였다. 먼저 표준의 내용과 같이 다음과 같이 기능 및 시스템 사양 파악을 수행하였다.

- 기능 목표 : 전체 군집이 차선 변경을 필요로 하는 경우, 전체 군집을 안전하게 이웃 차선으로 변경한다.
- 의도한 기능이 활성화 또는 비활성화되는 사용 사례 : 전방 고장 및 저속 차량의 우회, 주행의 효율성 증가를 위하여 활성화된다. 또 차선 변경이 금지된 도로이거나, 주변 교통량이 밀집되는 경우 비활성화된다.
- 시스템 관련 : 자율주행차량이 차선변경을 할 때와 동일한 시스템을 기반으로 수행된다.

또한 위험 식별 단계에서 본 연구에서 제시한 방법을 통하여 그 예시로 다음 〈표6〉과 같이 네 가지의 위험 시나리오를 식별하였다.

〈표 6〉 위험 시나리오의 식별

(B) 군집 내부에서의 상호작용에서 발생 가능한 시나리오
1) 군집 리더가 차선 변경을 위하여 군집 구성 차량으로부터 수집한 정보들이 서로 상이하여 판단이 어렵다.
2) 개별 차량이 자체 비상안전 기능을 활성화하여 군집의 통제에 따르지 않는다.
(C) 군집 외부와의 상호작용에서 발생 가능한 시나리오
3) 변경하려는 차선의 후방에서 차량이 고속으로 접근한다.
4) 변경하려는 차선의 반대편 차선에서 동시에 같은 차선에 진입을 하려 한다.

위의 식별된 네 가지 시나리오를 토대로 다음 〈표 7〉와 같이 SOTIF 달성 기준에 도달하여 위험을 충분히 감소시킬 수 있을 때까지 〈그림 2〉의 SOTIF 활동 흐름도에 따라 반복적으로 SOTIF 활동을 진행하는 사례 연구를 수행하였다.

사례 연구에서와 같이 SOTIF 종료 기준에 도달하면 SOTIF 활동을 종료하게 된다. 그러나 실제 군집기술 개발 과정에서는 훨씬 더 많은 발생 가능한 위험 시나리오를 대상으로 SOTIF 활동을 진행해야 하며, 차량의 모든 기능에 대하여 SOTIF 활동을 수행해야 하기 때문에 막대한 시간과 비용, 노력을 필요로 할 것이다. 또한 SOTIF Area 3에 해당하는 알려지지 않은 위험 시나리오가 S10 단계를 수행하면서 추가적으로 식별될 수 있다. 이러한 경우 해당 시나리오는 알려진 위험 시나리오로 전환된다. 새롭게 식별된 시나리오는 SOTIF 활동의 흐름도에 따라 S11, S12, S14 단계의 활동을 거쳐 알려진 안전 시나리오로 전환해야 한다.

〈표 7-1〉 차선 변경 기능의 SOTIF 활동 (1)

Step	SOTIF 활동 내용
S1	군집 리더 차량이 차선 변경의 필요성을 인식할 경우, 해당 방향으로 전체 군집의 차선 변경이 가능한 상황이면 통신을 통해 전체 군집의 차선을 변경시킨다.
S2	군집 리더가 전체 군집의 차선 변경이 가능한 상태인지를 판단하였으나 군집 구성 차량들로부터 수집된 데이터들이 상이하여 판단이 어렵다.
S3	**(No)** 잘못된 정보를 통해 내린 판단은 충돌 등의 위험을 만들 수 있다.
S4	센서의 간섭 및 음영지대로 인해 서로 다른 값을 전송하였다.
S5	**(No)** 좁은 차량 간 간격을 유지하는 군집주행에서는 자주 발생 가능한 상황이다.
S6	센서의 간섭을 최소화하는 조치를 취하고, 안전에 더 보수적인 값을 행동의 결정에 사용한다.
S1	**추가명세1)** 차선 변경 시 주변 상황에 대한 정보가 충돌하는 경우, 안전에 더 보수적인 값을 사용한다.
S2	개별 차량에 가해진 위험으로 인해 자체 비상안전 기능이 활성화되어 군집의 통제를 벗어난다.
S3	**(No)** 여러 대의 자율주행차량이 기민하게 상호작용하는 군집주행의 특성 상 통제되지 않는 차량은 다른 차량들에게 위협요소가 될 수 있다.
S4	개별 차량에 대한 보안 공격은 비상안전 기능을 활성화시킨다.
S5	**(No)** 발생 가능성이 현저히 낮지만 발생시 매우 큰 위험 심각도를 가지므로 대책을 필요로 한다.
S6	차량의 개별 행동 인식 즉시 차선 변경 전략의 수행을 취소하고 차량 사이의 안전거리를 확보하도록 하한다. 이어서 적절히 군집을 분리 또는 해체하여 개별 행동 차량을 군집으로부터 이탈시킨다.
S1	**추가명세2)** 비상안전 기능을 활성화하여 군집의 통제에서 벗어난 차량을 군집으로부터 이탈시킨다.
S2	변경하려는 차선의 후방에서 고속으로 접근중인 차량이 존재한다.
S3	**(No)** 접근중인 차량의 속도와 군집 최후방 사이의 거리를 고려했을 때 충돌이 발생할 수 있다.
S4	리더 차량이 충분한 안전거리를 감안하지 않고 차선 변경 판단을 하였다.
S5	**(No)** 고속도로 환경에서는 발생 빈도가 매우 높으므로 위험하다.
S6	차선 변경 여부의 판단 알고리즘을 개선한다.

(⇒ 표 7-2로 이어짐)

〈표 7-2〉 차선 변경 기능의 SOTIF 활동 (2)

Step	SOTIF 활동 내용
S1	**추가명세3)** 차선 변경 여부를 충분히 더 안전하게 판단하도록 개선하였다.
S2	군집의 차선 변경시, 변경하고자 하는 차선의 반대편 차선으로부터 같은 차선으로 동시에 변경을 시도한다.
S3	측방 충돌을 유발할 수 있으므로 높은 위험성을 갖는다.
S4	차선 변경 판단을 할 시점에서는 변경하고자 하는 차선이 비어있었으나 차선 변경을 시작하자, 반대편에서 차량이 같은 차선으로 진입을 하였다.
S5	단독 자율주행차량보다 군집은 더 길이가 길기 때문에 해당 시나리오의 발생빈도가 크다.
S6	차선 변경을 수행하는 동안 주변 차량의 상황을 모니터링한다. 같은 차선으로 진입하는 차량을 감지한 경우 즉시 차선 변경을 취소하고 모든 군집 구성 차량들을 원래 차선으로 되돌린다.
S1	**추가명세4)** 같은 차선으로 동시에 차선 변경을 시도시, 차선 변경을 취소한다.
S5	**(Yes)** 주어진 시나리오에서 SOTIF 관련 위험이 허용된다.
S7	알려진 위험 시나리오(Area2)의 검증을 위하여 다양한 날씨, 도로 환경, 교통 상황 등에서 개선된 차선 변경 기능의 테스트 계획과 구체적인 검증 목표를 세운다. 또한 알려지지 않은 위험 시나리오(Area 3)의 검증을 위하여 다양한 실제 상황에서 차선 변경 기능을 1000 시간 이상 테스트하고 구체적인 검증 목표를 세운다.
S8	정의된 알려진 위험 시나리오의 검증 방법대로 실제 테스트를 수행한다.
S9	**(Yes)** 다양한 차선 변경 상황에서 사전에 고려된 위협요인이 발생하였음에도 추가된 안전대책이 적절히 작용하여 안전한 주행이 가능하였다.
S10	정의된 알려지지 않은 위험 시나리오의 검증 방법대로 실제 테스트를 수행한다.
S11	**(Yes)** 다양한 실제 상황의 주행 테스트에서도 다른 차량과 주어진 최소 안전거리를 항상 유지하며 차선을 변경하였다. 또한 테스트 과정에서 새로운 위험 시나리오가 발견되지 않았다.
S12	S8과 S10에서 수행된 테스트의 결과가 S7에서 정의한 검증 목표에 도달하였다.
S13	해당 시나리오에서 잔여 위험은 없는 것으로 판정하였다.
S15	차선변경기능에 대한 SOTIF 활동을 종료한다.

〈표 7〉 차선 변경 기능의 SOTIF 활동

5. 결론 및 향후 연구

본 연구에서는 ISO/PAS 21448 표준을 군집주행에 적용하기에 적합하도록 표준의 일부 내용을 수정 및 확장하여 군집주행 기술의 적절한 위험 분석 방법으로 사용할 수 있도록 하였다. 또한 확장된 표준을 군집의 차선변경 기능에 대하여 적용하는 사례 연구를 수행함으로써 다른 군집주행의 기능에 대해서도 군집주행의 안전을 확보하는 기술을 개발하는데 도움이 되도록 하였다.

향후 연구로는 차량의 안전을 위한 또 다른 표준인 ISO 26262 표준과 통신 기능을 가진 차량에 대한 보안 위협을 다루는 ISO/SAE 21434 표준을 군집주행의 특성에 맞게 확장하여 다방면에서 군집주행의 안전을 확보하는 방안에 대하여 연구하고자 한다.

Acknowledgement

본 연구는 과학기술정보통신부의 재원으로 한국연구재단의 지원을 받아 수행되었음.

(NRF-2020R1A2C1007571, NRF-2017M3C4A7066479)

참 고 문 헌

[1] NHTSA, The Economic and Societal Impact Of Motor Vehicle Crashes, 2010(Revised), 2015.

[2] ISO 26262-1:2011, Road vehicles - Functional safety, 2011.

[3] ISO/PAS 21448:2019, Road vehicles — Safety of the intended functionality, 2019.

[4] Tsugawa, Sadayuki, Shin Kato, and Keiji Aoki. "An automated truck platoon for energy saving." 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2011.

[5] Bergenhem, Carl, et al. "Overview of platooning systems." Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria, 2012.

[6] 현대차 트럭 플래투닝(군집주행) https://www.youtube.com/watch? v=-l-S-N3xijQ (accessed at December, 2020.)

[7] V. Milanés, and S. E. Shladover, "Modeling cooperative and autonomous adaptive cruise control dynamic responses using experimental data," Transportation Research Part C: Emerging Technologies, Vol.48, pp.285-300, 2014.

[8] L. Xiao, M. Wang, and B. van Arem, "Realistic Car-Following Models for Microscopic Simulation of Adaptive and Cooperative Adaptive Cruise Control Vehicles," Transportation Research Record: Journal of the Transportation Research Board, Vol.2623, No.1, pp.1-9, 2017.

[9] L. Xiao, M. Wang, W. Schakel, and B. van Arem, "Unravelling effects of cooperative adaptive cruise control deactivation on traffic flow characteristics at merging bottlenecks," Transportation Research Part C: Emerging Technologies, Vol.96, pp.380-397, 2018.

[10] IEEE 802.11p-2010 - IEEE Standard for Information technology-- Local and metropolitan area networks-- Specific requirements-- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments, 2010.

[11] 3GPP, 3GPP Standard Release 14

[12] Amoozadeh, Mani, et al. "Platoon management with cooperative adaptive cruise control enabled by VANET." Vehicular communications 2.2: 110-123, 2015.

[13] Lam, Stanley, and Jayantha Katupitiya. "Cooperative autonomous platoon maneuvers on highways." 2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. IEEE, 2013.

[14] Kirovskii, O. M., and V. A. Gorelov. "Driver assistance systems: analysis, tests and the safety case. ISO 26262 and ISO PAS 21448." IOP Conference Series: Materials Science and Engineering. Vol. 534. No. 1. IOP Publishing, 2019.

[15] Santini, Stefania, et al. "A consensus-based approach for platooning with intervehicular communications and its validation in realistic scenarios." IEEE Transactions on Vehicular Technology 66.3, 1985-1999, 2016.

[16] Ploeg, Jeroen, and Redmer de Haan. "Cooperative Automated Driving: From Platooning to Maneuvering." SMARTGREENS. 2019.

[17] Monkhouse, Helen E., Ibrahim Habli, and John McDermid. "An Enhanced Vehicle Control Model for Assessing Highly Automated Driving Safety." Reliability Engineering & System Safety : 107061, 2020.

[18] Horowitz, Roberto, and Pravin Varaiya. "Control design of an automated highway system." Proceedings of the IEEE 88.7 : 913-925, 2000.

[19] Hsu, Harry Chia-Hung, and Alan Liu. "Kinematic design for platoon-lane-change maneuvers." IEEE Transactions on Intelligent Transportation Systems 9.1 : 185-190, 2008.

[20] ISO/SAE DIS 21434 : Road vehicles — Cybersecurity engineering

[21] Khattak, Zulqarnain H. "Impact of cyberattacks on safety and stability of connected and automated vehicle platoons under lane changes.", Forthcoming in Accident Analysis and Prevention, 2020.

[22] Singh, Pranav Kumar, et al. "Intergrating Blockchain with CACC for Trust and Platoon Management", Cryptocurrencies and Blockchain Technology Applications : 77-97, 2020.

[23] VENTOS, https://maniam.github.io/VENTOS/ (accessed at December, 2020.)

[24] Behrisch, Michael, et al., "SUMO — simulation of urban mobility: an overview." SIMUL 2011 : The Third International Conference on Advances in System Simulation, Barcelona, pp.63-68, 2011,

[25] OMNet++ : Discrete Event Simulator, https://omnetpp.org/ (accessed at December, 2020.)

근로계약서를 위한 SPL 기반 실행 가능한 리카르디안 컨트랙트 생성 프레임워크

홍준기 ○, 김순태, 류덕산, 박수용, 박용범
전북대학교 소프트웨어공학과, 서강대학교 컴퓨터공학과, 단국대학교 소프트웨어학과
[e-mail: {rlwns012, stkim, duksan.ryu}@jbnu.ac.kr, sypark@sogang.ac.kr, ybpark@dankook.ac.kr]

Executable Ricardian Contract Generation Framework for Labor Contract based on SPL(Software Product Line)

Joongi Hong○, Suntae Kim, Duksan Ryu, Sooyong Park, Youngbeom Park
Dept. of Software Engineering, Jeonbuk National University
Department of Computer Science & Engineering, Sogang University
Department of Software, Dankook University

요 약

리카르디안 컨트랙트는 사람과 기계가 모두 읽을 수 있는 디지털 계약서로 사실 증명이 가능하고 법적 증거 자료로 활용할 수 있다. 최근 블록체인과 함께 사용되어 무결성이 보장되는 계약서로써 가능성을 주목받고 있다. 하지만 기존 종이 계약서와 마찬가지로 계약 사실 증명만 가능할 뿐 계약금 지불과 같은 계약 이행 여부 증명은 불가능하다. 또한 작성된 다양한 계약서를 일일이 리카르디안 컨트랙트로 변환해야하는 번거로움이 있으며, 계약서의 재사용성이 낮다. 우리는 계약 이행 증명도 가능한 계약서를 구현함과 동시에 사용자에게 다양한 계약서 템플릿을 제공하여 변환의 번거로움을 줄이고 계약서의 재사용성을 높이고자 한다. 본 연구에서는 목표를 달성하기 위해 피처 모델을 활용한 SPL 기반 실행 가능한 리카르디안 컨트랙트 생성 프레임워크 제안한다. 실행 가능한 리카르디안 컨트랙트는 사실 증명과 이행 증명이 함께 가능한 계약서이며 프레임워크의 자동화된 과정으로 생성된다. 더불어 피처 선택에 따른 다양한 계약서를 사용자에게 제공하여 계약서의 재사용성을 높인다. 우리는 근로계약서 도메인에 적용한 프로토타입을 구현하였으며 GQM 방법을 통해 프레임워크 검증 및 평가를 진행하였다. 마지막으로 실험 결과를 바탕으로 실행 가능한 리카르디안 컨트랙트와 프레임워크의 가능성을 확인한다.

1. 서 론

리카르디안 컨트랙트(Ricardian Contract)는 둘 이상의 계약당사자들이 서로 행동하기 위한 계약 조건과 내용을 정의한 디지털 문서다. XML(Extensible Markup Language) 또는 JSON(JavaScript Object Notation) 형태로 작성되며, 산문으로 작성된 계약서 중에 파라미터 값을 태깅(tagging)한다. 리카르디안 컨트랙트는 산문으로 계약서를 작성하고 계약 내용 중 일부에 태깅을 한 뒤 계약당사자들의 전자 서명을 입력받아 암호화하여 저장한다[1].

최근 블록체인(Blockchain)[2]과 함께 사용되어 리카르디안 컨트랙트는 무결성이 보장되는 계약서로써 가능성을 주목받고 있다. 블록체인은 탈중앙화 및 분산 저장소, 합의 알고리즘, P2P(Peer to Peer)등을 한데 뭉쳐 저장된 데이터의 무결성을 보장해주는 기술이다. 이에 따라 리카르디안 컨트랙트를 활용하여 다양한 비즈니스 계약서를 작성하고 블록체인상에 배포하여 위변조 불가능한 계약이 가능해진다.

하지만 리카르디안 컨트랙트는 2가지 문제를 안고 있다. 첫 번째로 기존 종이 계약서와 마찬가지로 계약 내용에 대한 사실 증명이 가능할 뿐 이행 증명은 불가능하다. 두 번째로 다양한 계약서를 일일이 디지털 문서로 변환해야하는 번거로움이 있고 계약서의 재사용성이 낮다.

먼저 리카르디안 컨트랙트에는 일반적인 계약서처럼 계약 사실만 담긴다. 근로계약서를 기준으로 예를 들자면 사업주와 근로자 정보, 계약 기간, 근로 시간, 임금 등 계약 사실에 관한 내용만을 파악할 수 있다. 하지만 계약상 분쟁이 발생하였을 때 계약 사실에 대한 증명뿐만 아니라 계약의 이행 여부도 따져보게 되는데 계약서만으론 이행 증명은 불가능하다. 따라서 우리는 디지털화된 계약에서 사실 증명뿐만 아니라 이행 증명도 가능하도록 하고자 한다.

다음으로 현실에서는 다양한 형태의 계약서가 존재하는데 이를 디지털 문서로 변환하기 위한 작업을 일일이 해야 하기 때문에 번거롭다. 예를 들어 근로계약서만 하더라도 정규직, 인턴, 일용직, 계약직에 따라 양식이 달라지는데 계약서가 변경될 때마다 변환해야 한다. 우리는 자동화된 과정으로 리카르디안 컨트랙트 작성의 번거로움을 줄이고 다양한 계약서의 재사용성을 향상시키고자 방법을 모색하였다.

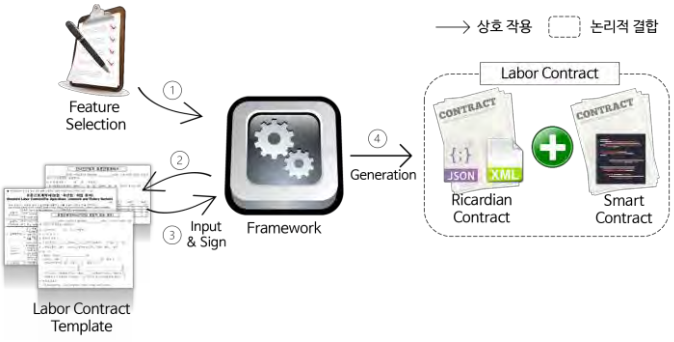


그림 1 실행 가능한 리카르디안 컨트랙트 생성 프레임워크 개요

본 논문에서는 언급한 문제를 해결하고자 실행 가능한 리카르디안 컨트랙트 생성 프레임워크를 제안한다. 그림 1은 프레임워크의 개요다. 프레임워크는 피처 모델(Feature Model)[3]을 활용하는 SPL(Software Product Line)[4] 방법을 기반으로 한다. 사용자는 계약서 항목 선택에 따라 프레임워크로부터 계약서 템플릿을 제공받는다. 실행 가능한 리카르디안 컨트랙트는 리카르디안 컨트랙트와 스마트 컨트랙트가 함께 생성되어 사실 증명과 이행 증명이 가능한 계약서이며 프레임워크를 통해 자동화된 과정으로 생성된다. 프레임워크는 사용자에게 다양한 계약서 템플릿을 제공하고 계약서의 재사용성을 높일 수 있다.

연구를 진행하기 위해 우리는 연구의 범위를 한정하고 고려사항을 정리하였다. 먼저 연구의 범위는 근로계약서 도메인으로 한정하였다. 왜냐하면 모든 계약서에 적용할 수도 있지만 시간과 자원의 한정으로 범위를 축소하여 선정하였다. 근로계약서의 경우 통계청에 따르면 2020년 기준 근로자 수가 약 1,867만명으로 일반적으로 많은 사람들이 경험할 수 있고, 근로계약서는 비교적 짧은 계약서로 복잡하지 않다[5]. 근로기준법[6]에 의해 필수/공통 항목이 많고 기업, 직종, 국적 등에 따라 약간씩 차이가 있어 SPL 방법을 사용하기에 적합하기도 하다. 더불어 근로 시장에는 임금체불 관련 문제가 있는데 고용노동부에 따르면 2019년 기준 임금체불자는 34.5만명, 임금체불액은 약 1.72조원에 달한다[7]. 우리는 근로 시장의 임금체불 문제를 실행 가능한 리카르디안 컨트랙트의 위변조 불가능한 계약서와 자동화된 계약 이행, 블록체인에 이행 증명 기록을 통해 분쟁 발생 요소를 줄이고 분쟁 발생 시 계약 이행 증명을 블록체인상에서 빠르게 파악하여 해결할 수 있도록 지원 가능할 것이라 기대한다.

고려 사항으로는 먼저 근로계약서를 기준으로 정부, 공공기관, 법, 계약당사자 등 다양한 관련 주체들이 참여할 수 있는데 본 연구에서는 일부를 가정하여 배제하고 연구를 진행하였다. 그리고 적용되는 법에 따라 제약사항을 반영하고자 하였으며 블록체인과 스마트 컨트랙트의 특성을 반영한 프레임워크를 구성하고자 노력했다. 마지막으로 본 연구에서는 SPL을 차용한다. 전통적인 SPL과 일치하는 과정을 거치지 않았으며 블록체인과 계약서 등의 특성을 반영하여 약간의 튜닝을 했다. 해당 사항은 각 단계에서 언급하도록 하겠다.

실행 가능한 리카르디안 컨트랙트 생성 프레임워크를 검증 및 평가하기 위해 평가 척도를 식별하기 위한 방법인 GQM(Goal Question Metrics)[8]을 활용하였다. 실행 가능한 리카르디안 컨트랙트를 달성하기 위해 실행 가능 여부와 다양한 계약서 템플릿 지원이라는 2가지의 목표를 설정하였고 목표를 달성하기 위한 질문과 지표로 실험 결과를 냈다. 우리는 웹페이지 형태의 프레임워크 프로토타입을 만들었으며 실제로 작성된 근로계약서를 입력하여 동일한 계약서가 생성되는지를 판별하였다.

마지막으로 리카르디안 컨트랙트를 활용한 관련 사례와 연구를 비교분석하였다. 이오스(EOS)[9], 오픈바자(Openbazaar)[10] 등 실사례와 우리 연구와 매우 흡사한 스마트 컨트랙트 템플릿(Smart Contract Template)[11]을 5점 척도 활용하여 비교해보았다. 또한 우리 연구에서 추가적으로 논의되어야 할 내용과 향후 연구에서 고려되어야 할 사항을 토의 장에서 알아본다.

2. 배경지식

2장에서는 본 연구의 이해를 돕기 위해 근로계약서와 SPL, 피처 모델, 리카르디안 컨트랙트, 블록체인, 스마트 컨트랙트 순으로 배경지식을 서술한다.

2.1. 근로계약서

근로계약서는 근로자와 사업주 사이에 작성되는 계약서이다. 근로기준법에 따라 필수 항목이 존재하며, 최저임금법, 주52시간제, 전자계약서로 작성될 경우 전자서명법, 전자문서 및 전자거래 기본법 등 지켜야 하는 법적 제약사항이 존재한다. 고용노동부는 표준근로계약서를 제공하고 있으며, 사업주는 필수 항목을 포함하고 근로 형태, 직종, 기업, 국적 등에 따라 항목을 부분적으로 추가하여 근로계약서를 작성할 수 있다. 근로계약서의 경우 통계청에 따르면 2020년 기준 근로자 수가 약 1,867만 명으로 일반적으로 많은

사람이 경험할 수 있는 계약이며 고용노동부에 따르면 2019년 기준 임금체불 관련 문제로 임금체불자는 34.5만 명, 임금체불액은 약 1.72조 원에 달한다. 임금체불 문제는 고용노동부가 수사하고 중재하여 계약의 교차검증과 증언 등을 통해 판별된다[12].

2.2. SPL(Software Product Line)과 피처 모델(Fearture Model)

SPL은 소프트웨어공학 기법의 하나로 모듈화를 통해 재사용할 수 있는 자산을 만들고 자산을 활용하여 유사한 소프트웨어 제작을 위해 사용된다. SPL에는 몇 가지 종류가 있지만 본 연구에서는 피처 모델을 기반으로 한 방법을 적용하였으며 도메인 분석, 요구사항 분석, 도메인 구현, 파생물 생성 단계를 거친다. 피처 모델이란 도메인을 분석하여 도메인의 특징적인 요소들을 뽑아내고 분류하여 SPL을 활용한 소프트웨어 제작에 반영하기 위해 사용하는 모델이다. SPL은 피처 모델의 구성에 따라 디자인 패턴과 프레임워크 방식 등을 이용하여 구현해낼 수 있다. SPL의 장점으로는 소프트웨어의 재사용성과 확장성, 코드 품질 향상을 꼽을 수 있다.

2.3. 리카르디안 컨트랙트(Ricardian Contract)

리카르디안 컨트랙트는 1995년 이안 그릭에 의해 최초로 제안된 개념이며, 비즈니스 계약서를 담기 위해 사용된다. 최근 블록체인 분야에서 계약 사실 증명을 위한 전자계약서 가능성을 주목받고 있다. JSON 또는 XML 형태로 되어 있는 문서이며, 산문으로 되어 있는 계약서 내용 중 파라미터로써 활용되는 내용에 태그(Tag)를 삽입하여 정보를 표시한다. 리카르디안 컨트랙트는 사람과 기계 모두가 이해할 수 있으며, 계약 사실 증명이 가능하고 법적 증거 자료로 활용될 수 있다. 대표적으로 이오스와 R3 Corda, 오픈바자에서 사용 중이다.

2.4. 블록체인(Blockchain)

블록체인은 탈중앙화와 합의 알고리즘, 분산 저장소, P2P(Peer to Peer) 등을 활용하여 데이터의 무결성을 보장해주는 기술이다. 블록에는 사용자들의 트랜잭션 정보들이 들어가며, 블록들은 이전 블록의 해시값을 포인터로 하여 체인을 형성한다. 생성된 블록에 많은 블록이 이어 붙을수록 해당 블록의 위변조는 더욱 어려워지게 된다. 블록체인은 2009년 비트코인[13]에 의해 탄생하였으며, 이더리움[14]과 하이퍼레저[15] 등 새로운 블록체인이 나오면서 발전하고 있다.

2.5. 스마트 컨트랙트(Smart Contract)

스마트 컨트랙트[16]는 닉 사보(Nick Szabo)에 의해 제안된 개념으로 프로그래밍 언어로 작성된 전자 계약서이다. 스마트 컨트랙트는 조건에 따라 자동으로 동작하도록 작성된 프로그램이라고 할 수 있으며 계약 이행을 가능하게 한다. 다양한 블록체인에서 핵심 기능으로 활용되고 있으며 대표적으로 슬락잇(Slock it)[17]이 있다.

3. 근로계약서를 위한 SPL 기반 실행 가능한 리카르디안 컨트랙트 생성 프레임워크

본 장에서는 실행 가능한 리카르디안 컨트랙트를 생성하는 프레임워크에 대해 논한다. 3.1절에서는 프레임워크의 개요를 설명하고 3,2절에서는 프레임워크의 피처 모델을 생성하기 위해 근로계약서를 분석하고 과정을 자세히 설명한다. 마지막 3.3절에서는 피처 모델을 기반으로 한 프레임워크를 활용하여 실행 가능한 리카르디안 컨트랙트를 생성하게 되는 과정을 자세하게 설명한다.

3.1 근로계약서를 위한 SPL 기반 실행 가능한 리카르디안 컨트랙트 생성 프레임워크 개요

SPL 기반 실행 가능한 리카르디안 컨트랙트 생성 프레임워크를 만들기 위해 본 연구에서는 피처 모델을 활용한 SPL 방법을 적용하였다. 우리는 계약서를 관찰하였고 근로계약서는 다양한 템플릿이 존재하며 관련 법 등에 따라 필수/공통 항목이 많다는 것을 확인했다. 또한 일부 기업, 직종, 근로 형태, 국적 등에 따라 상황에 맞게 가변 항목이 적용될 수 있다. 이에 따라 비슷한 형태의 파생물을 만들기에 SPL 방법이 적절하다고 판단하였다.

우리는 SPL을 차용하여 프레임워크에 적용하였으며 전통적인 프로세스를 동일하게 따랐지만, 계약서의 특성과 블록체인 및 스마트 컨트랙트를 고려해야 하는 실행 가능한 리카르디안 컨트랙트에 적합하도록 과정 일부에 튜닝을 진행하였다.

본 연구에서 제안하는 프레임워크는 그림 2와 같이 크게 두 과정로 나뉜다. 첫 번째로는 근로계약서 피처 모델링을 위한 단계이며, 두 번째는 만들어진 피처 모델을 활용한 프레임워크를 통해 실행 가능한 리카르디안 컨트랙트를 생성하는 과정이다.

피처 모델링 과정에서는 근로계약서를 수집(1.1)하고 분석(1.2)을 진행한다. 분석한 자료를 기반으로 피처 모델을 생성(1.3)하며 피처 모델의 이행 가능한 피처는 스마트 컨트랙트 코드로 모듈화하고 바인딩(1.4) 가능하도록 구성한다. 만들어진 피처 모델은 저장소(1.5)에 저장 후 프레임워크의 핵심으로 사용된다.

근로계약서 생성 과정에서는 피처 모델을 통해 구성된 프레임워크를 사용자가 피처 선택(2.1)을 하여 계약서 템플릿을 제공받고 내용 입력(2.2) 및 내용 확인, 전자서명(2.3)을 하여 실행 가능한 리카르디안 컨트랙트를 생성(2.4)한다. 생성된 계약서는 블록체인에 배포(2.5)된다.

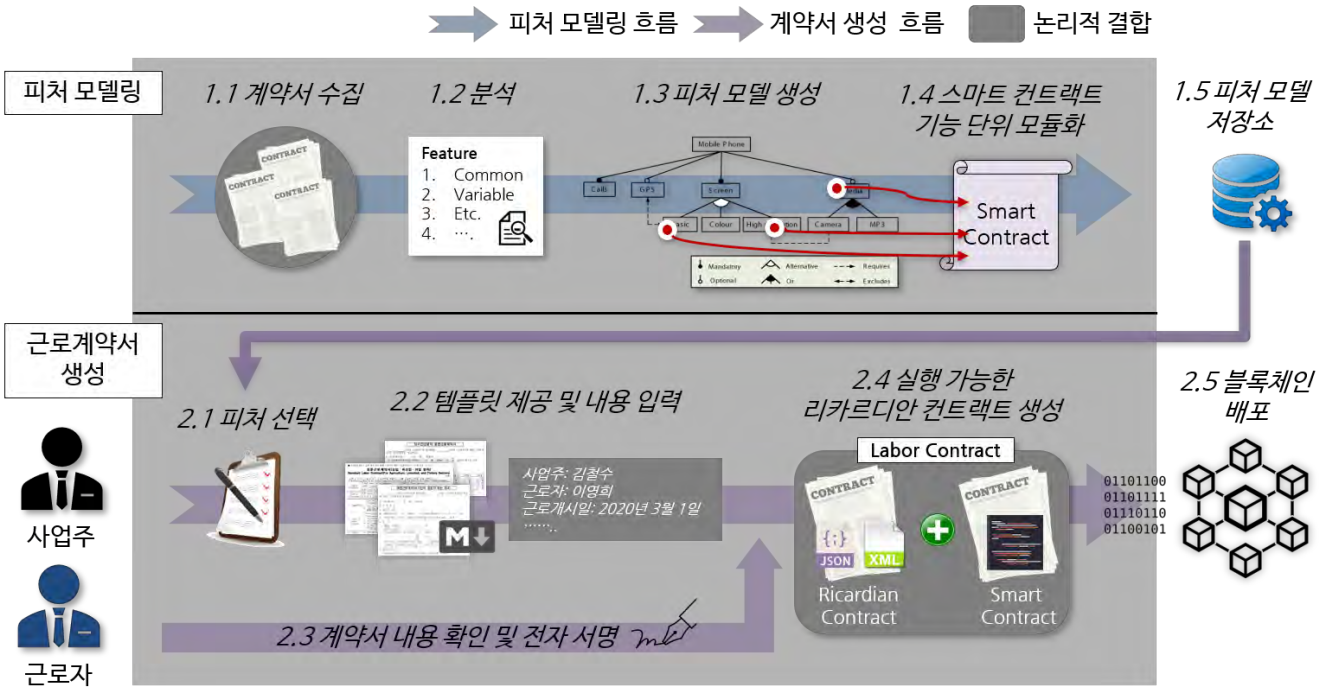


그림 2 프레임워크 전체 과정

3.2 피처 모델링 과정

피처 모델은 프레임워크를 동작하기 위한 가장 핵심이며, 해당 내용에 따라 생성되는 소프트웨어가 달라진다. 우리는 근로계약서 도메인을 피처 모델에 반영하기 위해 고용노동부에서 제공하는 표준근로계약서를 포함한 총 31개의 근로계약서를 수집 및 분석하였다. 또한 근로계약서와 관련된 법과 전자계약서로 활용하기 위해 관련 법을 함께 조사하였다. 2.2 절에서 언급한 4단계 중 도메인 분석, 요구사항 분석, 도메인 구현을 수행한다.

3.2.1 도메인 분석(Domain Analysis)

도메인 분석 단계는 그림 2의 1.1과 1.2를 포함한다. 수집한 근로계약서 수는 총 31개이며, 표준근로계약서 7종은 정규직, 비정규직, 일용직, 단시간근로자, 18세 미만 근로자, 외국인 노동자, 외국인 농업 관련 종사자를 포함하고 있다. 또한 사기업, 병원, 유치원, 연구기관, 교육기관, 대학교 등의 근로계약서를 수집하였다. 근로 형태로는 정규직뿐만 아니라 인턴, 계약직, 파견직 등이 포함되어 있다.

근로계약서가 법적 효력을 얻기 위해서 반드시 적용되어야 하는 근로기준법을 조사하였으며 필수/공통 항목을 정리하였다. 해당 항목은 5가지로 임금, 소정근로시간, 휴가 및 휴일, 근로 조건, 계약서 교부 등이 계약서에 반드시 명시되어야 한다. 또한 전자계약서의 형태로 전자계약서를 위한 법률도 조사하여 적용하고자 하였다. 정리된 항목은 총 14가지이며 표 1과 같다.

표 1 필수/공통 항목

구분	내용
계약 당사자 정보	이름, 회사 정보, 연락처, 주소
근로계약 개요	사업주와 근로자는 다음과 같이 근로계약을 체결한다.
근로계약기간	근로 시작 날짜(비정규직은 유효기간)
근무장소	정해진 근무 장소 작성
업무 내용	정해진 업무 내용
소정근로시간	출퇴근시간 및 휴게시간
근무일 및 휴일	주 출근 일수 및 유급휴일 지정
임금	기본급, 임금지급일, 지급방법 등
연차유급휴가	근로기준법에 따른 연차유급휴가 지급
사회 보험 적용 여부	고용보험, 산재보험, 국민연금, 건강보험
근로 계약서 교부	사업주는 근로계약을 체결함과 동시에 본 계약서를 사본하여 근로자의 교부요구와 관계없이 근로자에게 교부함(근로기준법 제17조 이행)
근로계약 이행 의무	사업주와 근로자는 각자가 근로계약, 취업규칙, 단체협약을 지키고 성실하게 이행하여야 함
기타	이 계약에 정함이 없는 사항은 근로기준법령에 의함
계약 날짜	계약서를 작성한 날짜

가변 항목은 근로 형태, 직종, 기업, 국적 등에 따라 달라진다. 표준근로계약서만으로도 다양한 가변 항목을 찾을 수 있으며, 사기업, 병원, 유치원, 연구기관, 교육기관, 대학교의 특성에 따라 적용되는 항목들을 찾아볼 수 있다. 가변 항목에는 근로계약서에 필수적으로 명시해야 하는 요소도 있고 상황에 따라 적용될 수 있는 항목이 있다. 정리된 가변 항목은 표 2와 같다.

표 2 가변 항목

구분	내용
내국인/외국인	외국인일 경우 계약서에 영어 근로 내용 표기
숙식 제공 여부	외국인의 경우 숙식 제공 여부 항목 제공
18세 미만 관련 법률	친권자 동의서, 관련 법률 변경
추가 임금	수당, 상여금, 식대, 교통비 등
지적재산권	지적재산 외부 사용 금지
개인정보보호	회사의 개인정보 유출 금지
자료 반환	사업주가 제공한 모든 자료 반환
사규 및 비밀준수	사기업의 규칙 준수 및 비밀 유출 금지
소정근로시간	단기간 근로자의 경우 테이블 형태의 시간표 작성, 24시간 격일제 근무
퇴직급여	조건에 따라 퇴직금 지급 여부 및 금액 결정
계약 기간	일용직, 계약직에 따른 계약 기간 산정, 수습 기간
복지 관련 규정	출산 휴가, 생리 휴가, 육아 휴직, 배우자 출산 휴가
포괄임금제	직종에 따른 허용 법률 명시

근로계약서의 디지털화 및 블록체인상에 배포하여 사용하기 위해 우리는 추가로 필요한 정보를 필요로 한다. 해당 항목은 실제 근로계약서에는 명시하여 작성되는 것이 아니며 본 연구에서 분석한 결과 계약 사실 증명과 이행 증명을 위해 추가로 필요하여 계약서에 담았다. 추가 필요 정보는 표 3과 같다.

표 3 추가 항목

구분	내용
근로 형태	정규직, 비정규직, 단기간 근로자, 일용직 등
포괄임금제	포괄임금제 여부
업종	포괄임금제 가능 여부 판단을 위해 업종 구분

마지막으로 근로계약서에서 고려해야 하는 다른 측면은 법과 관련된 내용이다. 법은 근로계약서의 제약사항을 만든다. 예를 들어 정규직은 계약 기간의 정함이 없어야 하며, 1년 이상 근로자에게는 정규직이든 비정규직이든 퇴직금을 지급해야한다. 이와 같은 요소를 피처 모델의 제약사항에 반영하였다.

3.2.2 요구사항 분석(Requirement Analysis)

근로계약서에는 일련의 절차가 있다. 계약 절차를 BPM(Business Process Modeling)[18] 방식으로 분석하고 요구사항을 추출하였으며, 해당 내용을 피처 모델에 반영하고자 하였다. 근로계약서의 절차적인 요구사항 이외에도 법적 제약사항을 반영하고 디지털화하기 위한 요구사항들을 파악하였다. 또한 블록체인과 스마트 컨트랙트의 원격 제어 관련 요구사항을 포함하였다.

3.2.3 피처 모델 생성

앞 절에서 진행한 분석 자료를 토대로 피처 모델을 생성하였다. 피처 모델에는 계약서의 사실 증명을 위한 내용이 담겨있으며, 동시에 이행 가능한 요소를 구분하여 포함한다. SPL의 일반적인 피처 모델은 실행 가능한 리카르디안 컨트랙트를 모두 표현하지 못해 약간의 튜닝을 진행하였다. 기본적인 구성은 대부분 동일하나 이행 증명을 위한 피처를 구분해내는 과정이 추가되었으며, 색을 달리하여 표시하는 방법으로 한 차원을 더 생각하여 피처 모델을 생성하였다.

이행 증명을 위한 피처를 구분하기 위해서 우리는 다음과 같은 기준을 활용하였다. 해당 기준은 관련 연구인 스마트 컨트랙트 템플릿에서 제안된 내용을 근로계약서에 맞게 추가 및 수정, 변형하였다. 내용은 아래와 같다.

* 고정/가변적 계약 이행 구분 - 계약 이행이 일정 주기 또는 유연하게 이행되는지 구분
* 디지털화되어 계약 이행 증명 가능 여부 - 계약이 디지털화 가능한 항목이어야 한다.
* 프로그램 코드를 통해 자동화 가능한 데이터 - 데이터를 프로그램 코드로 자동화 가능한지 여부

우리는 위의 3가지 기준으로 피처 모델을 생성하였으며 색깔에 다른 의미를 부여하였다. 아래는 본 연구에서 구분한 색깔 별 내용이다. 그림 3, 4는 생성된 피처 모델의 일부를 보여준다.

파란색 - 고정적으로 이행 가능
노란색 - 가변적으로 이행 가능
빨간색 - 이행 불가능
초록색 - 이행 증명을 위해 함께 사용

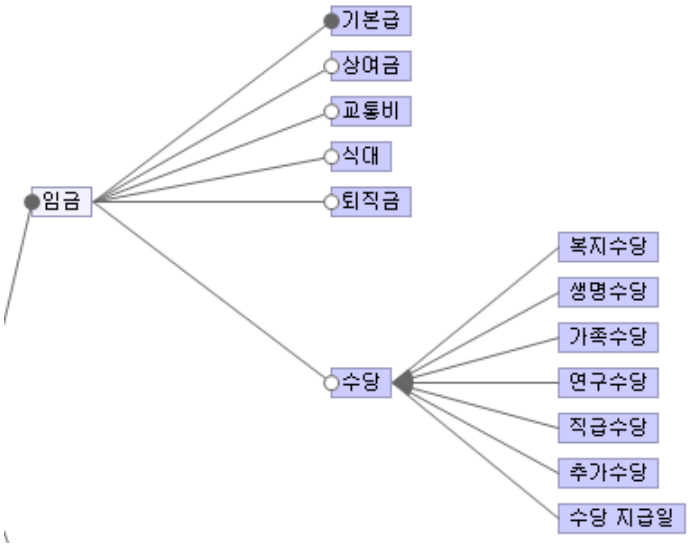


그림 3 고정적으로 이행 가능한 피처(파란색)

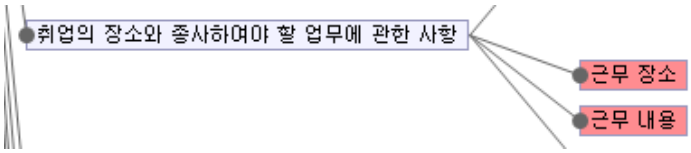


그림 4 이행 불가능한 피처(빨간색)

3.2.4 스마트 컨트랙트 코드 구현

스마트 컨트랙트는 블록체인상에서 동작하고 부분적으로 기능을 원격에서 호출하여 동작하는 프로그램이기 때문에 SPL의 적용 방식과 차이가 있게 구성된다. 스마트 컨트랙트는 근로계약서 템플릿의 수에 비례하여 배포되어야 하며, 근로계약서 관련 코드가 모두 포함(Full code)되어 배포되도록 구성한다. 배포된 스마트 컨트랙트는 로드 타임 바인딩(Load-Time Binding)의 형태로 블록체인 배포 시에 바인딩된다.

이와 같은 형태를 취한 이유는 블록체인상에 배포된 스마트 컨트랙트는 원격에서 호출하는 방식이고 블록체인은 여러 노드가 함께 공유하여 구성되는 네트워크로 저장소와 성능에 영향을 줄 수 있어 이를 최소화하기 위해 위와 같이 구성하였다.

프레임워크에서 사용자는 피처 선택을 하고 템플릿을 제공받는다. 블록체인에 배포된 스마트 컨트랙트를 템플릿과 바인딩하기 위해 우리는 그림 5와 같이 템플릿과 스마트 컨트랙트에 동일한 ID를 부여하고 템플릿에 맞는 스마트 컨트랙트가 연결될 수 있도록 구성하였다. 또한 템플릿이 선택되고 계약 내용 입력이 완료된 근로계약서는 사용자에 의해 승인되고 배포될 때 고유 ID 값을 부여받고 해당 값을 통해 함수를 호출하여 블록체인에서 활용된다.

3.3 프레임워크 활용 실행 가능한 리카르디안 컨트랙트 생성

본 장에서는 3.2 절에서 생성한 피처 모델이 적용된 프레임워크를 활용하여 근로계약서의 실행 가능한 리카르디안 컨트랙트 생성에 대해 논한다. 본 과정은 그림 2와 같이 5단계로 이루어져 있으며 각 단계별 상세 내용은 아래와 같다.

1. 피처 선택(Feature Selection) – 사용자에게 피처에 해당하는 계약서 항목 제공
2. 템플릿 제공 및 내용 입력 – 피처 선택에 따른 계약서 템플릿을 제공하고 사용자는 내용 입력
3. 계약서 내용 확인 및 전자서명 – 타 계약인 계약서 내용 확인 및 전자 서명
4. 실행 가능한 리카르디안 컨트랙트 생성 – 계약당사자 모두에게 최종 승인을 받은 뒤 실행 가능한 리카르디안 컨트랙트 생성
5. 블록체인 배포 – 생성된 실행 가능한 리카르디안 컨트랙트는 암호화 및 트랜잭션 과정을 거쳐 블록체인 상에 배포. 사용자는 블록체인에 배포된 스마트 컨트랙트를 활용할 수 있으며, 리카르디안 컨트랙트 조회 가능

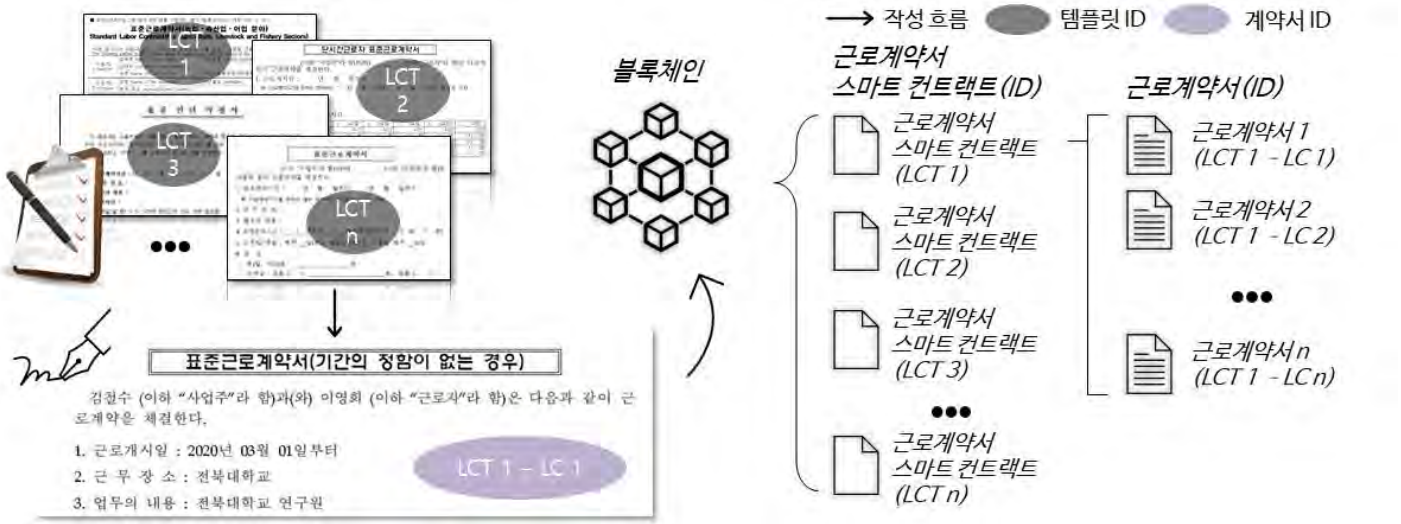


그림 5 근로계약서 작성시 템플릿과 스마트 컨트랙트 바인딩

제4장 실험 및 평가

4장에서는 프레임워크에 대한 검증과 평가를 진행한다. 본 연구에서는 GQM 방식을 적용하여 성능 측정을 위한 지표를 뽑아냈다. 더불어 프레임워크를 웹페이지의 형태로 프로토타입을 구현하여 실험을 진행하였다.

4.1 실험 개요

본 연구에서는 표준근로계약서를 포함한 총 31개의 근로계약서를 실험에 활용하였으며, 프레임워크를 통해 실행 가능한 리카르디안 컨트랙트를 생성하여 근로계약서로써 활용될 수 있도록 기준을 충족하고 계약 이행이 가능하도록 계약서가 실행 가능한지를 실험하였다.

4.1.1 GQM(Goal Question Metrics)

본 연구에서는 실행 가능한 리카르디안 컨트랙트 생성 프레임워크를 제안하였으며, 목표는 첫 번째로 다양한 계약서를 생성하고 두 번째로 계약 사실 증명뿐만 아니라 계약 이행 증명도 가능한 계약서를 생성하고자 한다. 선정한 목표에 하위 목표로 4개가 있으며, 4가지의 질문과 4가지의 지표를 활용하여 검증 및 평가를 진행했다.

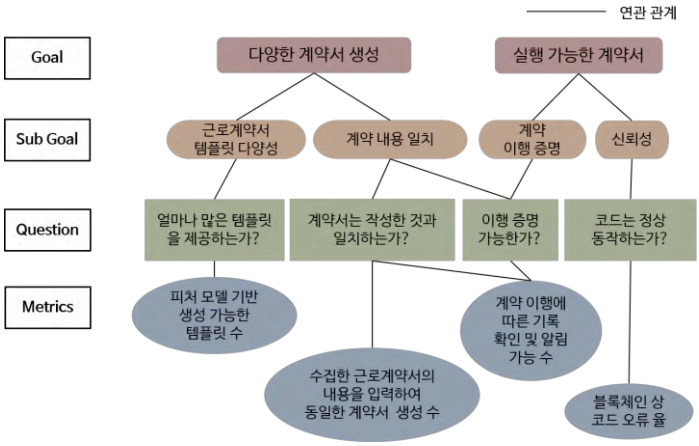


그림 6 프레임워크 평가를 위한 GQM

4.1.2 프레임워크 프로토타입

프레임워크의 프로토타입을 구현하여 검증하고자 하였으며, SPL을 적용하기에 적합한 Java를 활용할 수 있는 블록체인을 조사하여 선정하였다. 하이퍼레저 패브릭의 체인코드를 활용하여 스마트 컨트랙트를 구현하였으며, 내부에는 스프링 프레임워크를 활용하고 디자인 패턴과 프레임워크, 파라미터 방식 등을 혼재하여 적용하였다. 옵저버 패턴, 빌더 패턴, 템플릿 메소드, 스트리티지 패턴 등이 활용되었다. 또한 화이트 박스, 블랙 박스 프레임워크 방법을 혼재하였다. 개발 환경으로는 IBM 블록체인 플랫폼[19]을 활용하여 블록체인 네트워크를 구성하였고, 프레임워크에서 하이퍼레저 SDK를 활용하여 실행 가능한 리카르디안 컨트랙트를 생성하고 블록체인에 배포한다. 계약서 생성은 3.3절에서 언급한 바와 동일하게 진행된다.

그림 7은 프레임워크 프로토타입으로 웹페이지 왼쪽에는 사용자가 피처를 선택할 수 있는 근로계약서 항목이 제공되며, 피처 선택에 따라 웹페이지의 오른쪽 근로계약서 템플릿이 변경된다. 사용자는 제공된 템플릿에 계약 내용을 입력하고 해당 내용을 기반으로 실행 가능한 리카르디안 컨트랙트를 생성할 수 있다.

스마트 컨트랙트는 스프링 프레임워크(Spring Framwork)를 기반으로 제작되었으며, 로드 타임 바인딩 방식으로 처음 배포되었을 때 한 번 환경 설정을 한 후 유지한다. 스프링 프레임워크의 XML을 활용하여 템플릿별 구성을 완성하며, 계약서의 파라미터에 따라 로직이 수행되도록 구성하였다.

4.2 실험 결과

우리는 실행 가능한 리카르디안 컨트랙트 생성이 올바르게 이루어졌는지 평가하기 위해 GQM 방법으로 도출된 4가지 질문에 대해 답변하고 평가를 진행하였다. 질문에 답변하기 위해 우리는 4가지의 메트릭을 위한 시험을 진행하였다. 목표와 질문을 함께 포함하여 표 4와 같이 정리하였으며 메트릭 별 결과를 함께 작성하였다.

4.3 실험 평가

피처 모델에서 선택할 수 있는 피처 수는 총 24개이며, 제약사항을 고려하여 이론상 만들 수 있는 피처의 수는 6,389,760개이다. 템플릿 수 계산한 식은

표 4 실험 결과

Goal	Sub Goal	Question	Metric	Result
계약서 생성	근로계약서 템플릿 다양성	얼마나 많은 템플릿을 제공하는가?	생성 가능한 템플릿 수	1,048,576 /6,389,760 (16.41%)
	계약 내용 일치	계약서는 작성한 것과 일치하는가?	수집한 근로계약서의 내용을 입력하여 동일한 계약서 생성 수	6/8 (75%)
실행 가능	계약 이행 증명	이행 증명 가능한가?	계약 이행에 따른 기록 확인 및 알림	50%
	신뢰성	코드는 정상 동작하는가?	블록체인상 코드 오류율	0%

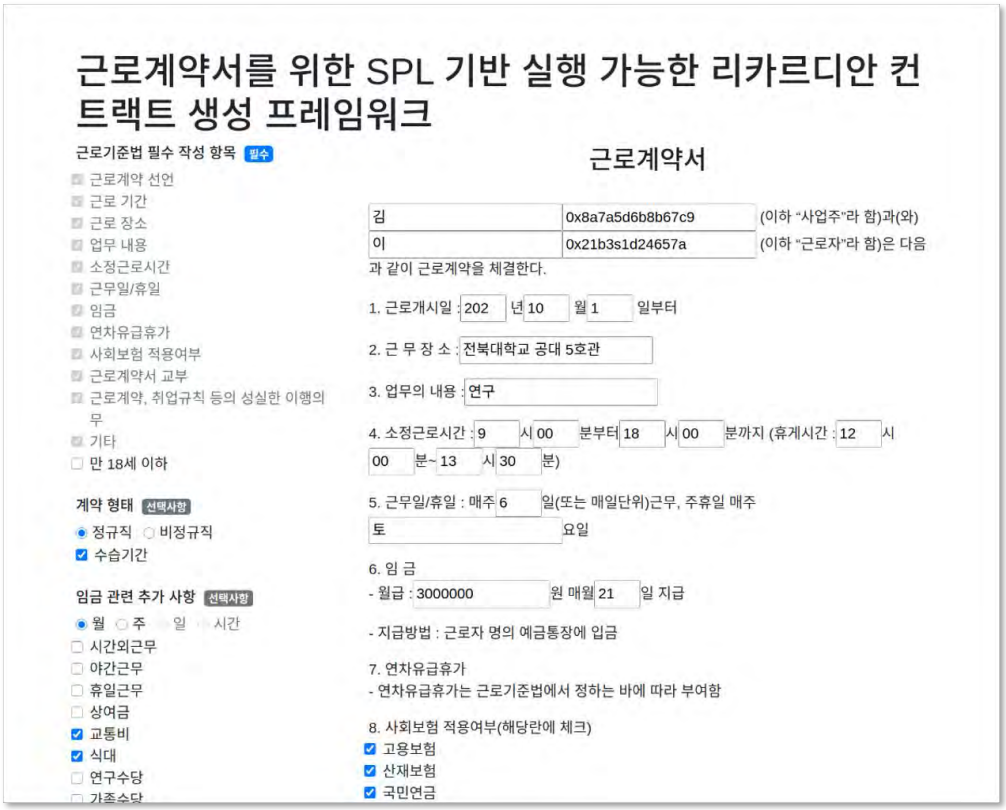


그림 7 근로계약서 도메인에 적용한 프레임워크 프로토타입

식 1과 같다. n은 피처의 Optinal에 해당하는 피처수가 입력되며 알파값에는 피처의 alternative 요소와 제약사항에 따라 달라지는 값들이 들어간다. 예를 들어 임금 주기, 국적, 나이 등에 따라 알파값은 달라질 수 있다.

$$total\,templates = 2^n \times \alpha$$

수식 1 생성 가능한 근로계약서 템플릿 수 계산식

실제로 구현된 프레임워크의 프로토타입을 통해 구현할 수 있는 근로계약서의 수는 1,048,576개로 전체 대비 약 16.41%를 생성할 수 있다. 우리는 월급을 받는 정규직과 계약직에 한해서 발생할 수 있는 다양한 계약서 템플릿을 대부분 충족할 수 있다. 하지만 일용직, 아르바이트, 파견직, 외국인, 특수고용직을 위한 계약서 등의 조건은 현재 구현 정도로는 충족하지 못한다. 향후 연구에서 추가적인 개발로 보완할 수 있는 부분이라고 판단하고 있다.

두 번째로 계약서는 민감한 개인정보로 실제로 임금과 근로 조건이 작성된 계약서는 수집하기 어려운 점이 있었다. 내용이 채워져 있는 계약서의 수는 8개였으며 작성된 계약서의 내용을 프레임워크에 동일하게 입력하여 일치하는 계약서가 생성되는지 확인하는 실험을 진행하였다. 실험은 내용이 일치하는지 연구자가 수작업으로 비교분석 하였다. 결과로는 8개 중 6개를 완벽하게는 아니지만, 근로계약서의 형태로 대부분 반영할 수 있었다. 세부적인 항목을 표현하는데 한계가 있는 것을 확인하였다. 예를 들어 "휴일/휴가" 항목의 휴일과 휴가 지정은 표준근로계약서에 명시되어 틀은 비슷하지만 한 기업의 근로계약서에는 "2) 연차휴가 등에 대하여는 관계법령에 따르되 구체적인 내용은 취업규칙에 정하는 바에 의한다" 로 세부 항목이 추가되었고 기업의 세부 규정은 충족하지 못했다.

세 번째로 계약 이행 증명은 하이퍼레저 패브릭의 체인코드 로그를 통해 확인해 볼 수 있었다. 체인코드의 로그는 계약의 이행 내역을 나타내며 본 연구에서 진행한 실험의 경우 단순하게 계약 이행이 된 임금 지불 내역을 로그의 형태로만 출력한다. 하지만 현실에 실제화하여 적용하게 된다면 비트코인과 같은 가상화폐 또는 은행과 연계하여 실제 화폐를 지급하는 방식으로 블록체인상에 이행 기록을 남기고 계약 분쟁 발생 시 이행 증명을 증거 자료로 활용될 수 있다.

마지막으로 블록체인상에서 오류 없이 동작하는지를 파악하였다. 현재 우리가 제작한 스마트 컨트랙트는 8개이며 모든 스마트 컨트랙트 기능을 호출하여 실험을 진행하였으며 실험 결과 일반적인 사용에서는 오류가 없는 것으로 확인하였다.

5. 토의

5장에서는 본 연구에서 더 논의되어야 하는 부분과 향후 연구에 필요한 내용에 대해 논한다.

5.1 피처 모델 완성도

SPL은 피처 모델의 완성도에 따라 생산되는 소프트웨어의 품질이 달라질 수 있다. 본 연구에서 작성한 피처 모델의 경우 총 31개의 근로계약서를 수집 및 분석하여 작성된 내용이며, 다양하게 존재하는 근로계약서를 모두 담지 못했다. 따라서 피처 모델의 완성도가 높지 않아 실행 가능한 리카르디안 컨트랙트의 품질이 낮을 수 있다.

5.2 프레임워크 적용

우리는 Java 프로그래밍이 가능한 하이퍼레저 패브릭 체인코드만을 대상으로 연구를 진행하였다. Java를 활용할 경우 SPL을 만들기 위해 객체지향, 디자인 패턴 등을 손쉽게 적용할 수 있다. 하지만 하이퍼레저 패브릭을 제외한 타 블록체인 플랫폼의 경우 언어의 한계로 적용이 불가능할 수 있다.

5.3 SPL 차용

우리는 SPL을 차용하여 실행 가능한 리카르디안 컨트랙트 생성 프레임워크에 접목하였다. SPL 기반의 소프트웨어 생산과 본 연구에서 제안하는 방법은 기존 개념과 달라 적합하지 않은 부분이 있을 수 있다

5.4 GQM

GQM 방식으로 도출한 목표와 질문, 메트릭이 프레임워크를 평가하는데 부족한 지표일 수 있다. 관련 연구에서 제안하는 평가 지표와 본 연구에서 중요도가 높다고 판단되는 지표를 넣었지만 많은 연구가 진행되지 못한 상황에서 지표가 적절하게 설정되었다고 판단하기 어렵다.

5.5 법

블록체인 기반의 근로계약서가 법적 효력을 얻기 위해서는 근로기준법뿐만 아니라 전자문서 및 전자거래 기본법, 전자서명법, 최저임금법 등 다양한 법령을 준수하여야 한다. 본 연구에서는 모든 법을 충족시키지 못할 수 있으며, 실제 계약서로 활용되기위해서는 엄격한 피처모델 생성이 필요하다.

6. 관련 연구

6장에서는 리카르디안 컨트랙트와 관련된 사례와 연구에 대해 살펴본다.

6.1 스마트 컨트랙트 템플릿

스마트 컨트랙트 템플릿은 법적 효력이 적용되는 스마트 컨트랙트를 제안한다. 개요는 그림 8과 같다. 먼저 스마트 컨트랙트 템플릿에는 작성하고자 하는 계약서의 항목을 제공하고 사용자는 산문으로 내용을 작성한다. 작성된 계약 내용은 의미 분석(Semantic Analysis)[20] 방법을 활용하여 파라미터를 추출한다. 파라미터는 스마트 컨트랙트 코드에서 사용되며 파라미터 내용에 따라 동작하는 로직이 달라진다. 스마트 리갈 컨트랙트는 작성된 산문 정보를 XML 또는 JSON, FpML(Financial products Markup Language) [21]와 같은 형태로 만들어 계약 사실 증명을 위해 사용된다. 해당 데이터는 계약당사자들이 읽고 이해할 수 있는 문서이기 때문에 법적 효력을 가질 수 있다.

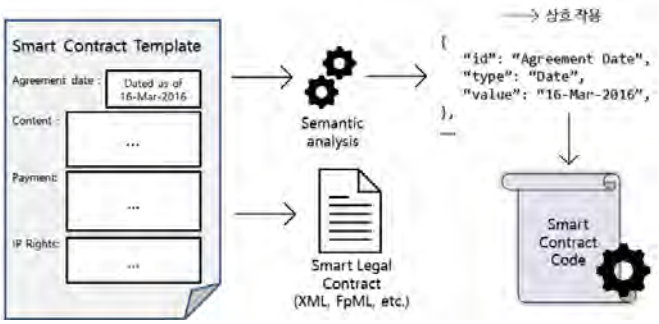


그림 8 스마트 컨트랙트 템플릿 개요

6.2 이오스

이오스는 블록체인 기반의 가상화폐 플랫폼 중 하나이다. 이안 그릭이 함께 참여하여 플랫폼 내의 계약에 리카르디안 컨트랙트를 도입하고자 연구개발을 진행하고 있다. 현재 플랫폼 내의 계약을 진행할 때 활용할 수 있으며, 이오스 자체적으로 지정한 헌법과 현실에서의 법을 적용할 수 있도록 하고 있다. 또한 스마트 컨트랙트도 연계되어 계약에 맞는 자동화된 처리를 가능하도록 구성하였다.

6.3 오픈바자

오픈바자는 암호화폐를 기반으로 하는 탈중앙화 마켓이다. 오픈바자에서 진행되는 거래들은 최종적으로 리카르디안 컨트랙트에 담겨 거래 내역을 남긴다.

6.4 인스트싸인(Instsign)

인스트싸인은 블록체인 관련 회사인 블로코(Blocko)와 법무법인 지석이 함께 협업을 통해 서비스하는 블록체인 기반 비대면 전자계약 서비스이다. 스마트 컨트랙트와 함께 사용할 수 있는 법적 근거와 효력을 갖출 수 있는 리카르디안 컨트랙트의 상용화를 목표로 한다.

6.5 관련 연구 비교분석

우리는 본 연구와 관련 연구를 5점 척도를 활용하여 앞에서 진행한 목표를 달성 가능 여부를 비교분석을 해보았다. 매우 적합하다면 5점 매우 부적합하다면 1점을 부여하였다. 표 5는 관련 연구 4개와 본 연구를 비교하여 정리한 표이다.

본 연구는 수작업 또는 단일 계약서만 지원하는 타 연구에 비해 본 연구는 다양한 계약서 지원과 계약서의 재사용성에서 매우 적합하다고 평가한다. 이는 SPL의 장점이 온전히 반영될 수 있기 때문이라고 생각한다. 또한 스마트 컨트랙트를 지원하지 않는 경우를 제외하고 코드의 품질을 높일 수 있는 접근방법이라고 생각하고 계약의 유효성의 경우 스마트 컨트랙트 템플릿의 수작업 방식에 비해 모자랄 수 있지만 피처 모델의 완성도를 높힌다면 세부적인 요소는 충족하지 못하겠지만 전체적인 틀은 만족할 수 있는 계약서를 생성할 수 있어 적합한 방법이라고 판단한다.

표 5 관련 연구 비교분석 표

구분	스마트 컨트랙트 템플릿	이오스	오픈 바자	인스트 싸인	본 연구
다양한 계약서 지원	1	1	1	3	5
계약서 재사용성	3	1	1	1	5
파라미터 추출 자동화	1(5)	5	5	1	5
파라미터 정확도	5	5	5	5	5
코드 오류	4	3	x	x	5
계약 유효성	5	3	x	x	4
1점: 매우 부적합 2점: 약간 부적합 3점: 보통 4점: 약간 적합 5점: 매우 적합 ( ): 인공지능 의미분석 방법 적용시 x: 스마트 컨트랙트 미사용					

7. 결론

본 논문은 근로계약서를 위한 SPL 기반 실행 가능한 리카르디안 컨트랙트 생성 프레임워크를 제안하였다. 우리는 총 31개의 근로계약서를 수집 및 분석하였고 피처 모델을 생성하였다. SPL을 차용한 프레임워크의 고려사항을 알아봤다. 실험 및 평가는 GQM 방법을 기반으로 지표를 선정하고 프로토타입을 구현하여 실험을 진행하였다. 총 1,048,576개(전체의 16.41%)의 템플릿을 제공할 수 있는 것으로 계산되며 실제 계약서 8개 중 6개(75%)를 동일하게 실행 가능한 리카르디안 컨트랙트로 생성하는 것을 확인하였다. 더불어 블록체인상에 배포된 8개의 스마트 컨트랙트는 모두 오류 없이 동작하는 것을 확인하였고 계약 이행에 대한 로그를 보고 이행 증명이 가능하였다. 본 연구를 통해 실행 가능한 리카르디안 컨트랙트를 생성할 수 있었으며, 사용자에게 다양한 계약서 템플릿을 제공하고 계약서의 재사용성을 높일 수 있었다. 프레임워크는 일반적인 근로계약서 작성에 적용될 수 있으며, 블록체인을 활용하여 데이터 무결성과 스마트 컨트랙트의 자동화된 이행을 통해 디지털 계약서의 활성화와 분쟁 요소 감소를 기대하고 있다.

8. Acknowledgement

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음" (IITP-2020-2017-0-01628*)

9. 참고 문헌

[1] Grigg, I. (2004, July). The ricardian contract. In Proceedings. First IEEE International Workshop on Electronic Contracting, 2004. (pp. 25-31). IEEE.

[2] Nofer, M., Gomber, P., Hinz, O., & Schiereck, D. (2017). Blockchain. Business & Information Systems Engineering, 59(3), 183-187.

[3] Satyananda, T. K., Lee, D., Kang, S., & Hashmi, S. I. (2007, August). Identifying traceability between feature model and software architecture in software product line using formal concept analysis. In 2007 International Conference on Computational Science and its Applications (ICCSA 2007) (pp. 380-388). IEEE.

[4] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. Feature-Oriented Software Product Lines. Springer, 2013.

[5] 통계청(2020), 「2020년 2사분기 임금근로 일자리동향 보도자료」

[6] 근로기준법, 법률 제17326호(2020)

[7] 고용노동부(2020), 「e-고용노동지표」, http://eboard.moel.go.kr/(2020. 11. 16.)

[8] Caldiera, V. R. B. G., & Rombach, H. D. (1994). The goal question metric approach. Encyclopedia of software engineering, 528-532.

[9] Xu, B., Luthra, D., Cole, Z., & Blakely, N. (2018). EOS: An architectural, performance, and economic analysis. Retrieved June, 11, 2019.

[10] OpenBazzar, https://openbazaar.org(2020. 11. 16.)

[11] Clack, C. D., Bakshi, V. A., & Braine, L. (2016). Smart contract templates: foundations, design landscape and research directions. arXiv preprint arXiv:1608.00771.

[12] 고용노동부(2020),「체불임금해결방법」, https://minwon.moel.go.kr/minwon2008/info/info_faq_typeA.do(2020. 10. 8.)

[13] Nakamoto, S. (2019). Bitcoin: A peer-to-peer electronic cash system. Manubot.

[14] Dannen, C. (2017). Introducing Ethereum and solidity (Vol. 1). Berkeley: Apress.

[15] Cachin, C. (2016, July). Architecture of the hyperledger blockchain fabric. In Workshop on distributed cryptocurrencies and consensus ledgers (Vol. 310, No. 4).

[16] Szabo, N. (1997). Formalizing and securing relationships on public networks. First Monday.

[17] Slock it, https://blog.slock.it/slock-it-decentralizing-the-emerging-sharing-economy-cf19ce09b957(2020. 05. 16.)

[18] Scheer, A. W. (2000). ARIS—business process modeling. Springer Science & Business Media.

[19] IBM(2020), 「IBM Blockchain Platform」, https://www.ibm.com/kr-ko/blockchain/platform(2020. 06. 05.)

[20] Goddard, C. (2011). Semantic analysis: A practical introduction. Oxford University Press.

[21] FpML at a Glance, https://www.fpml.org/about/(2020. 11. 16.)

다중 사용자 모바일 엣지 클라우드 환경에서 강화학습을 사용한 효율적 서비스 마이그레이션 방법

신준규 ○, 고인영
한국과학기술원
shinjunkyu@kaist.ac.kr, iko@kaist.ac.kr

Efficient Service Migration Using Reinforcement Learning in Multi-user Mobile Edge Clouds

Jun-Kyu Shin○, In-Young Ko
Korea Advanced Institute of Science and Technology

요 약

모바일 엣지 클라우드(Mobile Edge Cloud, MEC)는 근거리에서 서비스를 제공함으로써 사용자가 낮은 통신지연시간의 서비스를 이용할 수 있게 한다. 하지만 사용자의 위치 이동 때문에 연결되어있는 MEC와의 거리가 멀어지게 되어 낮은 통신지연시간의 서비스를 이용할 수 없게 되는 경우에는 사용자와 근접한 곳에 있는 MEC로 새롭게 연결하여 낮은 통신지연시간의 서비스를 유지할 수 있다. 이때 새롭게 연결할 MEC의 상태에 따라 서비스 마이그레이션이 필요한 경우가 발생하며, 품질 저하 없이 서비스를 받기 위해서는 사용자의 위치를 예측하여 미리 서비스 마이그레이션을 수행해야 한다. 하지만 사용자의 미래 위치를 예측하기 위해서는 많은 양의 개인정보가 필요하다는 문제가 있다. 또한, 사용자의 위치를 예측하는 것 외에도 다중 사용자 환경에서 특정 MEC가 과부하 되는 현상을 막아야 하는 문제도 있다. 본 연구에서는 다중 사용자 환경에서 사용자와 MEC 사이의 거리, MEC의 가용 자원 상태 정보만을 활용하여 사용자의 위치와 MEC의 가용 자원을 예측함으로써 효율적인 서비스 마이그레이션이 가능하도록 하는 방법을 제안한다. 또한, 실제 환경에서 수집된 위치 데이터를 활용한 실험을 통해 본 연구에서 제시하는 모델이 효율적으로 동작함을 확인하였다.

1. 서 론

모바일 엣지 클라우드(Mobile Edge Cloud, MEC)는 사용자와 가까운 곳에 있는 엣지 서버에서 서비스를 제공함으로써 통신 지연시간(Latency)을 줄여 서비스의 품질을 높이는 기술이다[1-3].

그림 1과 같이 MEC 1로부터 낮은 통신 지연시간(Low Latency)의 서비스1을 받고 있던 사용자가 MEC1로부터 멀어져 MEC2와 가까워지게 될 경우, 사용자는 MEC2로부터 서비스를 받아야만 낮은 통신 지연시간의 서비스를 유지할 수 있다. 이때 MEC2에 서비스1이 없는 경우에는 MEC1에서 MEC2로 서비스의 이동이 필요하며, 이렇게 서비스 제공에 필요한 프로그램과 데이터를 MEC 서버 간에 이동하는 것을 *서비스 마이그레이션(Service Migration)*이라고 한다[2-5].

사용자가 새로운 MEC의 영역으로 이동한 이후 서비스 마이그레이션이 수행되는 경우에는 서비스 마이그레이션에 필요한 데이터가 전송되는 시간 동안 서비스 단절이 일어나거나 낮은 지연시간의 통신 품질을 유지하지 못한 상태로 서비스를 받게 된다. 이러한 문제가 발생하는 것을 막기 위해서는 사용자가 어느 곳으로 이동할지 예측하여 이동할 것으로 예상하는 곳과 가까운 MEC로 미리 서비스 마이그레이션을 수행해야 한다[4-6].

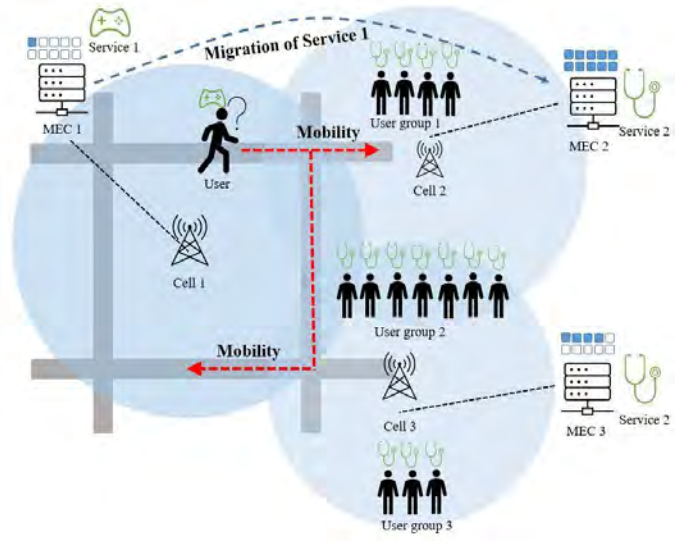


그림 1. MEC 환경에서의 서비스 마이그레이션 예제

이를 위해 MEC 환경에서 사용자의 이동 경로를 예측하여 서비스 마이그레이션을 사전에 수행함으로써 서비스의 중단 없이 낮은 통신 지연시간의 서비스를 제공하며, 경로 예측의 정확도를 높임으로써 불필요한 서비스 마이그레이션을 줄이기 위한 연구가 있었다[7]. 하지만 이 연구에서는 이동 경로 예측 모델을 만들기 위해 사용자의 위치정보 외에도 많은 종류의 개인정보 수집 및 처리를 필요로 한다[8].

이러한 단점을 보완하기 위해 이동 경로 예측 없이 서비스 마이그레이션의 효율을 높이려는 연구도 있었다[9]. 하지만 이러한 방법은 이동 경로 예측을 하는 모델보다 빈번한 서비스 마이그레이션이 발생으로 데이터 전송, MEC 자원 점유 등의 비용 소모가 더 발생하게 된다.

앞서 언급한 문제 외에도 기존의 연구들[7-9]은 공통적인 문제를 가지고 있는데, 첫째로는 다수의 사용자 또는 사용자 기기(User Equipment, UE)가 함께 움직이는 상황을 고려하지 않았다는 것이다. 실제 환경에서는 MEC의 가용자원이 제한되기 때문에 다수의 사용자가 밀집해 있거나 같은 지역으로 모이게 될 경우에는 특정 MEC의 과부하로 인한 서비스 품질 저하 현상이 발생할 수 있다. 그림 1에서 사용자 집단1과 사용자 집단2의 모든 사람이 MEC2로부터 서비스2를 받으려 할 경우에는 MEC2의 가용 자원을 초과하게 되는 상황이 발생하게 되며 사용자 집단2중 일부는 MEC3으로부터 서비스를 받는 것이 더 효율적임을 보여준다. 둘째로는 연구[8]에 따르면 사용자의 이동 경로를 예측하기 위해서는 사용자의 위치 정보 뿐만이 아니라 사용자가 특정 지역에 머무른 시간, 방문한 빈도 등 많은 양의 개인정보가 필요한데, 개인정보를 구할 수 없는 환경에서는 이동 경로 예측 모델을 만들기 어려운 문제가 발생할 수 있다는 것이다. 즉, 개인정보를 수집할 수 없는 환경에서도 최소의 정보만을 통해 이동 경로를 예측할 수 있도록 해야 한다.

이러한 문제점을 극복하기 위해 본 연구에서는 강화학습을 활용하여 다수의 사용자 또는 사용자 기기가 함께 움직이는 환경에서 효율적인 서비스 마이그레이션을 제공하는 모델을 제안한다. 이 모델은 사용자와 MEC간의 거리와 MEC의 가용 자원 상태만을 정보로 사용하며, 다수의 사용자 또는 사용자 기기가 존재하는 환경에서 하나의 MEC에 과부하가 발생하는 것을 방지할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 효율적인 서비스 마이그레이션 기법에 관한 관련 연구를 소개하며, 3장에서는 문제 해결을 위한 강화학습 기반의 접근 방법을, 4장에서는 해당 접근 방법을 통한 결과를 설명하고 평가한다. 그리고 5장에서는 결론 및 앞으로의 연구를 설명한다.

2. 관련 연구

2.1 효율적인 서비스 마이그레이션에 관한 연구

관련 연구 [10]에서는 3단계의 프레임워크를 설계하여 서비스 마이그레이션의 대상이 되는 부분을 나누고, 필요한 부분만 마이그레이션을 수행하여 전송되는 데이터의 양을 줄이는 방법으로 효율적인 서비스 마이그레이션 방법을 제안하였고, 연구 [11]에서는 압축 기법을 통해 전송되는 데이터의 크기를 줄여 서비스 마이그레이션의 효율을 향상하는 방법을 제안하였다. 또한 연구 [11]에서는 컨테이너 환경에서 OpenFace[12]라는 얼굴 인식 서비스를 마이그레이션 시키는 경우, 전송 시간이 26초 정도 걸리고, 전송하는 데이터의 크기도 2.17GB가 되는 것을 확인하여 서비스 마이그레이션에서의 효율성을 향상하는 것이 얼마나 중요한지를 보여주었다.

2.2 움직임 기반의 서비스 마이그레이션에 관한 연구

과거의 연구에서는 사용자가 이동하게 되어 새로운 MEC로 서비스 마이그레이션이 발생하게 될 때에도, 서비스 중단이나 품질 저하 없이 서비스를 제공받을 수 있도록 하는 것에 연구의 초점을 맞추었다.

Ksentini et al.은 Follow Me Cloud(FMC) 개념을 제안하고 데이터 전송, MEC의 자원 점유 등의 서비스 마이그레이션 비용과 사용자 관점에서의 서비스 품질(Quality of Service, QoS)간의 균형을 맞추는 것에 관해 연구를 하였다[13,14]. 이 연구에서는 최적의 서비스 마이그레이션 결정을 위해 Markov Decision Process(MDP) 기반의 모델을 제안하였다. 하지만 사용자의 이동 경로 예측 모델은 제안하지 않았다.

Nadembega et al.은 이동 경로 예측 모델에 기반한 서비스 마이그레이션 모델을 제안했다[7]. 이 연구는 높은 정확도로 사용자의 경로를 예측할 수 있었기 때문에 효율적인 서비스 마이그레이션을 제공할 수는 있었지만, 이동 경로 예측 모델을 만들기 위해 많은 양의 개인정보가 필요하였다[8]. 따라서 개인정보를 쉽게 구할 수 없는 환경에는 적용하기 어려우며, 지역의 변화에 따라 새로운 이동 경로 예측 모델을 만들어야 한다는 한계를 가졌다.

인공지능을 활용하여 MEC 환경에서 서비스 마이그레이션의 효율 향상을 시도한 연구도 있었다[9, 15]. 연구 [8]에서 이동 경로 예측 모델을 만들기 위해 많은 개인정보가 필요했던 단점을 보완하기 위해 C.Zhang et al.은 경로 예측 모델 없이 강화학습만을 사용하여 서비스 마이그레이션의 적절한 시점과 대상 MEC를 결정하도록 하였다[9]. 이 연구에서는 Deep Reinforcement Learning(DQN) 기반 알고리즘을 통해 서비스 마이그레이션 수행에 따른 보상을 공식화 했으며, 동적 프로그래밍 기반 알고리즘을 사용한 모델[5]와의 비교를 통해 자신이 제시한 모델의 성능이 더 좋음을 증명하였다. 하지만 이것은 실제 사용자의

이동 경로를 통해 학습한 모델이 아니라, 사용자 기기가 임의의 확률로 다른 영역으로 이동한다고 가정하여 만든 것이기 때문에 실제 사용자의 이동 경로 예측을 사용한 모델보다 좋은 성능을 보이지 못한다는 한계가 있었다. 또한, 기존의 연구들은 모두 다중 사용자 환경은 고려하지 않았기 때문에 실제적인 MEC 환경에 적용하기는 적합하지 않다[7,9,13,14].

3. 강화학습 기반의 효율적 서비스 마이그레이션 방법

이 절에서는 시스템 환경과 강화학습을 기반으로 한 MEC 예측 모델을 정의한다.

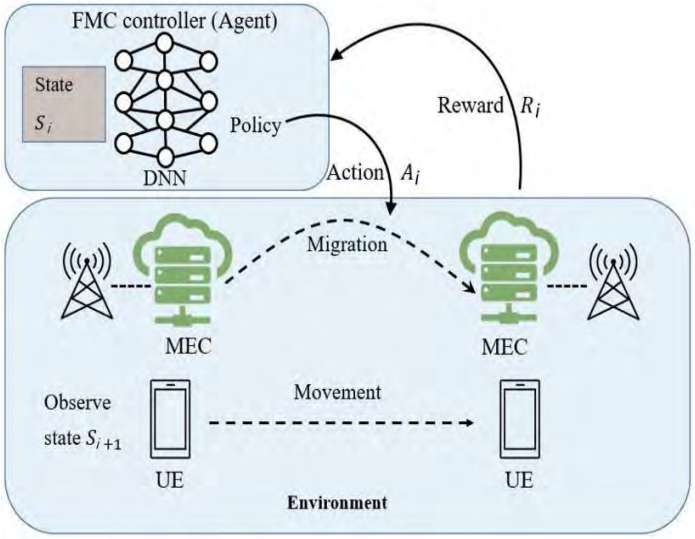


그림 2. MEC 환경을 위한 DQN 구조

3.1 다중 사용자 환경에서의 서비스 마이그레이션 환경 모델 정의

이 절에서 우리는 $U = \{u_1, u_2, \dots, u_p, \dots\}$로 표현되는 다수의 사용자 기기와 $M = \{m_1, m_2, \dots, m_q, \dots\}$로 표현되는 다수의 MEC가 있다고 가정한다. $T = \{t_1, t_2, \dots, t_i, \dots\}$의 시점 t_i에서 사용자 기기(u_p)와 MEC(m_q)의 거리는 $d_{p,q,i}$ 로 표기하며, 현재 연결된 MEC와의 거리는 $d_{p,connected_q,i}$로 표현한다. $d_{p,connected_q,i}$가 MEC의 낮은 지연시간의 서비스 제공 가능 범위(SR_m_q)를 벗어난 경우 인접한 MEC까지의 거리($d_{p,q,i}$)와 시점 t_i 에서의 MEC의 자원 상태($R\_m_{p,i}$)를 확인하여 사용자의 기대 품질에 만족하는 MEC로 서비스 마이그레이션을 수행한다. 또한, 사용자 기기가 움직임에도, 연결된 MEC의 낮은 지연시간 서비스 제공 범위 (SR_m_q)를 벗어나지 않을 것으로 예측될 경우, 현재 연결된 MEC보다 사용자 기기와 가까운 곳에 있는 MEC($\min.(d_{p,q,i})$)가 존재하더라도 서비스 마이그레이션에서 발생하는 비용(c_t)을 줄이기 위해 더 가까운 MEC로의 서비스 마이그레이션을 수행하지 않는다.

또한 MEC 환경에서 서비스 마이그레이션을 수행한 실제 관측 데이터가 존재하지 않기 때문에, 사전 학습 데이터가 존재하지 않더라도 실시간으로 보상(Reward)을 통한 학습이 쉬운 심층 강화학습을 적용한다. 실제 환경에 적용할 때에는 MEC와의 거리, MEC의 자원 상태 외에도 고려할 요소들이 많아질 수 있기 때문에 많은 관찰 공간을 요구하는 것에도 적합한 DQN을 적용한다.

표 1. 다중 사용자 환경에서의 서비스 마이그레이션 모델을 위한 표기법

표기	의미
U	사용자 기기의 집합 (u_p는 p번째 사용자 기기)
M	MEC(Mobile Edge Cloud)의 집합 (m_q는 q번째 MEC)
T	주어진 환경에서의 시간의 집합 (t_i는 i번째 시점)
$d_{p,q,i}$	시점 t_i에서 u_p와 m_q의 거리 ($d_{p,connected\_q,i}$ 는 u_p 가 m_q 와 연결됨을 의미)
SR_m_q	m_q 의 낮은 지연시간 서비스 제공 가능 거리
$R\_m_{q,i}$	시점 t_i에서 m_q의 자원 상태 (0~100%로 표현)
c_i	시점 t_i 에서 서비스 마이그레이션 수행으로 인해 발생된 비용
a_i	시점 t_i에서 FMC 컨트롤러의 액션 $a^0{}_i$: 서비스 마이그레이션을 수행하지 않음 $a^1{}_i$: 서비스 마이그레이션을 수행함
r_i	시점 t_i에서의 보상 값
$Overloaded\_cnt_i$	시점 t_i 에서의 과부하 된 MEC의 수
$Coeff\_C$	서비스 마이그레이션의 수행 비용에 대한 보상 값 가중치
$Coeff\_D$	UE와 연결된 MEC와의 거리에 대한 보상 값 가중치
$Coeff\_O$	과부하 된 MEC의 수에 대한 보상 값 가중치

그림 2에서 표현한 것과 같이 DQN의 에이전트는 FMC 컨트롤러로, 시점 t_i 에서 서비스 마이그레이션의 수행 여부를 결정하는 역할을 한다. FMC 컨트롤러는 Deep Neural Network(DNN)를 통해 학습하며 그림 2에서 사용자 기기(UE)가 연결되어있는 MEC의 지역을 벗어나 다른 MEC의 지역으로 이동하는 경우, 정해진 정책을 통해 마이그레이션의 수행 여부를 결정한다. 이때 서비스 마이그레이션의 수행 여부는 액션 A_i로 표현하며, 액션 $a^0{}_i$은 서비스 마이그레이션을 하지 않는 경우를, $a^1{}_i$은 서비스 마이그레이션을 하는 것을 의미한다.

시점 t_i에서의 액션에 따라 사용자 기기와 MEC와의 거리($d_{p,q,i}$), MEC의 가용자원상태($R\_m_{p,i}$), 과부하 된 MEC의 수($Overloaded\_cnt_t$)가 변경되거나 유지되며, 이때의 상태 s_i를 통해 보상 값(r_i)를 얻고, FMC 컨트롤러는 더 좋은 보상 값을 얻기 위한 방향으로 학습하게 된다.

해당 표기법은 표 1에서 기술되어 있으며, 각 시점 t_i에서의 사용자 기기와 MEC와의 거리($d_{p,q,i}$), 서비스 마이그레이션 수행으로 인해 소모한 비용(c_i), 과부하 상태의 MEC의 수($Overloaded\_cnt_i$)를 DQN의 관측 공간으로 하여 FMC 컨트롤러가 학습할 수 있도록 환경을 구성하였다.

3.2 다중 사용자 환경에서의 서비스 마이그레이션 모델을 위한 함수 정의

보상은 아래와 같은 수식(1)으로 표현되는데, 이는 서비스 마이그레이션 수행에 의해 발생하는 비용, 서비스를 제공하는 MEC와 사용자 기기와의 거리, 과부하 된 MEC의 수에 각각 다른 가중치를 부여하여 중요하게 여기는 요소를 변경할 수 있도록 하였다.

$$r_i = Coeff\_C * c_i + Coeff\_D * d_{p,connected_q,i} + Coeff\_O * Overloaded\_cnt_i \quad (1)$$

c_i 는 서비스 마이그레이션을 수행하는 경우 -1을, 서비스 마이그레이션을 수행하지 않는 경우 0의 값을 주어, 서비스 마이그레이션 수행으로 인해 발생하는 비용을 줄이고자 하였으며, $d_{p,connected_q,i}$ 가 SR_m_q 의 범위를 벗어나지 않는 경우는 0의 값을, 범위를 벗어나는 경우는 -1의 값을 주어 SR_m_q 의 범위 내에 있을 때는 더 가까운 MEC가 존재하더라도 불필요한 서비스 마이그레이션을 수행하지 않도록 하였다. $Overloaded\_cnt_i$ 는 $R\_m_{p,i}$ 가 과부하가 발생하여 100%를 넘어가게 될 경우, 과부하 된 MEC의 수만큼 $Overloaded\_cnt_i$ 값을 증가시켜 과부하 되는 MEC의 수를 줄일 수 있도록 했다.

3.3 다중 사용자 환경에서의 서비스 마이그레이션 알고리즘

그림 3에서는 심층 강화 학습 접근방법의 알고리즘을 설명한다. 1,2번째 줄에서는 DQN 에이전트의 경험 메모리 공간을 초기화하고 4번째 줄에서는 테스트 마다 환경을 초기화 한다. 3-6번째 줄에서는 정해진 시간 동안 다수의 사용자 기기가 이동하는 것을 표현했으며, 7-13번째 줄에서는 액션의 할당을, 14-18번째 줄에서는 각 액션에 의해 변경된 상태의 변화를 통해 r_{i+1}을 얻고 그에 따라 신경망을 학습시키는 것을 표현하였다.

Algorithm 1 Service Migration Algorithm in Multi-user

```
1: Initialize memory
2: Initialize network Q with random weight θ
3: while i = NumOfTest do
4:    Reset environment
5:    while t = StepCount do
6:       while n = NumberOfUE do
7:          Observe current state t_i
8:          p = get random value
9:          if p > ε then
10:            a_i = argmax(θ)
11:         else
12:            a_i = get random value
13:         end if
14:         Execute the action
15:         Observe the state s_{i+1}
16:         Observe the reward r_{i+1}
17:         Memory (s_i, a_i, r_i, s_{i+1})
18:         Train network Q
19:         n ← n + 1
20:      end while
21:      t ← t + 1
22:   end while
23:   i ← i + 1
24: end while
```

그림 3. 다중 사용자 환경에서의 서비스 마이그레이션 알고리즘

4. 평가 및 분석

본 절에서는 3절의 접근방법을 통해 도출된 경과를 평가하고 결과를 분석한다.

Unnamed: 0	TIMESTAMP	str_list
0	1372644253	[[-8.640792, 41.149026], [-8.640126, 41.146227...
1	1372651570	[[-8.609076, 41.162742], [-8.609094, 41.162751...
2	1372659148	[[-8.645283, 41.157387], [-8.645544, 41.157549...
3	1372663792	[[-8.598402, 41.161131], [-8.598474, 41.161113...
4	1372667190	[[-8.597034, 41.147469], [-8.597286, 41.147658...
...	...	...
14976	1404148175	[[-8.630064, 41.154174], [-8.630046, 41.154174...
14977	1404151433	[[-8.595405, 41.162283], [-8.59545, 41.162202]...
14978	1404148084	[[-8.647317, 41.176161], [-8.648181, 41.176845...
14979	1404166606	[[-8.619669, 41.137587], [-8.620542, 41.13756]
14980	1404151410	[[-8.628777, 41.169807], [-8.628615, 41.169807...

그림 4. Porto 도시에서의 택시 주행 기록

4.1 평가 데이터

평가 및 모델의 학습 데이터는 실제 환경에서 수집된 데이터를 사용하고자 하였다. 데이터는 포르투갈의 Porto라는 도시에서의 택시 운행 기록 데이터를 사용하였으며[16], 데이터의 구성은 그림 4와 같이 운행 시작 시각과, 25초 간격의 위도, 경도의 쌍으로 구성하였다. 또한, 동일한 환경에서의 비교를 위해 30분

동안의 주행 기록 내에서만 평가를 진행하도록 하였다.

해당 도시에서의 MEC 환경은 아직 구성되지 않았기 때문에 MEC의 위치나 성능 자료는 실제 자료를 수집할 수 없었다. 따라서 임의의 16개의 MEC를 도시의 중앙으로부터 동일한 간격으로 배치하였으며, 이는 단일 사용자 환경에서의 연구 [9]와 성능을 비교하기 위해 동일한 환경 설정을 따른 것이다.

표 2. 다중 사용자 환경에서의 서비스 마이그레이션 실험 환경 구성

소프트웨어	DQN 환경	Stable-baselines3
	Python	version 3.6.9
하드웨어	CPU	Intel® Xeon® CPU @ 2.00GHz (4v)
	RAM	27.3 GB
	GPU	Tesla V100-SXM2-16GB
설정 값	MEC	16개
	MEC의 서비스 제공 범위 (SR_m_q)	1.77 km
	다중 사용자 기기	20, 40, 60, 80, 100
	테스트 수행 수	100

4.2 실험 환경 및 조건

실험 환경은 표 1표 2와 같다. DQN 환경 구현을 위해 stable-baselines3를 사용했으며 Python 버전은 3.6.9를 사용하였다. 사용한 하드웨어는 Google Colab Pro에서 제공하는 하드웨어를 사용하였으며, 중앙처리장치는 Intel® Xeon® CPU @ 2.00 GHz (4v), RAM은 27.3 GB, 그리고 GPU는 Tesla V100-SXM2-16GB를 사용하였다.

연구 [9]에서 제시한 기본 조건과 동일하게 MEC의 수는 16개로 설정하였으며 SR_m_q 은 측정 시간 동안 사용자 기기가 평균 3회의 MEC의 영역을 이동할 수 있도록 1.77km로 설정하였다. 사용자 기기의 수는 20, 40, 60, 80, 100인 환경에서 실험하였는데, 이는 사용자 기기가 점유하는 MEC의 최소 자원을 1%라고 가정하면 MEC가 서비스를 제공할 수 있는 최대의 사용자 기기가 100이 되기 때문이다. 실험에 활용한 데이터 세트는 총 4,495개로, 모델 학습에 사용한 데이터와 분리하였으며 각 평가당 100번을 수행하여 평균치를 측정하였다.

4.3 단일 사용자 환경에서 과거 연구와의 성능비교

C.Zhang et al.은 자신이 제안한 모델이 기존의 동적 프로그래밍(Dynamic Programming) 기반의 연구 [13]과 서비스 마이그레이션을 수행하지 않는 모델보다 좋은 성능을 가졌음을 그림 5와 연구 [9]의 성능평가 섹션에서의 비교 자료를 통해 증명하였다[9]. 그림 5에서는 MEC가 12개 일 때 사용자 기기가 다른 영역으로 이동할 확률을 0~1로 변경시키며 다른 모델과의 성능을 비교하였다. 비교 결과로는 이동할 확률이 증가함에 따라 연구 [9]에서 공식화한 보상 값이 점점 증가하는 것을 확인하였으며, 이동할 확률이 100% 일 때를 제외하고는 다른 두 모델보다 높은 합산 보상 값을 가짐을 보였다. 연구 [9]에서의 보상에 대한 수식은 아래의 수식(2, 3)과 같다.

$$r_i(S_i, a_i) = q(S_i) = 36 - 18 * d_{p,connected_q,i} \quad if\ a^0{}_i \quad (2)$$

$$r_i(S_i, a_i) = D - C(S_i) = 36 - 10 * d_{p,connected_q,i} \quad if\ a^1{}_i \quad (3)$$

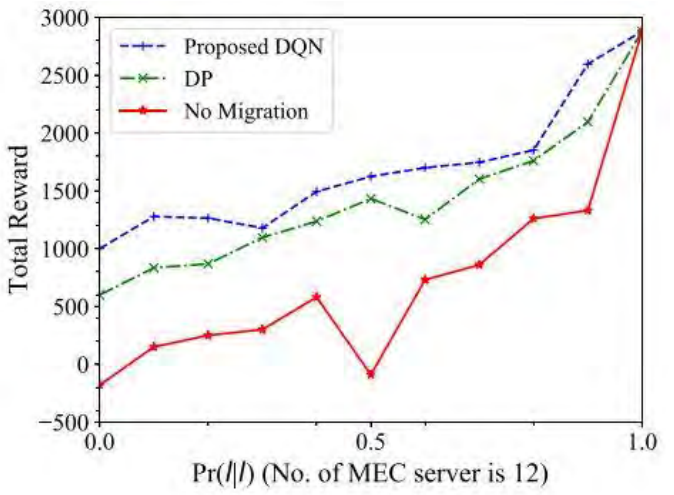


그림 5. 단일 사용자 환경에서 연구[4]의 제안 모델과 연구 [5]의 제안 모델, 그리고 서비스 마이그레이션을 하지 않는 모델 간의 성능 비교

그림 5를 통해 연구 [13]의 동적 프로그래밍 모델, 마이그레이션을 하지 않는 모델과 비교하여 성능이 더 좋음을 이미 보였으므로, 본 연구에서는 실제 환경에서 수집된 데이터를 활용한 모델이 연구[9]에서의 위치 예측을 하지 않는 모델보다 성능이 우수한지를 그림 6을 통해 비교하였다.

그림 6은 Porto 도시의 택시 운행 데이터 중 임의로 1,000개의 실험 데이터를 선정하여 30분 동안의 주행 중 발생한 서비스 마이그레이션의 수행 횟수를 비교하였으며, SR_m_q를 벗어나지 않는 조건에서 서비스 마이그레이션의 수행 수가 연구 [9]보다 적음을 확인하여, 이동 경로 데이터를 활용한 본 연구의 모델이 임의의 확률로 이동한다고 가정한 연구 [9]의 모델보다 효율적인 서비스 마이그레이션을 수행함을 확인하였다. 또한, 두 모델의 결과는 같다는 가정에서 유의 수준 0.05의 T-검정을 통해 P-value는 1.36e-19 임을 확인하여 가설을 기각하므로 두 모델을 통해 나온 결과가 유의미한 차이가 있음을 확인하였다.

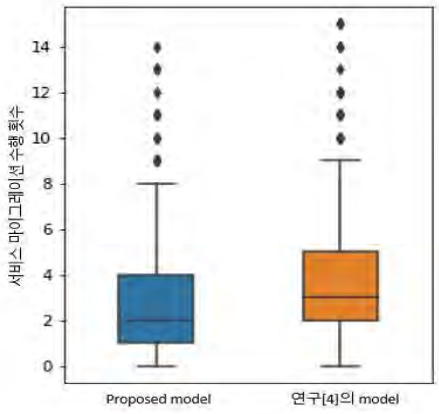


그림 6. 단일 사용자 환경에서 연구[9]의 모델과의 서비스 마이그레이션 수행 횟수 비교

4.4 다중 사용자 환경에서의 서비스 마이그레이션 효율성 비교 실험

그림 7, 그림 8, 그림 9에서는 사용자 기기에서 사용하는 서비스가 요구하는 MEC의 자원 점유율이 증가함에 따라 과부하 된 MEC, 사용자 기기와 연결된 MEC와의 거리, 마이그레이션 수행 횟수가 달라짐을 보여준다. 이번 실험에서는 사용자 기기가 20, 40, 60, 80, 100인 경우를 테스트하였으며 중복되는 결과를 나타내는 그래프는 표시하지 않았다.

DQN(0.8)은 $Coeff\_O$ (과부하 된 MEC의 수에 대한 보상 값 가중치)값이 0.8일 때의 DQN 기반 모델을, DDQN(0.8)은 $Coeff\_O$ 값이 0.8일 때의 DDQN 기반 모델을, DDQN(0.9)은 $Coeff\_O$ 값이 0.9일 때의 DDQN 기반 모델을, NoMig는 $R\_m_{q,i}$ (시점 t_i 에서 m_q 의 자원 상태), $d_{p,connected_q,i}$ (시점 t_i 에서 u_p 와 연결된 m_q 의 거리)와 상관없이 서비스 마이그레이션을 하지 않는 모델을, AllMig는 항상 주변의 가장 가까운 MEC로 마이그레이션을 하는 모델을 나타낸다.

그림 7에서는 사용자 기기에서 사용하는 서비스가 요구하는 MEC의 자원 점유율이 증가함에 따른 과부하 된 MEC의 개수의 변화를 그래프로 나타낸다. 사용자 기기의 수가 40개 이하일 때에는 사용자 기기당 요구하는 서비스 부하가 1%에서 4%로 증가함에도 불구하고 과부하 된 MEC가 관측되지 않지만, 사용자 기기가 60개로 다소 많아진 환경에서는 서비스 요구량이 3.5% 이상이 되는 경우, NoMig 모델과 AllMig 모델에서는 과부하 된 MEC의 수가 증가함을 보여준다. 사용자 기기가 80개 이상으로 많아진 경우에는 DQN(0.8), DDQN(0.8), DDQN(0.9) 모델 모두 과부하 되는 MEC의 수가 증가했지만, NoMig 모델이나 AllMig 모델만큼 과부하 된 MEC가 많이 관측되지는 않는 것을 확인하였다. 또한 그림 7의 실험 결과에서 ANOVA 테스트를 한 결과, 유의 수준 0.05에서 사용자 기기의 수가 100개인 경우에만 P-value가 0.021로, 5개의 모델이 유의한 차이가 있음을 확인했다.

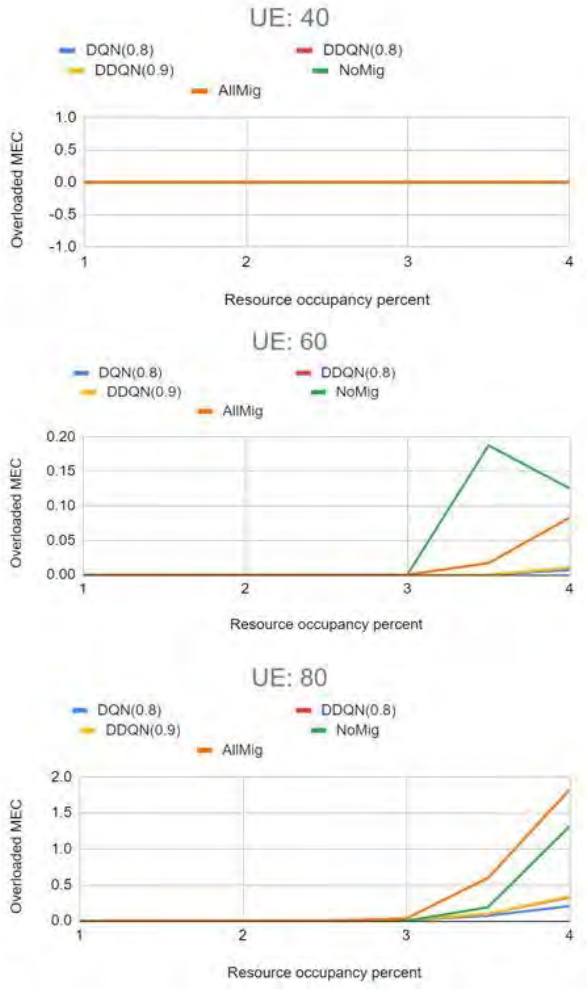


그림 7. 다중 사용자 환경에서 사용자 기기가 사용하는 서비스의 MEC 자원이 증가함에 따른 과부하 된 MEC의 수 비교

그림 8에서는 사용자 기기가 사용하는 서비스가 요구하는 MEC의 자원이 변함에 따라 $d_{p,connected\_q,i}/SR\_m_q$ (사용자 기기와 연결된 MEC와의 거리/MEC의 서비스 제공 가능 거리)가 변하는 것을 그래프로 보여준다. 이때 SR_m_q 는 모두 동일하여 모든 MEC가 동일한 서비스 제공 가능 범위를 가졌다는 것을 조건으로 한다. 사용자 기기가 60개 이하일 때에는 사용자 기기가 사용하는 서비스가 MEC 자원 요구량이 1%에서 4%로 증가함에도 불구하고 NoMig 모델을 제외한 다른 모델에서 $d_{p,connected\_q,i}/SR\_m_q$ 값이 1 이하로 유지되어, 멀어진 거리로 인한 서비스 품질의 저하는 관측할 수 없었다. 하지만 사용자 기기가 80개로 다소 많아진 환경에서 서비스의 자원 요구량이 2.5% 이상으로 증가하는 경우에는 $d_{p,connected\_q,i}/SR\_m_q$ 이 1에 가까워진 것을 볼 수 있었다. 또한, 사용자 기기가 100개이며 서비스의 자원 요구량이 4%일 때에는 AllMig 모델을 제외한 모든 모델의

$d_{p,connected\_q,i}/SR\_m_q$ 가 1을 초과하여, MEC의 서비스 제공 가능 범위를 벗어나게 되는 것을 알 수 있다. 그림 8의 결과에서 ANOVA 테스트를 한 결과 유의수준 0.05에서 사용자 기기의 수가 80개 이하인 경우는 P-value가 0.005 이하로 5개의 모델이 유의한 차이가 있음을 확인하였으나, 사용자 기기의 수가 100개인 경우는 P-value가 0.049로 5개의 모델이 유의한 차이가 있다는 것을 확인했지만 0.05에 매우 근사한 값으로 해당 결과가 큰 차이를 나타내는 것은 아님을 알 수 있었다.

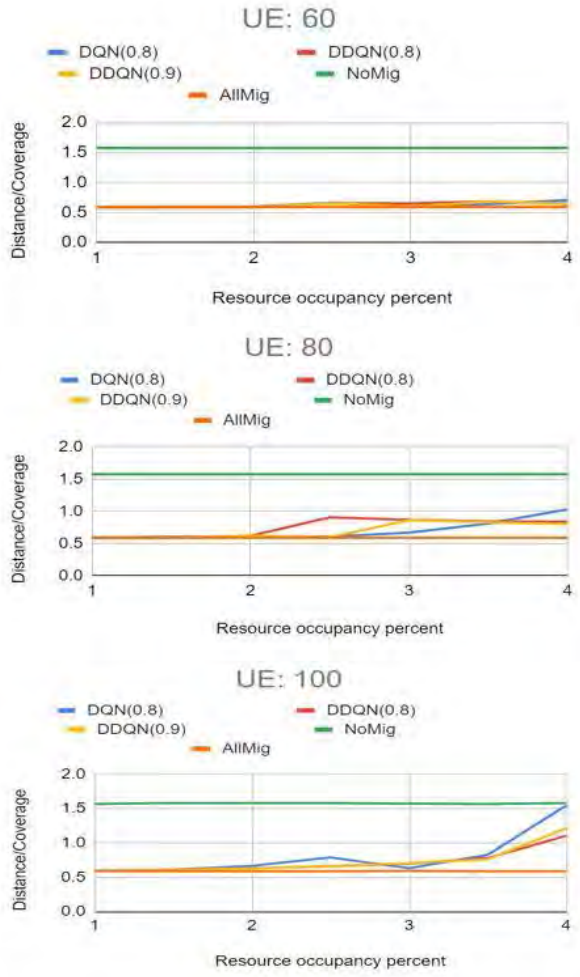


그림 8. 다중 사용자 환경에서 사용자 기기가 사용하는 서비스의 MEC 자원이 증가함에 따른 사용자 기기와 연결된 MEC와의 거리 비교

그림 9에서는 사용자 기기가 사용하는 서비스에서 요구하는 MEC의 자원이 증가함에 따른 서비스 마이그레이션 발생 수의 변화를 그래프로 나타낸다. 사용자 기기의 수가 60개 이하일 때에는 서비스의 MEC 부하가 1%에서 4%로 증가함에도 불구하고 NoMig 모델을 제외한 모든 모델에서 3~4회의 서비스 마이그레이션을 수행함을 확인할 수 있다. 하지만 사용자 기기의 수가 80개일 때, 사용자 기기가 사용하는 서비스의 자원 요구량이 2.5% 이상을 넘어가게 되는 경우에는 서비스 마이그레이션 수행 수가 줄어드는 것을 확인할 수 있다. 또한, 사용자 기기의 수가 100개 일 때에 서비스의 자원 요구량이 4% 일 때, 더 명확하게 서비스 마이그레이션의 수행 수가 줄어드는 것을 확인할 수 있었는데, 이는 사용자 기기와 근접한 MEC가 과부하 된 상태이기 때문에 서비스 마이그레이션을 수행하지 않는 것으로 해석할 수 있다. 이는 그림 8의 결과에서 사용자 기기와 연결된 MEC와의 거리가 증가하는 것을 통해서도 알 수 있다.

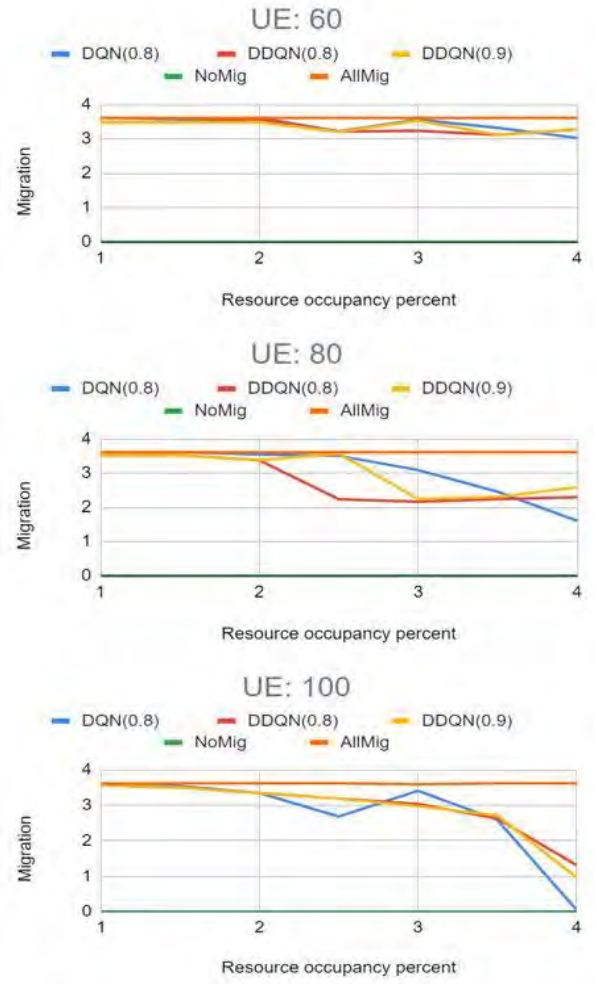


그림 9. 다중 사용자 환경에서 사용자 기기가 사용하는 서비스의 MEC 자원이 증가함에 따른 서비스 마이그레이션 수행 수 비교

그림 9의 결과에서 ANOVA 테스트를 한 결과, 유의수준 0.05에서 사용자 기기의 수가 80개 이하인 경우는 P-value가 모두 0.001 이하로, 5개의 모델이 유의한 차이가 있음을 확인하였으나, 사용자 기기의 수가 100개인 경우는 P-value가 0.032로 5개의 모델이 유의한 차이가 있다고는 말할 수 있지만, 0.05에 다소 근사한 값을 보여, 큰 차이를 보이지는 않는 것을 알 수 있었다. 이는 사용자 기기가 사용하는 서비스가 요구하는 MEC의 컴퓨팅 자원이 증가하는 것에 비해서, 전체 환경에서 지원하는 MEC의 수가 증가하지 않거나

MEC의 성능이 향상되지 않는다면 과부하 된 MEC가 발생하는 것을 막을 수 없으므로 정상적인 서비스를 제공할 수 없다는 것을 의미한다.

5. 결론

본 연구에서는 다중 사용자 환경에서 엣지 서버의 부하를 고려하였고, Proto 도시의 실제 택시 운행 정보를 기반으로 본 연구의 모델을 학습시키고 결과를 확인함을 통해서 보다 실질적인 환경에서 서비스 마이그레이션의 수행을 연구하고 검증하였다. 또한, 심층 강화 학습 기법을 적용하여 사전에 수집된 정보가 없는 환경에서도 본 모델을 적용하기가 쉽도록 하였으며, 사용자 기기와 MEC와의 거리 정보라는 최소한의 정보만을 통해 모델을 구현하여 개인정보가 민감한 환경에서 서비스 마이그레이션 모델을 구축할 수 있도록 하였다.

또한, 본 연구에서 제안한 모델의 적용을 통해 높은 품질의 서비스를 제공하면서도 서비스 마이그레이션의 수행 수를 줄여 데이터 전송으로 인해 발생하는 비용을 낮추고, 과부하가 발생하는 MEC의 수를 줄임으로써 보다 안정적인 서비스를 제공할 수 있음을 보였다.

이를 통해 5G 환경에서 초저지연(Ultra Low Latency)을 요구하는 자율주행차, 원격조종 드론 등의 서비스를 사용할 때, 사용자 관점에서는 서비스의 일시적 중단 또는 품질의 저하 없이 원활하게 서비스를 제공받으며, 서비스 제공자 관점에서는 불필요한 서비스 마이그레이션 비용을 줄일 수 있을 것을 기대한다.

향후 연구할 사항으로는 다양한 서비스를 사용하는 사용자와 다양한 성능을 가진 MEC가 존재하는 경우 효율적으로 서비스 마이그레이션을 할 수 있는 방법에 대한 연구가 있을 것이다.

Acknowledgement

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2019R1A2C1087430). 또한 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터 지원사업의 연구결과로 수행되었음(IITP-2020-2020-0-01795).

참고문헌

[1] Yun Chao Hu, et al. “Mobile Edge Computing A Key Technology towards 5G.” ETSI White Paper 11, ETSI, 2015, www.etsi.org/images/files/etsiwhitepapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf.

[2] Y. Yu, "Mobile edge computing towards 5G: Vision, recent progress, and open challenges." China Communications, vol. 13, no. 2, pp. 89-99, 2016.

[3] Reznik, Alex, et al. "Developing software for multi-access edge computing." ETSI White Paper 20, ETSI, 2017, https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp20_MEC_SoftwareDevelopment_FINAL.pdf.

[4] REJIBA, ZEINEB, XAVIER MASIP-BRUIN, and EVA MARÍN-TORDERA. “A Survey on Mobility-Induced Service Migration in the Fog, Edge, and Related Computing Paradigms.” ACM Computing Surveys, no. 5, September 2019.

[5] S. Wang, J. Xu, N. Zhang and Y. Liu. "A Survey on Service Migration in Mobile Edge Computing." IEEE Access, vol. 6, pp. 23511-23528, 2018.

[6] Wang, Shiqiang, et al. “Dynamic Service Migration in Mobile Edge Computing Based on Markov Decision Process.” ArXiv.org, 2019, arxiv.org/abs/1506.05261.

[7] A. Nadembega, A. S. Hafid and R. Brisebois. "Mobility prediction model-based service migration procedure for follow me cloud to support QoS and QoE." 2016 IEEE International Conference on Communications (ICC), pp. 1-6, 2016.

[8] A. Nadembega, A. Hafid and T. Taleb. "A Destination and Mobility Path Prediction Scheme for Mobile Networks." IEEE Transactions on Vehicular Technology, vol. 64, no. 6, pp. 2577-2590, June 2015.

[9] Cheng Zhang, Zixuan Zheng. “Task migration for mobile edge computing using deep reinforcement learning.” Future Generation Computer Systems, Volume 96, pp.111-118, 2019.

[10] K. Ha et al. “Adaptive VM Handoff Across Cloudlets.” School of Computer Science Carnegie Mellon University, June 2015, www.cs.cmu.edu/~satya/docdir/CMU-CS-15-113.pdf.

[11] Lele Ma College of William and Mary, et al. “Efficient Service Handoff across Edge Servers via Docker Container Migration.” Proceedings of the Second ACM/IEEE Symposium on Edge Computing, 1 Oct. 2017.

[12] Amos, Brandon, et al. “OpenFace: A General-Purpose Face Recognition Library with Mobile Applications.” School of Computer Science Carnegie Mellon University, June 2016, www.semanticscholar.org/paper/OpenFace:-A-general-purpose-face-recognition-with-Amos-Ludwiczuk/82e66c4832386cafcec16b92ac88088ffd1a1bc9.

[13] A. Ksentini, T. Taleb and M. Chen. "A Markov Decision Process-based service migration procedure for follow me cloud." 2014 IEEE

International Conference on Communications (ICC), pp. 1350-1354, 2014.
[14] T. Taleb, A. Ksentini and P. A. Frangoudis. "Follow-Me Cloud: When Cloud Services Follow Mobile Users." IEEE Transactions on Cloud Computing, vol. 7, no. 2, pp. 369-382, 1 April-June 2019.
[15] Cui, Chaoxiong, et al. “An LSTM-Method-Based Availability Prediction for Optimized Offloading in Mobile Edges.” Sensors, 2019.
[16] Cross, Chris. “Taxi Trajectory Data.” Kaggle, 12 Apr. 2018, www.kaggle.com/crailtap/taxi-trajectory.

데이터 변형이 딥러닝 모델의 안전에 미치는 영향 측정 연구

김한동 ○ 한종대
상명대학교 컴퓨터과학과
sw.engineering@kakao.com, elvenwhite@smu.ac.kr

An Experimental Study on Measuring Safety of Deep Learning Models by Data Augmentation

Handong Kim○ Jongdae Han
Dept. of Computer Science, Sangmyung University

요 약

본 연구에서는 DNN의 요구사항 중 정확성과 신뢰성을 데이터 변형을 통해서 검증할 수 있음을 보이려고 한다. 가설의 증명을 위해 CIFAR-10 데이터 셋을 기본으로 여러 데이터 변형을 가하여 변형 데이터 셋을 만들 데이터 변형을 가한 데이터 셋을 만든다. 이후 기본이 되는 데이터 셋을 활용하여 학습시킨 모델과 변형 데이터 셋을 활용하여 학습시킨 모델의 평가를 진행하여 정확성과 신뢰성을 얻어내는 실험을 진행하였다. 실험 결과 데이터 변형을 가한 데이터 셋을 활용한 모델의 정확성과 신뢰성이 그렇지 않은 모델에 비해 개선된 것을 확인할 수 있다. 이를 통해 데이터 변형이 DNN의 요구사항을 검증하는데 영향을 미칠 수 있다는 것을 확인할 수 있다.

1. 서 론

소프트웨어의 품질 검증을 위한 요소 중 하나는 요구사항(Requirement)이다. 요구사항은 소프트웨어 개발 생명주기에서 빠르게 검증될 수록 추후 발생할 수 있는 수정 비용을 감소시킬 수 있다는 특징을 가지고 있다.

최근 DNN(Deep Neural Network)을 필두로 한 AI 기술에 대한 많은 연구가 이루어짐에 따라 관련된 기술이 빠르게 발전하고 있다. 기술의 발전은 AI 기술의 활용이 사회 전반에 걸쳐 확대되는데 큰 기여를 하였다. DNN은 얼굴 인식, 신용 카드 위 · 변조 검출 [1, 2]과 같은 분야에서 활용되고 있을 뿐만 아니라 자율 주행 및 의료 진단 [3, 4]과 같이 학습 결과가 인간에게 큰 영향을 줄 수 있는 분야로까지 활용 분야가 넓어지고 있으며 이에 대해서 많은 연구가 진행되고 있다.

DNN의 학습 결과가 인간에게 미치는 영향이 커질수록 DNN의 품질을 검증하는 것이 중요하다. 인간에게 미치는 영향이 큰 DNN의 학습에 사용되는 데이터 셋은 처리해야 하는 데이터가 포함하고 있는 특징(feature)이 많고 복잡하여 데이터 자체의 크기가 매우 크고 데이터의 수가 많아 학습에 많은 시간이 소요된다. 학습이 완료된 이후 품질 검증을 진행하였을 때 문제가 발견된다면 해당 문제를 해결한 이후 다시 DNN 모델을 학습시켜야 하기 때문에 많은 비용이 소모된다. 그렇기 때문에 DNN의 품질을 검증할 때는 초기에 요구사항을 검증하여 수정 비용을 감소시키는 것이 중요하다. DNN 모델이 학습 결과가 인간에게 큰 영향을 미치는 분야에 적용될 때는 정확성(Accuracy) 뿐만 아니라 신뢰성(Reliability)과 같은 다른 비기능적 요구사항 또한 적절하게 검증되어야 한다. 요구사항을 검증하기 위해서는 요구사항 분석 및 설계 단계에서의 체계적 검증, 정형 분석, 코드 검사 등의 활동이 포함되어야 한다. 하지만 DNN 모델의 요구사항 검증에서는 DNN 모델이 갖고 있는 문제들로 인해 기존 DNN의 성능 평가를 위해 사용되는 정확성을 측정하는 방법으로는 온전히 요구사항을 검증하는 것이 쉽지 않다. DNN의 요구사항 검증에 어려움을 주는 원인 중 대표적인 3가지를 선정하면 다음과 같다.

첫째, DNN의 모델 구조가 Black-Box로 되어 있어 DNN이 데이터 셋으로부터 어떤 특징을 학습하여 어떠한 과정을 거쳐 학습 결과를 도출해냈는지 인간이 쉽게 이해할 수 없다. 기존의 소프트웨어들은 소스 코드 분석을 통해서 확인할 수 있었지만 DNN에서는 모델이 Black-Box로 되어 있기 때문에 소스 코드 분석만을 통해서는 요구사항들을 검증하는 것이 어렵다. 이로 인해 학습 결과를 실제 활용 분야에서 적용하려고 하더라도 학습 결과에 대한 신뢰성을 완전히 보장할 수 없다. 학습 결과를 그대로 사용하는 것이 아닌 인간의 추가적인 검증 절차를 필요로 한다.

둘째, DNN의 학습 결과는 학습에 사용된 데이터 셋에 큰 영향을 받는다. 입력 데이터에 대해 원하는 DNN의 학습 결과를 얻기 위해서는 적절한 데이터로 이루어진 데이터 셋을 구성하여 DNN을 학습시켜야 한다. 원하는 학습 결과와 상이한 데이터로 데이터 셋을 구성하거나 데이터 셋을 이루고 있는 개별 데이터의 품질이 좋지 않은 경우에는 원하는 DNN의 학습 결과를 얻지 못한다. 적절한 데이터로 구성한 데이터 셋을 활용하여 원하는 성능의 DNN을 학습시켰다 하더라도 해당 데이터 셋에 포함된 편향(bias)으로 인해서 데이터 셋에 포함되어 있지 않은 데이터가 입력으로 들어왔을 때 적절하지 못한 결과를 발생시킬 수 있다. 학습에 사용된 데이터 셋의 편향으로 인한 문제는 2020년 발표된 PULSE [5] 기계학습 알고리즘을 활용하여 모자이크 처리된 얼굴 이미지를 고화질 이미지로 재구성하는 오픈 소스 프로그램 [6]에서 발생하였다. 해당 프로그램에서 유색 인종 이미지를 입력으로 주었을 때 유색 인종 이미지를 생성하는 것이 아닌 백인 이미지로 생성하는 문제가 발생하였다. 이는 학습에 사용된 데이터 셋에 유색 인종 데이터가 포함되어 있지 않고 백인 데이터 위주로 구성되어 있기 때문에 발생한 문제이다. 데이터 셋의 편향으로 인한 문제는 2019년 12월 미국 국립 표준 기술 연구소(NIST)의 보고서 [7]에서 확인할 수 있다. 보고서에 따르면 안면 인식에 사용되는 대다수의 AI는 편향을 가지고 있음이 드러나 있다. 뿐만 아니라 학습에 사용되는 데이터 셋에 적대적인 공격(Adversarial-attack)이 포함될 경우 DNN의 학습 결과에 악영향을 미칠 수 있다.

[그림 1.] 적대적 공격이 학습 결과에 미치는 영향 예시

[그림 1.]에서 확인할 수 있듯이 좌측의 정상적인 이미지에 약간의 회전(rotation)을 가하는 것만으로도 모델의 학습 결과는 완전히 달라질 수 있다. 언급한 사례에서 확인할 수 있듯이 DNN의 학습 결과가 적절하게 도출되기 위해서는 학습에 사용되는 데이터 셋을 적절히 구성해야만 한다.

셋째, DNN 학습을 진행할 때 과적합(Over-fitting) 문제가 발생하지 않도록 해야 한다. 과적합 문제가 발생할 경우 실험 환경에서의 DNN 성능이 높게 나올 수 있으나, 데이터 셋에 포함되어 있지 않은 데이터가 존재할 수 있는 실제 환경에 DNN이 적용 되었을 때는 입력 받은 데이터가 데이터 셋에 포함되지 않아 실험 환경에 비해서 낮은 성능을 얻게 되는 문제가 발생한다. 실제 DNN이 적용되는 환경에서는 데이터 셋에 포함되어 있지 않은 데이터가 입력으로 주어질 수 있는 경우가 있다. 앞서 언급한 적대적 공격 또한 그 예시 중 하나라고 할 수 있다. 이를 막기 위해 과적합 문제가 발생하지 않도록 DNN의 학습이 이루어져야 한다.

위와 같은 문제점들로부터 알아낼 수 있는 점은 DNN의 품질은 학습에 사용되는 데이터 셋을 검증하는 단계를 통해 DNN에서 발생할 수 있는 문제를 해결할 수 있다는 것이다. DNN 모델의 내부 구조를 인간이 완벽히 이해하고 있다면 데이터 셋이 다소 좋지 않다 하더라도 모델의 내부 구조를 수정하여 적절한 결과를 얻어낼 수 있도록 유도해 낼 수 있다. 하지만 DNN의 모델 내부 구조가 Black-Box로 되어 있기 때문에 모델의 내부 구조를 DNN의 요구사항에 맞게 구성하는 것 보다 학습에 사용되는 데이터 셋이 요구사항에 맞게 적절하게 구성되어 있는지 검증하는 것이 상대적으로 난이도가 쉽다. 앞서 언급한 바와 같이 요구사항의 경우 빠르게 검증될 수록 추후 발생할 수 있는 수정 비용을 크게 줄일 수 있다. 이는 통계적 모델의 금언인 'Garbage in, garbage out'을 통해서도 DNN의 데이터 셋 검증이 빠르게 요구사항을 검증할 수 있다는 사실을 자명하게 확인할 수 있다.

데이터 변형(Data augmentation)은 데이터 셋에 포함된 데이터에 일정 수준의 변형을 가한 데이터를 추가하여 데이터의 양과 다양성을 증가시켜 DNN이 보다 많이 특징을 학습할 수 있도록 할 수 있도록 한다. 뿐만 아니라 데이터 변형을 통하여 DNN 모델이 보다 다양한 변형에 대해서 적응할 수 있도록 학습할 수 있다. 그렇기 때문에 데이터 변형은 과적합 문제를 해결하기 위한 방법으로 많이 사용된다. 그렇지만 데이터 변형도 적절하게 사용되지 않을 경우에는 학습 결과에 악영향을 미칠 수 있기 때문에 적절한 변형이 가해져야 한다.

본 논문에서는 앞서 언급한 바와 같이, 데이터 변형을 통해 DNN 모델의 요구사항 검증을 위해 모델이 우선시하여야 하는 추가적인 학습 및 검증 데이터를 추가한다. 적절한 데이터 변형을 가한 데이터 셋을 준비하여 DNN 모델을 학습시킬 경우 기존에 모델의 성능 평가 척도로 활용되는 정확성을 유지하거나 개선하면서 변형된 데이터, 즉 모델의 학습 결과에 대한 신뢰성에 영향을 미칠 수 있는 데이터를 잘 식별할 수 있을 것이라는 가설을 설정하고, 이를 증명하는 실험을 진행하였다.

이후 전개될 논문의 내용들은 다음과 같이 구성되어 있다. 2장에서는 서론에서 제안한 문제점들과 해당 문제점들을 극복하기 위해 진행된 선행 연구들에 대한 소개를 다룬다. 3장에서는 설정한 가설을 증명하기 위해 진행한 실험 방법에 대해 소개하고, 4장에서는 실험

결과에 대한 분석을 진행한다. 5장에서는 논문의 결론을 정리하고 향후 진행할 연구 방향에 대해 소개한다.

2. 관련 연구

데이터 셋의 사전 검증 또는 변형이 DNN 모델의 요구사항을 검증하는데 도움을 줄 수 있음을 보인 이전의 연구들이 존재한다.

Dwork [8]과 Feldmen [9]의 연구에서 데이터 셋이 포함하고 있는 편향에 따라서 DNN의 학습 결과는 DNN을 사용하는 각 개인과 집단에 따라서 차별적인 결과를 도출해낼 수 있다. 이는 DNN의 요구사항 중 공정성(Fairness)을 만족시키지 못하는 결과라고 논문에서는 언급하고 있다. Dwork의 연구에서는 분류에서의 공정성에 대한 연구를 진행하였다. 데이터의 편향으로 인해 각 개인의 인종, 성별, 나이 등과 같이 민감할 수 있는 속성에 따라 차별된 결과를 도출해낼 수 있다는 점을 지적하며 차별이 발생하지 않도록 하기 위해서 데이터 셋에서 차별 요소를 찾아낼 수 있는 통계적인 방법을 제시하였고 공정성이 DNN의 학습 결과의 대상이 되는 각 개인에게 부여되는 의미를 사생활(privacy)과 관련이 있음을 연구를 통해 보여주었다. 데이터 셋의 편향이 각 개인에 대해서 발생할 수 있는 차별과 그를 제거하여 공정성을 만족시키려고 한 Dwork가 진행한 연구와 달리 Feldmen이 진행한 연구에서는 데이터 셋이 가지고 있는 편향이 각 집단에 미칠 수 있는 차별에 대한 문제와 문제를 해결하기 위한 방법에 대해서 제안하였다. 마이크로소프트(MS)에서는 AI 시스템의 공정성을 개선하기 위하여 Fairlearn [10]이라는 오픈 소스 프로그램을 만들었다. 데이터 셋에 존재하는 편향을 완전히 제거할 수 없기 때문에 Fairlearn에서는 공정성에 악영향을 미칠 수 있는 요소들을 발견하여 제거하는 작업을 수행한다. 이를 통해 AI 시스템이 만들어낼 수 있는 차별 문제를 사전에 검증하고 보완하여 사용자들에게 제공할 수 있을 수 있게 됨을 보였다.

데이터 셋의 조정을 통하여 모델의 과적합 문제를 해결하면서 학습 결과를 개선시킨 연구 사례들 또한 존재한다. Huang [11], Kim [12], Kurakin [13], Pei [14], Tian [15], Tramèr [16]가 진행한 연구들에서는 과적합이 발생한 모델이 실제 사용 환경에 적용되었을 때 발생할 수 있는 문제들을 해결하려고 진행되었던 연구들이다. 각 연구들에서 사용한 방법들은 다르지만 공통적으로 제안한 내용은 최초 DNN 모델 학습의 결과를 분석하여 학습 결과에 악영향을 미칠 수 있는 예시들을 찾고, 해당 예시들과 유사한 데이터를 데이터 변형을 적용한 후 데이터 셋에 포함시켜 학습을 다시 진행한다. 다시 학습된 DNN 모델은 과적합 문제에서 자유로울 수 있을 뿐만 아니라 보다 설계된 목적에 맞는 학습 결과를 보인다는 것을 각 연구에서 실험을 통해 증명하였다. 더 나아가 Gao [17], ZHANG [18]이 진행한 연구에서는 입력으로 주어지는 데이터 셋의 특성을 학습한 후, 학습한 특성을 기반으로 입력으로 주어진 데이터와 유사한 데이터를 합성할 수 있는 GAN(Generative adversarial nets) [19]을 활용하였다. 이러한 연구들에서는 최초 학습된 모델의 결과 분석을 통하여 데이터 셋으로부터 학습 결과에 악영향을 미칠 수 있는 데이터를 식별하고, 원래의 데이터 셋과 유사하면서도 악영향을 미칠 수 있는 데이터와 유사한 특징들을 갖고 있는 데이터를 생성한다. 이후 생성한 데이터를 기반으로 DNN 모델을 다시 학습시켰을 때 학습 결과에서 발생할 수 있는 이상치가 줄어들고 보다 더 요구사항 대로 동작할 수 있음을 연구에서 증명한다. Gao의 연구에서는 특별히 이상치 데이터 특성을 생성하여 학습 데이터 셋과 함께 모델을 다시 학습시키는 것이 DNN 모델이 학습에 사용된 데이터 셋에 존재하지 않는 입력 데이터가 주어진 상황에 대해서도 대응할 수 있는 유연성(robustness)을 갖추는데 도움이 된다는 것을 증명하였으며, ZHANG의 연구에서는 제안한 방법을 적용하였을 때 DNN 모델의 공정성에 좋은 영향을 미칠 수 있음을 밝히고 있다.

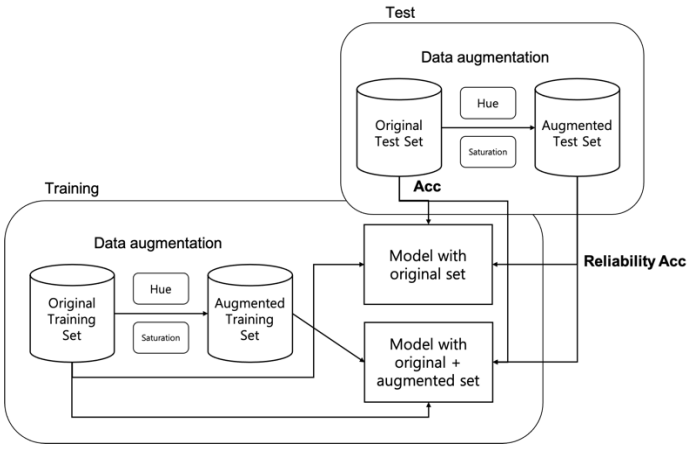
이전에 진행된 연구들을 통해서 DNN이 실생활에 활용되기 위해서 해결해야 하는 요소들, 그 중에서도 적절한 데이터 셋을 구축해야 한다는 점과 데이터 변형 또는 생성을 통해서 데이터 셋을 다시 구축할 경우 모델의 요구사항으로 판단할 수 있는 정확성 뿐만 아니라 유연성, 공정성에도 좋은 영향을 끼칠 수 있다는 점을 확인할 수 있었다. 본 논문에서는 이러한 이전 연구들로부터 영감을 얻어 '데이터 변형이 실제로 요구사항에 해당하는 요소들을 검증할 수 있을 것인가?' 라는 질문을 세우고, 이에 대한 답을 찾기 위한 실험을 진행하였다.

3. 실험 방법 및 내용

DNN의 목표가 다양하고 목표를 달성하기 위해 사용되는 데이터 셋의 종류 또한 매우 광범위하기 때문에 본 논문에서는 비교적 간단한 목표에 해당하는 *분류*를 위한 데이터 셋을 실험 데이터 셋으로 설정하였다. 또한, DNN이 만족시켜야 하는 요구사항들이 매우 다양하기 때문에 모델의 요구사항 가운데 *정확성*과 *신뢰성*에 대해서만 데이터 변형이 미칠 수 있는 영향에 대해서 실험을 진행하였다..

선정한 속성을 바탕으로 설정한 가설의 증명을 위하여 [그림 2.]와 같은 실험을 진행하였다. 분류를 위한 데이터 셋을 선정하고 각 데이터 셋에 대하여

데이터 변형을 진행한다. 데이터 변형이 정확성과 신뢰성에 미치는 영향을 파악하기 위하여 변형하지 않은 원본 데이터 셋만으로 학습시킨 모델과 변형하지 않은 원본 데이터와 변형이 가해진 데이터를 합친 데이터셋으로 학습시킨 모델을 준비한다. 각 모델에 대해서 변형 데이터를 포함하지 않은 원본 데이터 셋으로 진행한 평가를 통해서 학습된 모델의 정확성을 얻어낸다. 변형 데이터 셋으로 진행한 평가를 통해서 신뢰성을 얻어낸다.



[그림 2.] 실험 개요

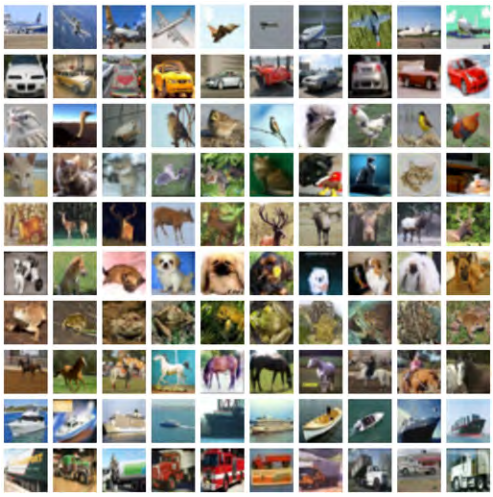
실험을 통해서 얻어낸 정확성과 신뢰성의 평가를 진행하였을 때, 원본 데이터 셋으로만 학습시킨 모델에 비해서 원본 데이터 셋과 변형 데이터 셋을 함께 학습시킨 모델에서 정확성 뿐만 아니라 신뢰성이 개선되는 것을 확인할 수 있었다.

본 논문의 실험을 통해서 데이터 변형이 기존에 알려진 것과 같이 DNN 모델이 검증해야 하는 요구사항 중 정확성을 개선하는데 도움을 줄 수 있을 뿐만 아니라 신뢰성 개선에도 영향을 미칠 수 있는 요인으로 볼 수 있음을 확인할 수 있다. 이를 통해 데이터 변형이 신뢰성 뿐만 아니라 DNN이 검증해야 하는 다른 요구사항 역시 검증할 수 있다는 가설을 세울 수 있는 근거로 제시한다. 나아가 실제로 다른 요구사항 속성으로 실험을 확장시켜 나가며 종국에는 기계 학습(Machine learning)과 심층 학습(Deep learning)을 사용하는 모든 소프트웨어에 대한 요구사항을 검증하기 위한 일련의 데이터 검증 프로세스를 제안할 수 있을 것으로 기대할 수 있다.

실험을 위해 사용한 데이터 셋과 데이터 셋에 적용한 데이터 변형 방법은 3.1.에서 확인할 수 있으며 실험 결과를 평가하기 위해 논문에서 사용하는 척도인 정확성과 신뢰성에 대한 정의와 수식 표현은 3.2.에서 확인할 수 있다. 가설을 증명하기 위해 설계한 실험 방법에 대해서는 3.3에서 다룬다. 실험에 코드는 모두 Python과 기계 학습과 심층 학습을 위해 사용되는 라이브러리인 PyTorch [20]를 사용하여 작성되었다.

3.1. 사용 데이터 셋과 데이터 변형 방법

실험에 사용한 데이터 셋은 분류를 목적으로 하는 DNN에서 가장 기초적으로 활용되고 있는 데이터 셋인 CIFAR-10 [21]을 사용하였다. CIFAR-10 데이터 셋은 10개의 클래스와 32X32 크기의 RGB색 채널을 갖고 있는 60,000개의 데이터로 구성되어 있다. 각각의 클래스는 6,000개의 데이터로 구성되어 있으며 이 중 5,000개의 데이터는 DNN을 학습하는데 사용되고 1,000개의 데이터는 학습된 DNN 모델을 평가하는데 사용된다. 학습 데이터는 총 50,000개로 구성되어 있으며, 평가 데이터는 총 10,000개로 구성되어 있다. 실험에서는 아무런 데이터 변형을 가하지 않은 CIFAR-10 데이터 셋을 원본 데이터 셋(Baseline)으로 설정하였다.



[그림 3.] CIFAR-10 데이터 셋

설정한 가설의 증명을 위해 Baseline에 변형을 가해야 한다. 실험에서 사용하기 위한 변형 데이터 셋을 만들기 위해서 PyTorch에 내장되어 있는 ColorJitter 클래스를 활용하여 원본 데이터 셋에서 변형된 데이터 셋을 불러올 수 있도록 하였다. ColorJitter 클래스는 밝기(brightness)와 대비(contrast), 채도(saturation)와 색상(hue)을 매개변수로 받아 랜덤하게 밝기, 대비, 채도, 색상을 변경한 이미지를 만드는 역할을 수행한다. 실험에서는 밝기와 대비는 기본 값인 0으로 두고 채도와 색상 값만을 조절하여 변형을 진행하였다. 색상 값을 0.5로 적용하여 색상 변형 데이터 셋을 생성하였고 채도 값을 0.5로 적용하여 채도 변형 데이터 셋을 생성하였다. 그리고 색상과 채도를 모두 0.5씩 적용하여 두 가지 변형이 모두 가해진 색상-채도 변형 데이터 셋을 생성하였다. 각 데이터 변형 과정을 통해 생성되는 데이터 셋은 [그림 4.]에서 확인할 수 있다.

[그림 4.] 데이터 변형이 적용된 데이터 셋
(좌측 : 색상 변형, 가운데 : 채도 변형, 우측 : 모두 적용)

이렇게 생성한 변형 데이터 셋의 경우 실험에서 원본 데이터 셋만으로 학습시킨 모델과의 성능 비교를 위해서 원본 데이터 셋과 합쳐 학습 데이터 셋을 생성하고 생성한 학습 데이터 셋을 활용하여 모델을 학습시켜 정확성을 평가하는데 사용된다. 변형된 평가 데이터 셋의 경우 학습된 각 모델의 신뢰성을 평가하는데 사용된다.

3.2. 정확성과 신뢰성의 정의

실험 결과의 평가를 위해 두 가지 평가 척도를 활용한다. 수식을 정의하기 앞서 수식에 사용되는 요소들을 정의한다. *total_all*은 평가에 사용된 데이터의 총 개수를 의미한다. *correct_all*은 DNN 모델이 정확하게 맞춘 데이터의 총 개수를 의미한다. *total_aug*는 평가에 사용된 변형 데이터의 총 개수를 의미한다. *correct_aug*는 DNN 모델이 정확하게 맞춘 변형 데이터의 총 개수를 의미한다.

정확성(acc)은 일반적으로 DNN 모델에서 사용되는 정확성과 같이 학습된 모델이 평가 데이터를 얼마나 정확하게 분류 하였는지를 파악할 수 있는 척도로 활용한다. 원본 데이터 셋만으로 학습된 모델의 경우 원본 평가 데이터 셋의 결과만을 반영하지만 변형 데이터 셋과 합쳐져서 학습된 모델의 경우 원본 데이터와 변형 데이터 모두를 포함하여 정확성을 계산한다. 정확성을 수식으로 표현하면 (1)과 같이 표현할 수 있다.

$$acc = correct\_all \;/\; total\_all \times 100 \quad \text{----------(1)}$$

신뢰성(reliability_score)은 본 논문에서 보이고자 하는 주 평가 척도로써 활용된다. 가정에 따르면 DNN 모델의 신뢰성을 검증하기 위해서는 DNN의 신뢰성에 영향을 미칠 수 있는 변형된 데이터를 학습된 모델이 얼마나 잘 분류하는지를 확인해야 한다. 실험에서는 변형된 데이터를 DNN이 신뢰성 있는 결과를 보이기 위해 반드시 분류해야 하는 것으로 가정한다. 신뢰성이 높을 수록 DNN의 결과를 신뢰할 수 있다고 판단할 수 있다. 이를 수식으로 표현하면 (2)와 같다.

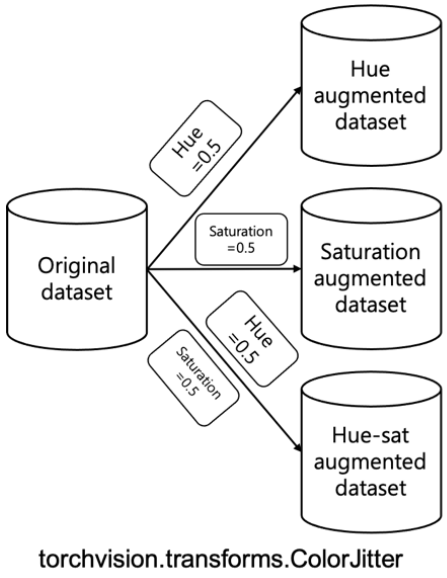
본 논문에서 사용하는 신뢰성은 분류를 목적으로 한 데이터 셋과 모델에 대해서만 적용되기 때문에 실제 DNN 모델의 신뢰성 항목에 대한 평가 척도를 반영하지 않는다. 본 논문에서 가정한 신뢰성은 일반적으로 통용되지 않는다. 학습된 DNN 모델이 변형된 데이터를 평가 데이터로 제공하였을 때 얼마나 정확하게 분류하는지를 확인하는 변형 데이터의 정확성과 같은 의미를 갖는다.

$$Reliability\_score = correct\_aug \;/\; total\_aug \times 100 \quad \text{- (2)}$$

3.3. 실험 방법

본 논문에서 설정한 가설을 증명하기 위해 설계한 실험 방법에 대해서 소개한다.

3.3.1. 데이터 변형



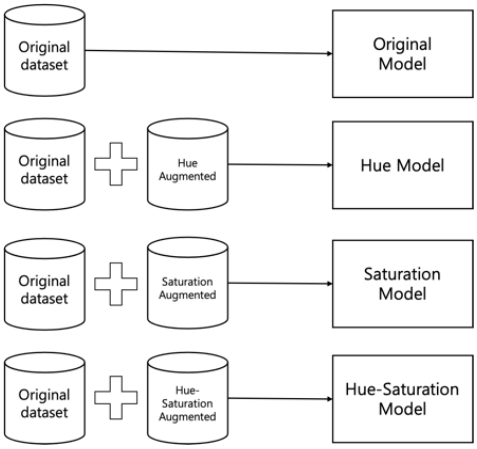
[그림 5.] 데이터 변형

[그림 5.]에서 확인할 수 있듯이 원본 데이터 셋에 PyTorch에서 데이터를 불러오는 과정에서 데이터 변형을 가할 수 있는 내장 클래스인 ColorJitter 클래스를 활용하여 원본 데이터 셋에 ColorJitter(hue=0.5), ColorJitter(saturation=0.5), ColorJitter(saturation=0.5, hue=0.5)를 각각 적용하여 변형된 데이터 셋을 생성한다. 생성된 변형 데이터 셋의 경우 원본 데이터 셋인 CIFAR-10과 같은 클래스와 수로 데이터가 구성되어 있으며 학습 및 평가 단계에서 불러와서 사용되게 된다.

3.3.2. 학습 모델 생성

3.3.1.에서 학습 및 평가에 사용될 데이터 셋들을 준비하고 나서는 DNN 모델을 학습시키는 단계를 거친다. 실험에 사용한 모델은 ResNet18 [22]을

사용하였다. CIFAR-10의 분류 모델의 경우 DNN의 발달에 따라서 매우 높은 성능을 보인다. 하지만 그럴 경우 논문에서 보이고자 하는 척도들에 대한 성능 향상 폭을 확인할 수 없기 때문에 Epoch과 Batch size를 조절하여 어느 정도 성능을 낮춘 상태로 학습을 진행하였다. 실험에서 사용된 파라미터 정보는 다음과 같다. Epoch : 30, Batch size : 64, learning rate : 0.001, momentum : 0.9



[그림 6.] 학습 모델 생성

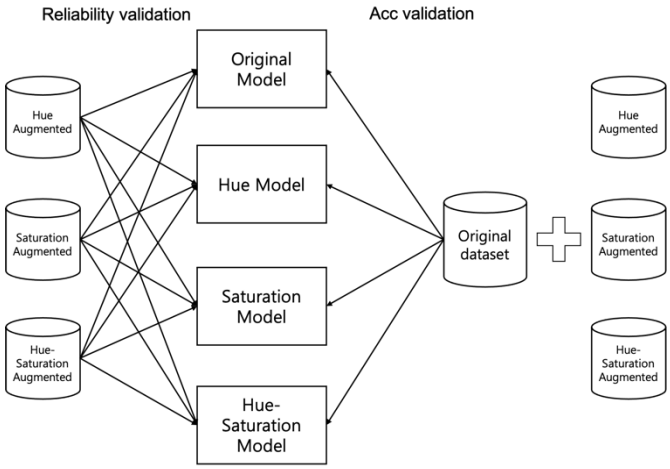
[그림 6.]과 같이 원본 데이터 셋과 만들어낸 변형 데이터 셋을 사용하여 학습을 진행, 총 4가지 모델을 생성한다. 첫 번째 모델은 실험의 성능 기준이 되는 모델로써 원본 데이터 셋만으로 학습된다. 나머지 모델의 경우 데이터 변형이 적용되었을 때 기준 모델에 비해서 얼마나 설정한 평가 척도의 차이를 확인하기 위해 학습시킨 모델이다. 논문에서는 추가적으로 어떤 데이터 변형이 가장 성능에 미치는 영향이 큰 지 확인하기 위해 원본 데이터 셋과 각각 변형 데이터 셋을 적용하여 모델을 학습시켰다.

3.3.3. 학습 모델 평가

학습 시킨 모델을 대상으로 평가를 진행한다.

모델 평가를 위해 사용되는 데이터 셋의 경우 학습에 사용된 데이터 셋과 마찬가지로 원본 평가 데이터 셋에 대해서 각 데이터 변형을 적용하여 변형 평가 데이터 셋을 생성하였다. 생성한 각 변형 평가 데이터 셋은 학습시킨 모델의 정확성과 신뢰성을 측정하는데 사용된다.

신뢰성의 경우 학습된 모델을 변형 평가 데이터 셋을 활용하여 평가를 진행한 결과이다. 정확성의 경우 원본 평과 데이터 셋과 변형 평가 데이터 셋을 모두 활용하여 평가를 진행한 결과이다. 정확성과 신뢰성을 계산하는 방법은 3.2.에서 확인할 수 있다.



[그림 7.] 학습 모델 평가

이를 위해 학습된 모델은 [그림 7.]과 같이 2번에 나누어 평가가 진행된다. 변형 평가 데이터 셋을 활용하여 학습 모델의 신뢰성에 대한 결과(Reliability validation)를 얻어내고, 첫 번째 평가의 결과를 누적하여 원본 평가 데이터 셋을 활용하여(Acc validation) 정확성에 대한 결과를 얻어낸다.

4. 실험 결과

4장에서는 3장에서 설계한 실험 방법을 따라 진행하였을 때 결과가 어떻게 나왔는지 소개한다. 설정한 가설에 따르면 데이터 변형을 포함한 데이터 셋으로 학습시킨 모델에서 원본 데이터 셋으로 학습시킨 모델에 비해서 정확성과 신뢰성이 향상된 결과를 얻어낼 수 있어야 한다.

[표 1.] 정확성 결과(단위 : %)

	Baseline	Hue	Saturation	Both
Acc	80.96	83.39	**85.37**	82.45

[표 1.]는 실험을 통해 얻어낸 정확성 결과를 보여준다. 결과를 보았을 때 변형 데이터를 포함하지 않은 원본 데이터 셋으로만 학습을 진행한 경우 가장 정확성이 낮은 것을 확인할 수 있고 색상(hue)와 채도(saturation)을 각각 변형한 데이터를 활용하여 학습시킨 모델과 색상과 채도를 모두 변형한 데이터 셋을 활용하여 학습시킨 모델에서 정확성이 증가했음을 확인할 수 있다. 특히 채도를 변형한 데이터 셋을 활용한 모델에서 가장 높은 정확성을 보이는 것을 확인할 수 있다.

[표 2.] 신뢰성 결과(단위 : %)

	Hue(Test)	Saturation(Test)	Both(Test)	Avg
Baseline	72.58	78.27	71.74	74.19
Hue	83.36	82.65	83.06	**83.02**
Saturation	76.62	85.26	76.53	79.47
Both	82.51	83.90	81.93	**82.78**

[표 2.]는 실험을 통해 얻어낸 신뢰성 결과를 보여준다. 각 행이 의미하는 것은 해당하는 데이터 셋으로 학습시킨 DNN 모델을 의미한다. 각 열에 해당하는 변형 데이터 셋으로 모델 평가를 진행하였을 때의 결과를 확인할 수 있다.

1행을 보았을 때 변형 데이터를 포함하지 않은 원본 데이터 셋으로만 학습을 진행한 모델의 경우 평균 74.91%로 가장 낮은 성능을 보이는 것을 확인할 수 있다. 반면 각각 색상과 채도를 각각 변형한 데이터를 활용하여 학습시킨 모델과, 색상과 채도를 모두 변형한 데이터를 활용하여 학습시킨 모델에서는 각각 평균 신뢰성이 83.02%, 79.47%, 82.78%로 원본 모델에 비해 높은 신뢰성 성능을 보이는 것을 확인할 수 있다.

특히 표의 2행에서 확인할 수 있는 색상을 변형한 데이터 셋으로 학습시킨 모델의 경우 평균 신뢰성이 가장 높게 측정된 것을 확인할 수 있다.

3행의 채도를 변형한 데이터 셋으로 학습시킨 모델의 경우 평가 데이터 셋이 채도 변형 데이터 셋인 경우 가장 높은 성능을 보였지만 색상 변형 데이터 셋과 색상-채도 변형 데이터 셋에 대해서는 다른 모델들에 비해서 낮은 신뢰성을 보인다. 그럼에도 불구하고 원본 데이터 셋으로만 학습을 진행한 모델에 비해서 높은 성능을 보이는 것을 확인할 수 있다.

4행의 색상-채도 변형 데이터 셋으로 학습시킨 모델의 경우 색상 변형 데이터 셋으로 학습시킨 모델과 비슷한 신뢰성을 보이는 것을 확인할 수 있다. 평균 신뢰성을 비교하였을 때 변형 데이터를 포함하여 학습시킨 모델들이 원본 데이터 셋으로만 학습시킨 모델에 비해서 모두 높은 신뢰성을 보이는 것을 확인할 수 있다.

[표 1.]과 [표 2.]에서 확인할 수 있는 결과를 종합하면 원본 데이터 셋으로만 학습시킨 모델의 정확성과 신뢰성에 비해서 원본 데이터 셋과 변형 데이터 셋을 합쳐 학습시킨 모델의 정확성과 신뢰성이 더 높은 성능을 얻을 수 있는 것을 확인할 수 있다. 도출된 실험 결과를 통해 데이터 변형을 통해서 원본 데이터 셋에 변형된 데이터를 포함시켜 모델 학습을 진행할 경우 그렇지 않은 경우에 비해서 DNN 모델의 정확성 뿐만 아니라 신뢰성에 대해서도 성능 향상 효과를 볼 수 있음을 확인할 수 있다. 이는 본 논문에서 설정한 가설과 일치하는 결과이다.

추가로 논문의 실험에서는 가해진 데이터 변형의 종류가 모델의 정확성과 신뢰성에 미치는 영향을 보기 위해서 3가지 서로 다른 변형을 가한 데이터 셋을 활용하여 학습을 진행하였다. [표 1.]에서는 채도를 변형한 데이터 셋을 활용한 모델에서 가장 높은 정확성을 얻을 수 있었지만 [표 2.]에서는 채도를 변형한 데이터 셋을 활용한 모델에서 다른 변형들에 비해서 가장 낮은 신뢰성을 보인 것을 확인할 수 있다. 이를 통해서 정확성과 신뢰성에 영향을 미치는 데이터 변형이 서로 다름을 확인할 수 있다.

5. 결론 및 향후 연구

본 논문에서는 데이터 변형이 DNN 모델의 요구사항 중 정확성 뿐만 아니라 신뢰성을 검증할 수 있을 것이라고 가정하였고, 가설의 증명을 위해 적절한 데이터 변형이 DNN 모델의 비기능적 요구사항 검증을 위해서 사용될 수 있음을 보일 수 있는지 실험을 통해 증명하고자 하였다. 실험을 위해 분류를 위해 활용되는 데이터 셋인 CIFAR-10 데이터 셋과 PyTorch의 데이터 변형 클래스인 ColorJitter를 원본 데이터 셋에 적용하여 변형 데이터 셋을 구성하였다. 구성한 데이터 셋들을 활용하여 DNN 모델을 학습시키고 정확성과 신뢰성을 평가하는 실험을 진행하였다. 도출된 실험 결과를 통해 원본 데이터 셋과 변형 데이터 셋을 함께 사용하여 DNN 모델을 학습시킬 경우가 원본 데이터 셋만을 사용하여 학습시킨 DNN 모델에 비해서 더 높은 정확성을 보일 뿐만 아니라 더 높은 신뢰성을 보이는 것을 확인할 수 있었다. 이를 통해 데이터 변형이 DNN 모델의 요구사항을 검증할 수 있는 요소 중 하나로 활용될 수 있음을 확인할 수 있다.

단, 본 논문에서 진행한 실험의 경우 실험에 사용한 데이터를 분류 모델에만 적용하였으며 평가 척도 중 하나인 신뢰성에 대한 정의 또한 실제 신뢰성을 위협하는 데이터 대신 임의로 선택된 일부 데이터에 대해 변형을 가한 결과로 정의되어 일반적으로 활용할 수 없다는 제한 사항을 갖고 있다. 향후 연구에서는 보다 다양한 데이터 셋에 대해 유사한 실험을 진행하여 다른 데이터 셋에 대해서도 동일한 결과를 보이는지에 대한 연구를 진행할 예정이다. 이후 연구 결과에 따라 분류가 아닌 탐지(detection)와 같이 다양한 모델에 대해 실험을 확장 해나가며 적용하는 데이터 변형과 평가 척도를 일반화 해나가는 연구를 진행할 예정이다.

참고문헌

[1] Fu, Kang, et al. “Credit card fraud detection using convolutional neural networks.” *International Conference on Neural Information Processing*. Springer, Cham, 2016.

[2] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. “Facenet: A unified embedding for face recognition and clustering.” *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[3] Bojarski, Mariusz, et al. “End to end learning for self-driving cars.” *arXiv preprint arXiv:1604.07316*(2016).

[4] Vieira, Sandra, Walter HL Pinaya, and Andrea Mechelli. “Using deep learning to investigate the neuroimaging correlated of psychiatric and neurological disorders: Methods and applications.” *Neuroscience & Biobehavioral Reviews* 74 (2017): 58-75.

[5] Menon, Sachit, et al. “PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models.” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

[6] https://github.com/tg-bomze/Face-Depixelizer

[7] Grother, Patrick J., Mei L. Ngan, and Kayee K. Hanaoka. “Face recognition vendor test part 3: Demographic effects.” (2019).

[8] Dwork, Cynthia, et al. “Fairness through awareness.” *Proceedings of the 3rd innovations in theoretical computer science conference*. 2012.

[9] Feldman, Michael, et al. “Certifying and removing disparate impact.” *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015.

[10] Bird, Sarah, et al. Fairlearn: A toolkit for assessing and improving fairness in AI. Technical Report MSR-TR-2020-32, Microsoft, May 2020. URL https://www.microsoft.com/en-us/research/publication/fairlearn-a-toolkit-for-assessing-and-improving-fairness-in-ai.

[11] Huang, Ling, et al. “Adversarial machine learning.” *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. 2011

[12] Kim, Jihan, Robert Feldt, and Shin Yoo. “Guiding deep learning system testing using surprise adequacy.” *2019 IEEE/ACM 41st International Conference on Software Engineering(ICSE)*. IEEE, 2019.

[13] Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. “Adversarial machine learning at scale.” *arXiv prepreint arXiv:1611.01236*(2016).

[14] Pei, Kexin, et al. “Deepxplore: Automated whitebox testing of deep learning systems.” *proceedings of the 26th Symposium on Operating Systems Principles*. 2017.

[15] Tian, Yuchi, et al. “Deeptest: Automated testing of deep-neural-network-driven autonomous cars.” *Proceedings of the 40th international conference on software engineering*. 2018.

[16] Tramèr, Florian, et al. “Ensemble adversarial training: Attacks and defenses.” *arXiv prepreint arXiv:1705.07204*(2017).

[17] Gao, Xiang, et al. “Fuzz testing based data augmentation to improve robustness of deep neural networks.” *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 2020.

[18] ZHANG, Peixin, et al. “White-box fairness testing through adversarial sampling.” (2020).

[19] Goodfellow, Ian, et al. “Generative adversarial nets.” *Advances in neural information processing systems*. 2014.

[20] Paszke, Adam, et al. “Pytorch: An imperative style, high-performance deep learning library.” *Advances in neural information processing systems*. 2019.

[21] https://www.cs.toronto.edu/~kriz/cifar.html

[22] He, Kaiming, et al. “Deep residual learning for image recognition.” *Proceedings of the IEEE conference on computer vision and pttern recognition*. 2016

마코프 모델을 이용한 PM 추론 및 행위 분석

김 소 연 Tung La-Ngoc 권 기 현

요 약

소프트웨어 개발 팀 프로젝트에서 PM은 계획, 개발 및 관리에 대한 책임을 가지므로 소통에 대하여 적극적이고 책임감이 강할 것으로 기대된다. 본 논문에서는 비대면으로 진행된 팀 프로젝트의 채팅 데이터를 받아 확률 모델 체킹으로 PM을 추론하였다. 총 11개의 모든 팀에서 실제 PM을 정확하게 추론하였으며, 확률 논리 언어로 명세한 질의의 확률을 통해 수치적으로 소통에 대해 적극적임을 확인하였다. 또한, 웹 기반 학습 관리 시스템(LMS)의 로그를 분석하여 PM과 이외의 팀원의 행위를 비교하였다. 그 결과 PM이 다른 팀원에 비하여 공지를 더 자주 확인 하는 등 책임감이 높음을 확인하였다.

Ⅰ. 서 론

비대면 시대에 소프트웨어 개발의 인력적 측면에서 프로젝트 관리자(Project Manager, PM)은 팀 프로젝트 성공에 많은 영향을 준다[1]. PM은 프로젝트 계획, 개발 및 관리의 종합적 책임을 가지므로 다른 팀원에 비하여 능동적이고 책임감이 강할 것으로 기대된다. 본 논문에서는 실제 데이터를 이용하여 수치 값으로 이를 확인하는 것을 관심사로 가진다. 이를 위하여 팀 프로젝트 수행 시 사용되었던 온라인 채팅 데이터를 수집하여 사전 정보 없이 확률 모델 체킹만을 이용하여 PM을 추론한다.

또한 PM의 행위를 분석하고 다른 팀원과의 비교를 위해 학교 웹-기반의 LMS(Learning Management System) 사용 로그를 분석하여 PM의 행위를 분석하였다.

Ⅱ. 채팅 데이터를 이용한 PM 추론

채팅 및 웹 서비스 사용 로그는 시간의 흐름에 따라 순차적으로 변하는 이산 확률 과정이다[2]. 이를 기반으로 팀 프로젝트에서 사용되었던 채팅 데이터를 이용하여 PM을 추론한다. 채팅 데이터는 5개의 유형으로 구분하여 내용을 추상화하였다.

모든 채팅 데이터는 (참여자, 유형, 감정유형) 3-튜플로 구성된다. 감정유형은 pos, neg, NA로 세분된다. 채팅 데이터와는 별개로 채팅의 시작과 끝을 나타내는 'START'와 'STOP'을 추가하였다.

채팅 데이터를 기반으로 상태 간 전이 확률을 도출한다. 계산한 전이 확률을 상태에 적용하여 채팅 데이터 기반 이산 마코프 모델을 구축한다. 또한, 정량 추론을 위하여 상태 및 전이에 보상 값을 부여한다.

구축한 마코프 모델로부터 PM을 추론하고자 PM

항목	세부 항목	질의	질의 명세
소통	반응	다른 팀원에게 긍정적 응답을 표현하는 정도?	filter(avg, R {"r_참여자"}=? [F "ack" & "감정"], "START")
		다른 팀원의 응답을 얼마나 이끌어내는가?	filter(avg, R{"r_참여자_sug"}=[F "STOP"], "START")
		다른 팀원에게 응답하는 정도?	filter(avg, R{"r_참여자_ack"}=? [F "STOP"], "START")
	참여	다른 팀원의 정보요청에 얼마나 빠르게 응답하는가?	filter(avg, R {"r_steps"}=? [F "참여자1"&"ack"], "참여자2"&("info"|"sug"))
		전체 대화에 어느 정도 참여하는가?	filter(avg, R{"r_참여자"}=? [F "STOP"], "START")
		T시간 동안 대화에 어느 정도 참여하는가?	filter(avg, R{"r_참여자"}=? [C<=T], "START")
		정보 제공 정도?	filter(avg, R{"r_참여자_share"}=? [C<=T], "START")

표 1 PM 추론에 사용된 질의

항목	세부항목	질의	질의 명세	PM	팀원
책임감	제출기한	과제 확인 후 얼마나 빠르게 제출하는가?	filter(avg, R {"r_steps"}=? [F "homework"], "notice")	9.182	58.887
	민첩	특정 페이지에 얼마나 자주 접근하는가?	filter(avg, R {"r_페이지"}=? [F "STOP"], "START")	모든 페이지에 대하여 PM의 방문 비율이 높음	
		웹 서비스 사용 기간?	filter(avg, R {"r_steps"}=? [F "STOP"], "START")	184.7612	52.92303
	정보확인	정보 확인을 위한 페이지에 얼마나 자주 방문하는가?	filter(avg, R {"r_steps"}=? [F "notice"], "START")	11.45964	13.94034
		T 시간 동안 어떠한 페이지를 사용하는가?	filter(avg, R {"r_페이지"}=? [C <= T], "START")	두 그룹 모두 lecture에 가장 많이 방문하며 PM의 경우 homework에 도달하는 횟수가 상대적으로 높음	

표 2 웹 로그 분석에 사용된 질의

이 가져야 할 속성을 확률 논리로 질의한다. 핵심 속성은 '소통'으로 이를 세분화 한 속성에 해당하는 질의는 표 1과 같다.

PM의 특징과 가장 유사한 대화를 하는 참여자를 추론하기 위하여 질의 별 가장 높은 수치를 가지는 참여자에게 1의 가중치를 부여하고 총합이 가장 큰 경우를 PM으로 추론한다. 추론 결과를 실제와 비교한 결과 아래의 표와 같이 모두 일치한다.

	1	2	3	4	5	6	7	8	9	10	11
추론	C	B	A	C	B	A	A	C	C	A	A
실제	C	B	A	C	B	A	A	C	C	A	A

Ⅱ. 로그 데이터 기반 웹 이용 방식 추론

추론한 PM의 행동을 분석하기 위하여 데이터를 전처리한다. 총 30만개의 LMS 로그 데이터를 사용하였으며 PM과 팀원의 기록을 분류한다. 분류한 그룹마다 로그 발생 시간과 방문 페이지에 대한 정보를 추출한다. 또한, 정량적 추론을 위하여 상태 및 전이에 보상 값을 부여한다.

보상 값을 가지는 마코프 모델에서 PM과 팀원의 웹 이용방식을 비교하기 위하여 확률 논리 언어로 속성을 질의한다. 흥미로운 속성은 '책임감'이며 이를 '제출기한', '민첩', '정보확인'으로 세분화한다. 세분화 된 속성에 해당하는 질의 및 분석 결과는 표 2와 같다.

Ⅳ. 결 론

본 논문에서는 확률 모델 체킹을 이용하여 아무런 사전 정보 없이 채팅 데이터만을 이용하여 팀별 PM을 추론하였다. 이를 위하여 총 11개의 팀 채팅 데이터를 받아 마코프 모델을 구축하였고, PM이 지녀야할 속성 7개를 확률 논리 언어로 질의하였다. 분석 결과 11개의 팀에 모두 PM을 정확하게 추론하였다. 또한 PM의 행위를 분석하기 위하여 모든 수강생들이 이용하는 LMS 로그를 조사하였다. 약 30만 건의 로그를 분석한 결과 PM은 공지사항을 더 자주 확인하는 등 책임감이 높은 것을 정량적 값으로 확인하였다. 본 연구를 보완한 향후 연구로 일반화된 결론을 도출하기 위해 더 많은 자료를 확보하고 end-to-end 종단 분석이 요구된다. 즉, 프로젝트 참여 수치 정보와 최종 결과물 같의 상관 분석 및 종속 변수를 파악할 필요가 있다. 또한, 기존 온라인 채팅 분석과의 비교가 요구된다[3].

References

[1] I. Sommerville, Software Engineering: Chapter 22 Project Management, Pearson, 2016.

[2] F. Biagini and M. Campanino, Elements of Probability and Statistics, Springer International Publishing, 2016.

[3] J. Meredith, "Conversation Analysis and Online Interaction," Research on Language and Social Interaction, Vol. 52, No. 3, pp.241-256, Aug 2019.

머신 러닝을 사용한 코드와 요구 사항 기반 테스트 케이스 우선 순위 지정 방법

범준석, 백종문

한국과학기술원

junseok.beom@kaist.ac.kr, jbaik@kaist.ac.kr

Code & Requirement-based Test Case Prioritization using Machine Learning

Junseok Beom, Jongmoon Baik

KAIST

요 약

회귀 테스트는 소프트웨어 개발 중에 변경되거나 새로운 요구 사항이 구현되어 확장 된 소프트웨어를 다시 테스트하는 작업이다. 전체 소프트웨어을 재 테스트하는 것은 합리적인 시간과 비용으로 실행 가능하지 않기 때문에 일반적으로 모든 테스트 케이스의 하위 집합으로 회귀 테스트 세트를 구성해서 회귀 테스트를 실행한다. 시스템 요구 사항을 검증하는 시스템 테스트 레벨과 같은 블랙 박스 테스팅은 요구 사항 기반 테스트 케이스 우선 순위 지정 방법이 이용되고 있다. 기존의 요구 사항 기반 방식은 코드 메타 데이터는 이용하지 않는다. 그러나 모든 요구 사항은 코드로 구현되기 때문에 효과적인 테스트 케이스 우선 순위 지정을 위해서는 코드 메타 데이터를 완전히 무시 할 수는 없다. 이 연구에서는 테스트 케이스 우선 순위 지정을 위해 코드 메타 데이터와 요구 사항 메타 데이터를 모두 고려하고 머신 러닝의 지도 학습 알고리즘을 이용하여 테스트 케이스 실패 확률을 예측하여 실패 확률이 높은 순으로 테스트 케이스 우선 순위를 지정하는 방법을 제안한다.

1. 서론

소프트웨어 테스트 활동은 소프트웨어 품질을 보장하기 위해 반드시 실행해야 하는 중요한 작업 중 하나이다. 테스트는 가능한 한 빨리 시작해야하며 소프트웨어 개발 단계의 산출물(코드, 디자인, 요구 사항)을 검증하기 위해 다양한 소프트웨어 테스트 레벨을 적용해야 한다. 최신 소프트웨어 시스템은 매우 복잡하므로 안전성 있는 품질을 확보하기 위해서 소프트웨어 검증 활동은 개발 프로세스에서 중요한 역할을 담당하고 있다. 또한, 소프트웨어는 알려진 오류를 수정하거나 새로운 기능을 추가하기 위해 지속적으로 변경된다. 따라서 모든 소프트웨어 버전에 대한 전체 테스트는 소프트웨어 품질을 보장하기 위해 반드시 실시해야 하는 기본적인 활동이지만 안타깝게도 모든 소프트웨어를 테스트 하는 것은 시간과 비용이 많이 필요하기 때문에 현실적으로 불가능하다. 이러한 시간과 비용의 제한 때문에 회귀 테스트는 변경 사항이 이전에 구현 된 기능에 영향을 미치지 않는지 중점을 두어 실시한다. 소프트웨어 테스트 활동을 통해서 모든 소프트웨어 결함을 발견하는 것은 불가능하기 때문에 제한된 시간과 비용 안에서 최대한의 테스트 결과를 위해서 효과적인 테스트 케이스 우선 순위 지정 방법이 필요하다.

소프트웨어 산출물의 변경 사항을 반영하여 테스트 케이스 우선 순위를 지정하면 중요한 테스트 케이스를 먼저 실행 할 수 있게 된다. 자연어로 작성된 요구 사항을 검증하는 테스트 레벨에서는 코드 메타 데이터는 제외하고 요구 사항과 테스트 케이스 등의 메타 데이터를 사용하는 테스트 케이스 우선 순위 지정 방법을 사용한다. 또한, 머신 러닝을 이용한 요구 사항 기반 방법들은 요구 사항이 자연어로 작성되어 있기 때문에 자연스럽게 자연어 처리(Natural Language Processing)를 많이 이용한다 [1, 7, 9]. 즉, 각 요구 사항이나 테스트 케이스를 설명하는 문장들을 분석하고 다른 메타 데이터들을 활용해서 요구 사항과 관련된 테스트 케이스의 실패 확률 또는 테스트 케이스의 우선 순위 값을 예측한다. 하지만, 이 방법의 문제점은 모든 요구 사항이나 테스트 케이스가 자연어로 작성되어 있다는 것을 전제로 한다는 점이다. 다르게 얘기하면, 자연어로 작성되지 않은 요구 사항(상태-전이 다이어그램, 순서도)이 있다면 자연어 처리 분석에서 제외하거나 제대로 분석을 할 수 없어서 성능이 낮아질 것이다. 또한, 요구 사항을 구현하는 것은 코드이지만,

요구 사항 기반 테스트 케이스 우선 순위 지정 방법은 코드의 메타 데이터를 고려하지 않는다. 코드는 끊임 없이 변경되지만 요구 사항 기반 우선 순위 지정 방법은 이 변경사항을 고려하지 않는다. 결국, 요구 사항의 메타 데이터가 변경 되지 않는다면 테스트 케이스의 우선 순위는 변경되지 않는 문제점이 있다. 그렇다면, 왜 코드 메타 데이터를 사용하지 않는지 의문이 생긴다. 이는 코드와 요구 사항의 추적성(traceability)이 확보되지 않았기 때문이다. 즉, 코드와 관련된 요구 사항을 파악 할 수 없고 반대로 요구 사항과 관련된 코드를 파악 할 수 없어서 이전의 테스트 케이스 우선 순위 지정 방법들은 코드와 요구 사항 정보를 같이 사용 할 수 없었다. 하지만, 자동차 소프트웨어 개발 프로세스인 Automotive-SPICE (ASPICE)에서는 코드와 요구 사항의 추적성을 요구하고 있다 [5]. [그림1]은 ASPICE에서 요구하는 소프트웨어 산출물 간의 추적성 정보이다.

ASPICE 개발 프로세스를 준수하는 개발 프로젝트는 코드와 관련된 요구 사항을 파악할 수 있어서 요구 사항과 관련된 코드의 메타 데이터도 고려해서 테스트 케이스 우선 순위 지정에 활용 할 수 있다.

이번 연구에서는 ASPICE 개발 프로세스를 준수하는 자동차 산업의 프로젝트에서 코드와 요구 사항 간의 추적성 정보를 이용해 코드와 요구 사항의 메타 데이터를 모두 고려하는 새로운 테스트 케이스 우선 순위 지정 방법을 제안한다. 그리고, 여러가지 머신 러닝 알고리즘을 사용하여 접근 방식을 다양화하고 확장하여 모델의 성능을 평가한다. 사용한 머신 러닝 알고리즘은 Neural network, SVM, XGBoost 이고 추가로 우리가 제안하는 방법이 코드 메타 데이터를 사용하지 않는 요구 사항 기반 방식에 비해 소프트웨어 결함 발견 비율을 향상시키는 것을 증명한다.

2. 배경과 관련 연구

이 섹션에서는 테스트 케이스 우선 순위 지정의 개념과 이번 연구와 유사한 머신 러닝 모델을 사용해서 코드 기반과 요구 사항 기반의 테스트 케이스 우선 순위를 지정한 방법들에 대해 소개한다.

2.1 테스트 케이스 우선 순위 지정

테스트 케이스 우선 순위 지정(TCP: Test Case Prioritization)은 특정한 기준에 따라 테스트 케이스의 중요도와 우선 순위를 식별해서 우선 순위가 높은 테스트 케이스가 우선 순위가 낮은 테스트 케이스보다 먼저 실행되도록 순서를 조정하는 방법론이다. 또한, 테스트 케이스의 우선 순위는 고정되지 않고 소프트웨어 산출물(코드, 요구사항, 테스트 케이스 등)의 변경에 맞춰서 계속 변경되어야 한다 [6]. 즉, 소프트웨어 버전마다 중요한 테스트 케이스를 파악해서 빨리 실행함으로써 소프트웨어 결함을 좀 더 빨리 찾을 수 있다 [3].

테스트 케이스 우선 순위 지정 방법의 효율성을 측정하는 지표는 APFD를 사용한다 [1, 2, 3, 8, 10]. APFD는 테스트 케이스 우선 순위 지정이 완료되고 지정된 순서로 테스트를 실행 했을 때 테스트 실행 대비 소프트웨어 결함을 얼마나 빠르게 찾는지를 나타내는 지표이다.

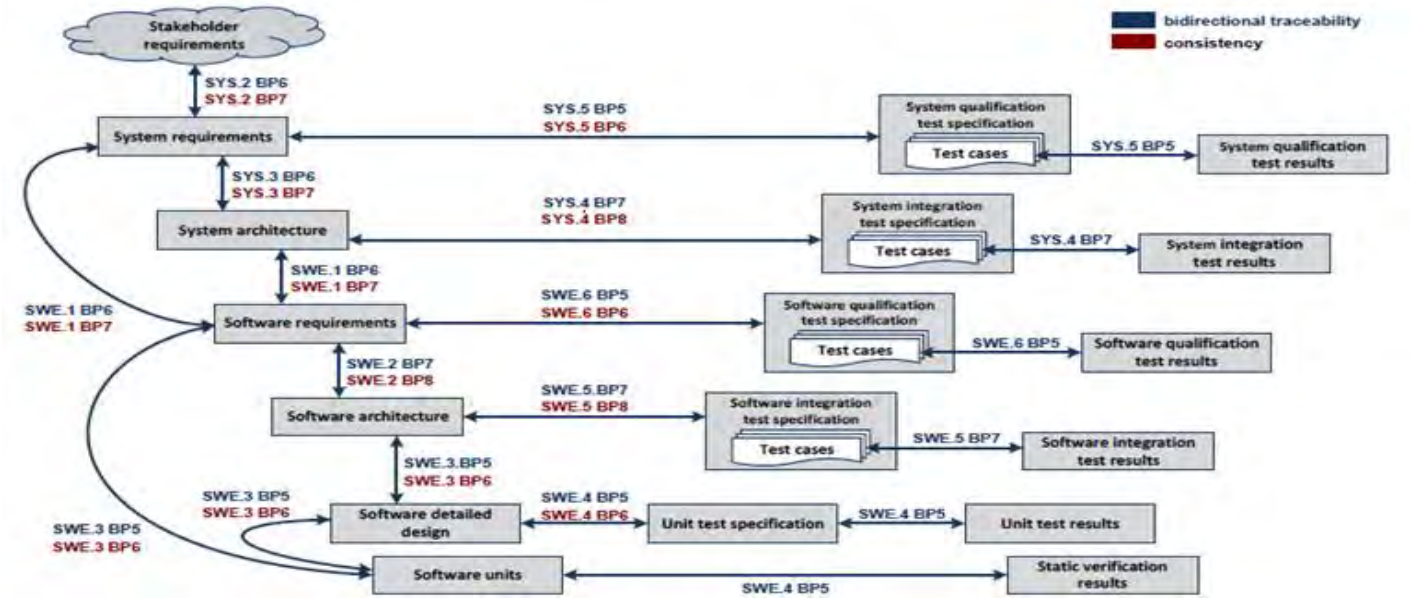


그림 1. ASPICE 추적성

Ensemble 학습을 위해 사용한 알고리즘 중에서 가장 성능이 좋은 모델의 출력을 결합하여 성능을 평가한다.

APFD는 결함 감지 비율을 높이는 데 중점을 두고 있으며, 테스트 과정에서 결함을 발견하는 속도를

높이고 감지 된 결함의 평균 누적
비율을 측정한다. APFD 식은 [식1]과 같다.

$$\mathrm{APFD} = 1 - \left\{\frac{\sum_i^m TF_i}{\mathrm{n} \times \mathrm{m}}\right\} + \frac{1}{2 \times \mathrm{n}}$$

식 1. APFD

n은 실행한 테스트 케이스의 수, m은 발견한 소프트웨어 결함 수, 그리고 TF_i는 i번 결함을 발견한 테스트 케이스의 실행 번호이다.

2.2 코드 기반 테스트 케이스 우선 순위 지정

Xiaobin Zhao은 Unit 테스트 레벨에서 머신 러닝을 이용한 코드 데이터 기반 테스트 케이스 우선 순위 지정 방법을 제안했다 [4]. Unit 테스트 레벨은 시스템 테스트 레벨과는 달리 코드에 접근 할 수 있기 때문에 코드 메타 데이터를 활용하여 테스트 케이스 우선 순위를 지정한다. 사용한 메타 데이터는 클래스 및 함수와 테스트 케이스의 테스트 커버리지 정보, 코드 변경 정보, SLOC, 복잡도 등이다. 사용한 머신 러닝 알고리즘은 함수와 테스트 케이스의 커버리지 정보를 이용해서 동일한 함수를 테스트하는 테스트 케이스의 집합인 클러스터를 형성하고, 다른 메타 데이터들은 베이지안(Bayesian) 네트워크에 입력해서 테스트 케이스의 실패 확률을 예측한다. 2개 모델에 다른 메타 데이터를 입력으로 사용하고, 각 모델의 결과를 합해서 최종 결과를 예측하는 Ensemble 모델을 제안했다. 베이지안 모델에서 예측한 실패 확률이 높은 테스트 케이스가 있는 클러스터의 우선 순위를 높여서 클러스터 내의 테스트 케이스들을 먼저 실행한다. 이 모델은 동일한 함수를 테스트하는 테스트 케이스들을 같이 실행하는 장점이 있다. 단점은 소프트웨어의 규모가 성능에 중요한 요소가 될 수 있다. 즉, 함수의 수와 테스트 케이스가 증가 함에 따라 함수와 테스트 케이스의 커버리지 정보가 복잡해져서 클러스터링의 성능이 낮아 질 수 있다.

2.3 요구 사항 기반 테스트 케이스 우선 순위 지정

Remo Lachmann은 시스템 테스트 레벨에서 머신 러닝을 이용한 요구 사항 데이터 기반 테스트 케이스 우선 순위 지정 방법을 제안했다 [1, 7]. 시스템 테스트 레벨에서는 코드에 접근 할 수 없기 때문에 코드 관련 메타 데이터는 제외하고 요구 사항과 테스트 케이스의 메타 데이터 만을 사용한다. Remo Lachmann이 사용한 메타 데이터는 테스트 케이스의 내용(자연어)과 과거 실행 결과, 테스트 케이스와 요구 사항의 커버리지 등이다. 사용한 머신 러닝 알고리즘은 자연어로 작성된 테스트 케이스의 내용에서 정보 추출을 위해 자연어 처리(NLP)와 테스트 케이스 우선 순위 학습을 위해 SVM(Support Vector Machine) 을 사용했다. 그리고 테스트 전문가들이 전체 테스트 케이스의 우선 순위 값(label)을 설정했고, 이를 학습 데이터 세트와 시험 데이터 세트로 나누어 사용했다. 즉, Remo Lachmann이 제안한 방법은 테스트 케이스의 우선 순위 값을 예측하는 모델이다. 이 방식은 테스트 케이스의 우선 순위 값을 테스트 전문가들의 개인적인 의견에 따라 정하기 때문에 이 부분이 큰 바이어스로 작용 할 수 있다. 또한, 테스트 케이스의 내용을 파악하기 위해 자연어 처리(NLP)를 사용하는데 자연어로 작성되지 않은 테스트 케이스인 경우에는 성능이 떨어지는 문제점이 있다.

3. Approach: 머신 러닝을 사용한 코드와 요구 사항 기반 테스트 케이스 우선 순위 지정

이 섹션에서는 우리가 제안하는 방법과 사용 하는 코드와 요구 사항의 메타 데이터에 대해 소개한다.

3.1 Proposed Approach

우리의 접근 방식은 테스트 케이스의 우선 순위를 결정하기 위해 요구 사항과 관련된 코드를 파악하여 이를 모두 사용하는 것에 중점을 둔다. 그림 2는 우리가 제안하는 방식에 대한 전체 구조를 보여준다.

요구 사항과 코드 간의 추적 정보를 이용해서 각각의 요구 사항과 관련 있는 소스 코드 저장소(repository)를 파악하고 이를 기반으로 코드의 변경 정보, 관련된 저장소의 수 등의 정보를 사용한다. 이러한 정보는 기존의 요구 사항 기반 테스트 케이스 우선 순위 지정 방법과 쉽게 결합되어 사용 할 수 있다. 또한, 요구 사항과 관련된 테스트 케이스와 소프트웨어 결함의 메타 데이터도 사용한다. 이러한 메타 데이터들을 feature로 사용하고 이는 머신 러닝 알고리즘의 입력으로 사용한다.

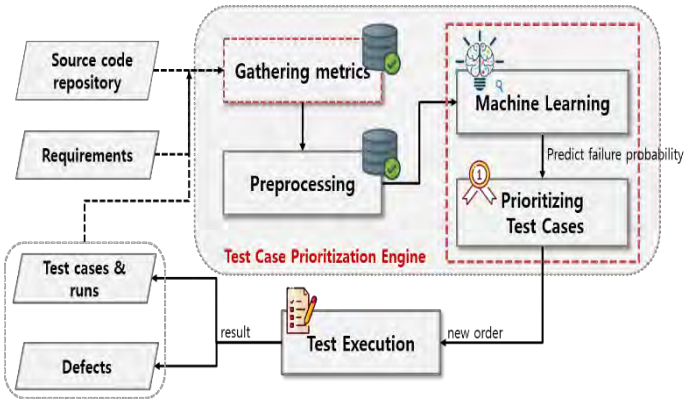


그림 2. 시스템 구조

3.2 소프트웨어 산출물 간 연결 정보와 메타 데이터

우리는 코드, 요구 사항, 테스트 케이스, 결함의 메타 데이터를 사용하는 테스트 케이스 우선 순위 지정

방법을 제안한다. 따라서, 각 산출물 간의 연결 정보는 매우 중요한 요소이다. 우리는 5개의 산출물 간 연결 정보를 사용한다.

- 요구 사항과 관련된 테스트 케이스 연결 정보
- 요구 사항과 관련된 소스 코드 저장소 연결 정보
- 요구 사항과 관련된 SAD(Software Architectural Design) 연결 정보
- 테스트 케이스와 관련된 테스트 실행 연결 정보
- 테스트 케이스와 관련된 결함 연결 정보

이 정보들을 바탕으로 우리는 소프트웨어 결함에서 코드, 그리고 반대로 코드에서 소프트웨어 결함까지 파악 할 수 있다. 5개의 산출물 간 연결 정보와 각 산출물의 정보를 이용해서 총 16개의 메타 데이터를 식별한다. 이 중 3개는 다른 테스트 케이스 우선 순위 지정 방법에서 사용하지 않았던 메타 데이터이다. 이 연구에서 사용하는 전체 메타 데이터는 [표1]과 같다. 이 연구에서 새롭게 제안하는 메타 데이터는 [proposed] 로 구분한다.

Category	메타 데이터
요구 사항	요구 사항 변경
	New 요구 사항
	관련 기능(Component)
코드	코드 변경
	SAD 수 [proposed]
테스트 케이스	테스트 케이스 변경
	New 테스트 케이스
	테스트 케이스 절차 수
	테스트 환경 [proposed]
	필요한 주변 기기 수
	Historical 실행 결과
	Weighted 테스트 커버리지
결함	결함 수
	결함 심각도(Severity)
	Reopened 수 [proposed]
	결함 상태

표 1. 소프트웨어 산출물 별 메타 데이터

3.2.1 요구 사항 메타 데이터

소프트웨어 요구 사항에서 3개의 메타 데이터를 수집한다.

1) 요구 사항 변경: 이전 소프트웨어 버전과 새로운 소프트웨어 버전 사이에 요구 사항의 내용(description)이 변경 되었는지 식별한다 [2, 10].

Description	Scale
변경 있음	1
Else	0

표 2. 요구 사항 변경 메타 데이터 값

2) New 요구 사항: 새롭게 구현되어 신규 소프트웨어 버전에 처음으로 포함된 요구 사항을 식별한다 [2].

Description	Scale
New 요구 사항	1
Else	0

표 3. New 요구 사항 메타 데이터 값

3) 관련 기능(Feature): 각 요구 사항이 어떤 기능(Radio, Navigation, Bluetooth, etc)과 관련 있는지 식별한다 [6, 12]. 이 데이터는 One-hot encoding 해서 사용한다.

3.2.2 코드 메타 데이터

코드에서 2개의 메타 데이터를 수집한다.

1) 코드 변경: 요구 사항과 관련된 코드가 변경 되었는지 식별한다 [4, 11]. 이를 위해서 새로운 소프트웨어의 Release note를 참고한다. Release note는 특정 소프트웨어의 변경 사항을 기록 및 설명하는 문서이다. 이 문서에는 소스 코드 저장소(repository) 마다 commit 된 변경 사항: source file, code diff, commit message 등을 확인 할 수 있다.

Description	Scale
요구 사항의 코드 변경	1
Else	0

표 4. 코드 변경 메타 데이터 값

2) SAD(Software Architecture Design) 수[proposed]: 코드와 관련된 소프트웨어 디자인 문서들의 수를 식별한다. 요구 사항 및 코드 저장소와 관련된 SAD 문서의 수가 많을수록 해당 코드 저장소는 시스템에 미치는 영향력이 크다고 판단 할 수 있다. 이 값은 정규화 하여 사용한다.

3.2.3 테스트 케이스 메타 데이터

테스트 케이스에서 7개의 메타 데이터를 수집한다.

1) 테스트 케이스 변경: 이전 소프트웨어 버전과 새로운 소프트웨어 버전 사이에 테스트 케이스의 내용(description) 또는 테스트 절차(Test step)이 변경 되었는지 식별한다 [10].

Description	Scale
변경 있음	1
Else	0

표 5. 테스트 케이스 변경 메타 데이터 값

2) New 테스트 케이스: 신규 소프트웨어 버전에서 처음으로 테스트 되는 테스트 케이스를 식별한다 [10].

Description	Scale
New 테스트 케이스	1
Else	0

표 6. New 테스트 케이스 메타 데이터 값

3) Weighted 테스트 커버리지: 요구 사항과 테스트

케이스, 그리고 결함의 심각도 정보를 이용해서 각 테스트 케이스의 weighted 요구 사항 커버리지를 계산한다 [8, 10]. 계산 방법은 아래와 같다.

4) 테스트 케이스 절차 수: 테스트 케이스의 테스트 절차(step) 수를 식별한다 [10]. 사전 절차(pre-condition)을 포함하여 계산한다. 이 값은 정규화 해서 사용한다.
5) 테스트 환경[proposed]: 어떤 테스트 환경에서 테스트 케이스를 실행하는지 식별한다. 자동차 소프트웨어 제품에서 많이 사용하는 테스트 환경은 Emulator, Bench, 자동차이다. 이 데이터는 One-hot encoding 해서 사용한다.
6) 필요한 기기 수: 테스트 케이스를 실행하기 위해서 필요한 주변 기기(Phone, CAN, etc) 의 수를 식별한다 [2]. 이 값은 정규화 해서 사용한다.

Description	Scale
기기 수	1 ~ N

표 7. 테스트에 필요한 기기 수 메타 데이터 값

7) Historical 실행 결과: 이번 테스트 실행 차수를 N차로 표현했을 때 테스트 케이스의 최근 N-1차 ~ N-5차 실행 결과를 식별한다 [1]. 테스트 케이스가 최근에 몇 번 실패 했는지 파악한다.

Description	Scale
테스트 실패	1
테스트 성공	0
초기값	0.5

표 8. Historical 실행 결과 메타 데이터 값

3.2.4 결함 메타 데이터

결함(defect)에서 4 개의 메타 데이터를 수집한다.

1) 결함 수: 각 테스트 케이스에서 찾은 결함 수를 식별한다 [10]. 이 값은 정규화 후 사용한다.
2) 심각도(Severity): 테스트 케이스에서 찾은 결함의 심각도 정보를 식별한다 [2, 10, 11]. 기준은 [표 9] 와 같다. 테스트 케이스에 여러 개의 결함이 있다면, 결함의 심각도 중 가장 큰 값을 사용한다. 이 값은 정규화 후 사용한다

Severity	Scale
Critical	4
Major	3
Minor	2
Comment	1
None	0

표 9. 결함 심각도 메타 데이터 값

3) Reopened 수[proposed]: 결함이 개발자에 의해 개선 된 이후에 몇 번이나 다시 발생(reopen)되는지 식별한다. Reopened 수가 많다는 것은 이 결함과 관련된 요구 사항이 다른 요구 사항과 관련성이 높거나 영향을 많이 받는 다고 할 수 있고, 또한 개발자들이 실수를 많이 하는 요구 사항이라고 할 수 있다. 이 데이터는 결함의 상태가 “Closed” 또는 “Resolved” 에서 “Open” 으로 변경된 수로 식별한다. 테스트 케이스에 여러 개의 결함이 있다면, 결함의 reopened 수 중 가장 큰 값을 사용한다. 이 값은 정규화 후 사용한다.

Input
1. 요구 사항: r_i,
2. 전체 요구 사항 수: n
3. 요구 사항과 관련된 테스트 케이스: t_j

Output
각 테스트 케이스의 요구 사항 커버리지

Procedure
Step1: 요구 사항과 테스트 케이스를 수집한다.
Step2: Step2.1~2.2를 $i \le n$ 까지 반복한다.
Step2.1: r_i의 테스트 케이스 수 k를 구한다.
k = r_i에 연결된 테스트 케이스 수
Step2.2: 각 요구 사항 r_i 와 테스트 케이스 t_j 의 커버리지를 계산한다.
$$\mathrm{cov}(r_i, t_j) = 1 \ / \ k$$
Step3: 테스트 케이스 t_j의 전체 커버리지를 계산한다.
$$\mathrm{cov}(t_j) = \sum_{i=1}^{n} cov(r_i,\ t_j)$$
Setp4: weighted $\mathrm{cov}(t_j)$를 계산한다.
$$Wt\_cov(t_j) = \mathrm{cov}(t_j) * \text{avg. defect severity of } t_j$$

4) 결함 상태: “Resolved”는 개발자가 결함을 개선했고 테스트 엔지니어의 확인이 필요한 상태이다. 즉, 테스트 결과가 실패에서 성공으로 변경 될 수 있는 확률이 높아진다고 할 수 있다. 또한, “Closed” 상태이고 관련 코드가 변경되었다면 관련된 테스트 케이스의 실행 결과가 실패로 변경 될 수 있는 확률이 높아진다. 그래서, 결함 상태는 테스트 케이스의 실행 결과를 예측하기에 매우 중요한 요소 중에 하나이다 [6]. 테스트 케이스에 여러 개의 결함이 있다면, 결함의 상태 중 가장 큰 값을 사용한다. 이 값은 정규화 후 사용한다. 식별 기준은 [표 10]과 같다.

Status	Scale
Open	3
Resolved	2
Closed	1
None	0

표 10. 결함 상태 메타 데이터 값

4. 실 험

머신 러닝을 사용한 코드와 요구 사항 기반 테스트 케이스 우선 지정 방식의 효과를 검증하고 입증하기 위해 진행한 실험에 대해 설명한다. 이 실험을 통해 기존의 요구 사항 기반 방식에 코드 메타 데이터를

추가 했을 때 APFD 값이 높아지는지 확인한다. ASPICE 개발 프로세스를 준수하는 자동차 개발 프로젝트의 데이터를 실험에 사용했다. 머신 러닝 알고리즘에 따른 성능 차이를 평가하기 위해 SVM(Support Vector Machine), Decision tree 기반의 XGBoost(eXtreme Gradient Boosting), FCNN(Fully Connected Neural Network) 등으로 실험을 진행하여 가장 높은 성능의 모델을 프로젝트에 적용했다.

4.1 실험 절차

실험 절차에 대해 설명한다. 그림 3 은 실험 절차의 전체 구조를 보여준다. 신규 소프트웨어 버전이 Release 되면 회귀 테스트를 위한 테스트 세트를 구성하고 테스트 케이스 우선 순위 지정을 진행한다.

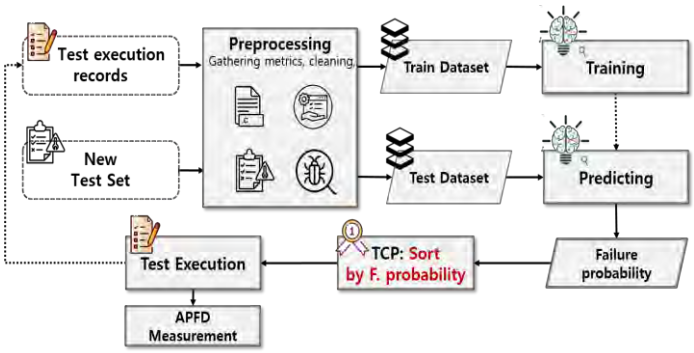


그림 3. 실험 절차

신규 소프트웨어의 테스트 세트를 실험 데이터로 사용하고 테스트 케이스의 과거 실행 결과를 훈련 데이터와 검증 데이터로 사용한다. 또한, 아주 오래된 테스트 케이스 결과로 신규 소프트웨어의 테스트 결과가 예측되지 않도록 학습 데이터로 사용하는 과거 테스트 케이스 실행 결과의 수를 조정한다. 학습된 머신 러닝 모델에 새로운 소프트웨어 버전에서 실행 할 테스트 세트를 입력하여 각 테스트 케이스의 실패 확률을 예측하고 실패 확률이 높은 순으로 테스트 케이스를 정렬하여 APFD 값을 계산하여 성능을 측정한다.

4.2 데이터 세트

이 실험에 사용되는 훈련 데이터 세트와 검증 데이터, 그리고 실험 데이터 세트의 구성은 [표 11]과 같다. 실험 데이터 세트는 신규 소프트웨어 버전에서 실행할 N 차 테스트 케이스이다.

데이터 세트	Description
Train (85%), Validation(15%)	9,800 개
Test	신규 테스트 세트의 테스트 케이스 수

표 11. 데이터 세트

4.3 Baseline

요구 사항 기반 방식에 비해 제안한 방식의 효과가 좋음을 증명하기 위해 요구 사항 기반 방식을 baseline 으로 설정한다. Baseline 모델은 코드 메타 데이터와 이 연구에서 새롭게 제안한 메타 데이터를 제외해서 APFD 값을 측정한다. Baseline 평가는 이번 연구에서 제안한 방식보다 성능이 좋지 않음을 확인하기 위함이므로 SVM(Support Vector Machine) 모델만을 이용해서 t-test 를 진행하였고, APFD 결과는 [표 12]와 같다.

APFD (%)		
Test	Baseline	Proposed
#1	74.93	76.72
#2	75.05	76.67
#3	74.79	76.46
#4	75.1	76.46
…		
#27	74.63	76.22
#28	74.58	76.31
#29	74.71	76.24
#30	75.93	77.65
평균	74.84	76.29

표 12. Baseline 과 Proposed 의 결과

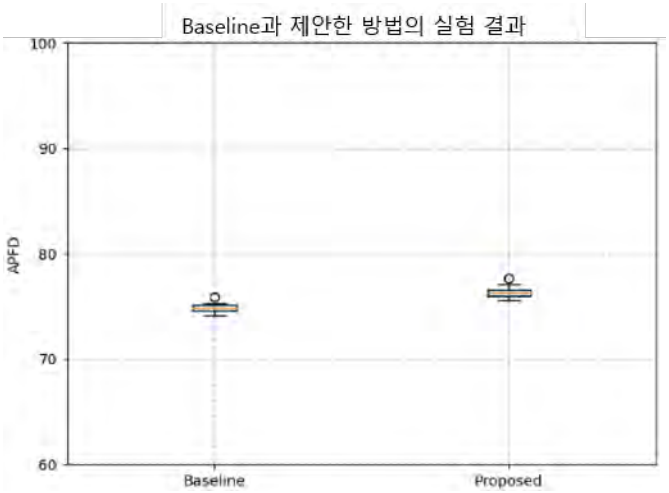


그림 4. 요구사항 vs 제안한 방식 APFD by SVM

위 표와 그림을 보면 제안한 방식의 평균이 Baseline 보다 더 높음을 알 수 있다. 또한, p-value 는 2.445e-20 으로 두 방식의 평균에는 차이가 있다는 가설을 채택할 수 있다. 이를 통해 Baseline 인 요구 사항 기반보다 제안한 방식의 성능이 더 높다고 할 수 있다. 다음은 제안한 메타 데이터를 모두 사용해서 머신 러닝 모델의 성능을 비교하고 여러 모델 중 가장 좋은

모델을 채택하여 프로젝트에 적용한 실험 결과에 대해 설명한다.

4.4 실험 결과

제안한 방식이 신뢰 할 수 있는지 평가하기 위해서 다양한 머신 러닝 모델에 데이터의 분포를 다르게 해서 실험을 각 30 회 진행해서 성능을 측정했다. 비교 실험을 진행한 머신 러닝 모델들의 APFD 값은 [표 13]과 [그림 6]과 같다.

APFD (%)					
Test	SVM	XGBoost	SVM + XGBoost	FCNN	RNN
#1	76.72	77.82	77.57	77.21	76.85
#2	76.67	77.56	77.45	76.55	76.33
#3	76.46	77.57	77.41	76.74	76.6
#4	76.46	77.61	77.42	76.73	76.33
...					
#27	76.22	77.39	77.23	76.38	75.49
#28	76.31	77.72	77.44	76.94	76.17
#29	76.24	77.24	77.12	76.2	76.37
#30	77.65	79.78	78.95	79.91	75.82
평균	76.29	77.57	77.33	76.69	76.26

표 13. 모델 비교 실험 결과

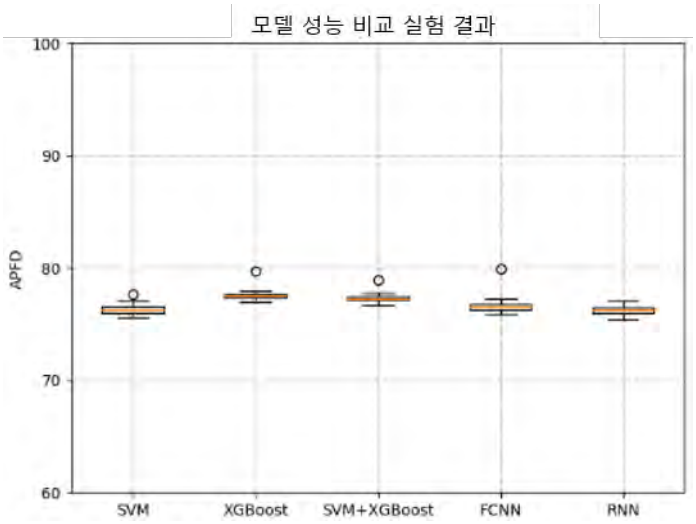


그림 5. 모델 성능 비교 실험 결과

[그림 5]의 그래프는 머신 러닝 모델 별 APFD 분포를 나타낸다. 머신 러닝 모델의 APFD 차이는 크지 않지만 XGBoost 모델이 가장 좋은 성능을 보여주고 있다. 머신 러닝 모델의 비교 결과를 바탕으로 XGBoost 모델을 개발 프로젝트에 적용하여 총 9 개의 신규 소프트웨어 버전의 테스트 세트에 제안한 방식으로 테스트 케이스 우선 순위를 지정하였다. 또한, 현재 이 프로젝트에서는 Rule 기반 테스트 케이스 우선 순위 지정 방법을 사용하고 있었다. 그래서, 현재 방법과 제안한 방법의 차이를 보여주고자 비교 실험을 진행했고, 결과는 [표 14]와 같다.

SW Version	Rule	Baseline	Proposed using XGBoost
#1	43.93	71.3	74.27
#2	43.97	71.94	73.26
#3	44.49	73.87	76.32
#4	41.79	73.86	78.28
#5	41.99	70.62	73.56
#6	42.62	71.22	74.61
#7	36.73	73.27	77.65
#8	45.88	75.7	79.79
#9	43.44	76.99	79.93
평균	42.76	73.2	76.41

표 14. 프로젝트 적용 결과

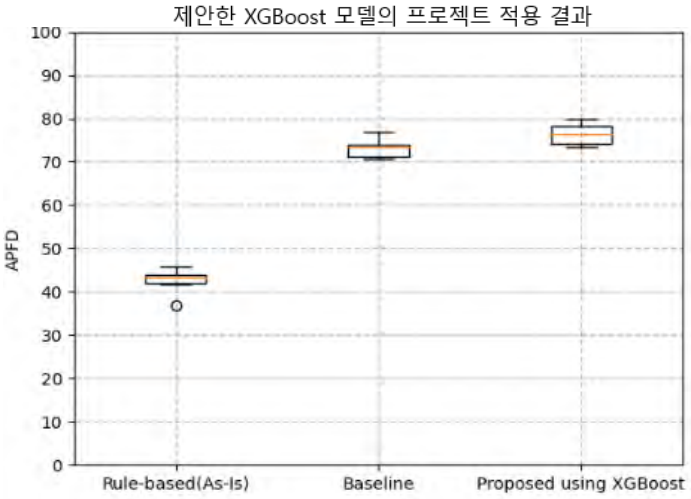


그림 6. 프로젝트 적용 결과

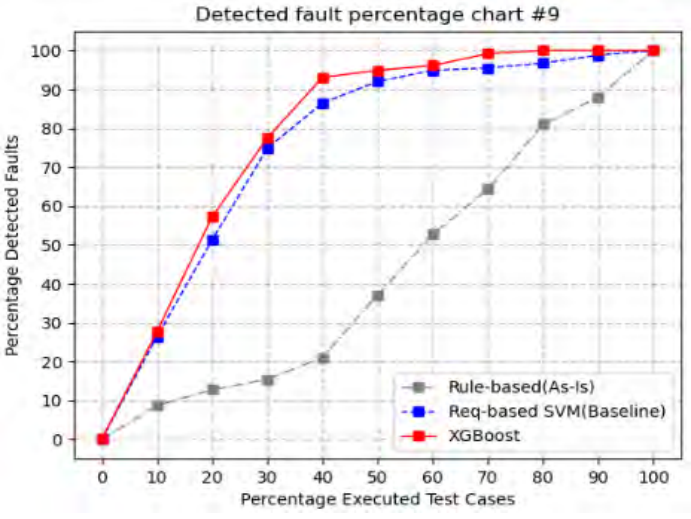


그림 7. 프로젝트 적용 결과

[표 14]와 [그림 6, 7]에서 보듯이 현재 사용 중인 Rule 기반 방법의 APFD 값은 36%~46%의 분포를 Req 기반인 Baseline 의 APFD 값은 70% ~ 77%의 분포를 각각 보여주고 있다. 반면에, 우리가 제안하는 메타 데이터를 사용하고 XGBoost 모델을 적용한 TCP 방법의 APFD 값은 73%~80%의 분포를 보여주고 있다. 또한, 실험을 진행한 9 개의 소프트웨어 버전에서 모두 Rule 기반 방법과 요구 사항 기반 방법의 APFD 값 보다 각각 30% 와 4%넘게 높아졌음을 확인 할 수 있다. 따라서 우리가 제안하는 코드와 요구 사항 기반의 방법이 다른 방법들 보다 안정적이고 효과적인 방법임을 알 수 있다.

5. 결 론

이번 연구에서는 소프트웨어 산출물의 추적성이 확보된 프로젝트에서 코드와 요구 사항의 메타 데이터를 사용해서 테스트 케이스 우선 순위를 지정하는 방법을 제안하였다. 기존의 요구 사항 기반 방법에서는 추적성이 확보되지 않아 사용 할 수 없었던 코드 메타 데이터와 우리가 제안한 새로운 메타 데이터를 사용하여 테스트 케이스의 실패 확률을 예측하는 시스템을 개발하였다. 비록 다양한 프로젝트의 데이터를 이용해서 비교 실험을 하진 못했지만, 이를 극복하기 위해서 여러가지 머신 러닝 모델과 테스트 세트를 이용해서 다양한 실험을 진행했다. 또한, 현재 개발 중인 자동차 소프트웨어 프로젝트의 데이터를 실험에 사용했기 때문에 우리가 진행한 실험이 높은 신뢰도를 가진다고 할 수 있고, 기존 방법에 비하여 성능이 높다는 것도 확인 할 수 있었다.

앞으로는 이번 연구에서 추적성이 확보되지 않아 적용하지 못했던 코드 관련 메타 데이터들을 추가하고 강화 학습 등의 모델에 적용하여 성능을 향상 시키고, 여러 프로젝트의 다양한 테스트 레벨에 적용해서 우리의 연구를 확대 할 계획이다.

References

[1] Remo Lachmann, Manuel Nieke, Christoph Seidl, Ina Schaefer, “System-Level Test Case Prioritization Using Machine Learning”, IEEE International Conference on Machine Learning and Applications, 2016.

[2] Md. Hasan Mahmood, Md. Shazzad Hosain, “Improving Test Case Prioritization Based on Practical Priority Factors”, IEEE, 2017.

[3] Gregg Rothermel, Roland H. Untch, Chengyun Chu, and Mary Jean Harrold, “Prioritizing Test Cases For Regression Testing”, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 2001.

[4] Xiaobin Zhao, Zan Wang*, Xiangyu Fan, “A Clustering – Bayesian Network Based Approach for Test Case Prioritization”, IEEE Annual International Computers, Software & Applications Conference, 2015.

[5] ASPICE, http://www.automotivespice.com/fileadmin/software-download/AutomotiveSPICE_PAM_31_Korean.pdf

[6] Nilam Kaushik, Mazeiar Salehie, Ladan Tahvildari, Sen Li and Mark Moore, “Dynamic Prioritization in Regression Testing”, International Conference on Software Testing, Verification and Validation Workshops, 2011.

[7] Remo Lachmann, “Machine Learning-Driven Test Case Prioritization Approaches for Black-Box Software Testing”, The European Test and Telemetry Conference, 2018.

[8] Rimsha Butool, Aamer Nadeem, Muddassar Sindhu, Qamar uz Zaman, “Improving Requirements Coverage in Test Case Prioritization for Regression Testing”, 22nd International Multitopic Conference (INMIC), 2019.

[9] Yilin Yang, Xinhai Huang, Xuefei Hao,Zicong Liu and Zhenyu Chen, “An Industrial Study of Natural Language Processing Based Test case Prioritization”, IEEE International Conference on Software Testing, Verification and Validation, 2017

[10] Mazeiar Salehie, Sen Li, Ladan Tahvildari, Rozita Dara, Shimin Li, Mark Moore, “Prioritizing Requirements-Based Regression Test Cases: A Goal-Driven Practice”, European Conference on Software Maintenance and Reengineering, 2011

[11] Kathrin Land, Eva-Maria Neumann, Simon Zeigltrum, Huaxia Li, Birgit Vogel-Heuser, “An Industrial Evaluation of Test Prioritisation Criteria and Metrics”, IEEE International Conference on Systems, Man and Cybernetics (SMC), 2019

[12] Dennis S´avio Silva, Ricardo Rabelo, Pedro Santos Neto, Ricardo Britto and Pedro Almir Oliveira, “A Test Case Prioritization Approach Based on Software Component Metrics”, IEEE International Conference on Systems, Man and Cybernetics (SMC), 2019

소프트웨어 결함 분석을 위한 딥 메트릭 러닝

강민구○, 최지원, 류덕산, 김순태
전북대학교 소프트웨어공학과
{202050166, jiwon.choi, duksan.ryu, stkim} @jbnu.ac.kr

Deep Metric Learning for Software Defect Analysis

Mingu Kang, Jiwon Choi, Duksan Ryu, Suntae Kim
Dept. of Software Engineering, Jeonbuk National University

요 약

소프트웨어 결함은 소프트웨어 공학 분야에서 해결해야 하는 가장 큰 위협 중 하나로, 머신 러닝 알고리즘을 사용하여 결함을 식별하고 수정하기 위해 다양한 접근 방식이 제안되었다. 최근 소프트웨어의 결함을 식별하기 위해 프로그래밍 언어로 작성된 소스코드 파일을 이미지 파일로 변환하는 접근 방식이 제안되었다. 이 제안된 방법은 기존 방법 대비 우수한 성능을 보였다.본 논문은 컴퓨터 비전 기반의 접근 방법이 소프트웨어 결함 데이터 분석 분야에 적용될 수 있는지 확인한다. 이를 위해 결함과 비 결함 이미지의 분포 차이를 통계적인 기법과 딥 메트릭 러닝, 이미지 유사도 비교 기법을 사용해 실험적으로 보인다. 실험 결과 딥 메트릭 러닝 방법은 결함 이미지와 비 결함 이미지를 더 잘 구분해내며 분류와 클러스터링의 정확도를 향상시킬 수 있을 것으로 기대된다.

1. 서 론

소프트웨어 결함 예측의 목표는 결함인 소프트웨어 모듈을 가능한 많이 찾아내서 한정된 자원을 효율적으로 분배하는 것이다[1]. 결함 여부를 예측하기 위해 개발된 모델을 사용하여 코드 영역(파일, 메소드 등)에서 결함이 있는 영역을 판별한다[2].

이전 연구는 코드의 정적 피처(feature)가 결함과 관련이 있는지 조사하고, 특정 코드의 피처를 측정하는 소스코드 메트릭을 정의 및 평가하는데 초점을 맞추고 있다[4]. 대표적인 소스코드 메트릭은 코드 라인(LOC), 객체 지향 메트릭(OO), McCabe의 복잡성이 있다. 동일한 메트릭 값을 가지고 있지만 결함 측면에서 동일한 라벨이 지정되지 않는 경우가 발생할 수 있다. 이는 메트릭 방법이 소스코드의 결함 예측을 하기에 충분하지 않음을 시사한다[4]. 코드의 결함 있는 피처를 식별하기 위해 추상 구문 트리 방법(AST)을 사용한 접근법이 제안되고 있다. AST는 소스코드의 구조적 정보를 포함하는 트리 형태의 데이터 구조이다[3]. 하지만 일부 프로젝트에서 AST는 숨겨진 피처와 같은 의미 있는 코드 정보가 효과적으로 표현되지 않는 경우가 종종 있다[5]. Elish et al.[6]의 연구에서는 의미 있는 코드 정보는 하나의 코드 영역을 다른 코드 영역과 구별하기 위한 구문 정보보다 더 중요함을 보였다. 따라서 AST가 전달하는 피처는 결함 예측에 유용하지만 AST를 구축하기 위해 추가적인 방법들이 필요하다.

최근 딥 러닝 네트워크는 이미지 분류 및 피처 추출에 매우 효과적임을 많은 연구에서 실험적으로 보였다[7-8]. Shippey et al.[4] 연구에서는 AST에서 추출한 토큰 벡터에서 피처를 학습하기 위해 심층 신뢰 신경망(Deep belief network)을 활용했다.

소프트웨어 시각화(Software Visualization)는 소프트웨어 공학에서 코드 실행 및 피처를 시각화하는데 사용되어 왔다[9]. Chen et al.[5] 연구는 AST와 같은 중간 표현을 피하고 소스코드를 이미지화시켜 의미 있는 정보를 직접 얻는 방법을 제안했다. 제안한 방법은 프로젝트 내 결함 예측(WPDP)과 교차 프로젝트 결함 예측(CPDP)에서 성능이 우수함을 보였다. 특히 CPDP에서 최신 CPDP 기법들 대비 성능 개선에 효과가 있음을 강조한다.

본 연구에서는 Chen et al.[5] 연구를 기반으로 소스코드를 이미지로 표현했을 때 결함과 비 결함 이미지의 분포에 차이가 있음을 보인다. 딥 메트릭 러닝은 기존의 피처로는 분류가 어려웠던 데이터에 대해 데이터의 라벨별로 구분될 수 있게 만들어주는 방법이다. 딥 메트릭 러닝을 사용하면 결함과 비 결함 데이터가 보다 더 잘 구분되는 모습을 볼 수 있다. 본 연구에서는 딥 메트릭 러닝을 사용해 결함과 비 결함 데이터가 구분됨을 보인다. 추가적으로 통계적인 분석과 이미지 유사도 측정 방법을 사용해 실험 결과에 신빙성을 더한다. Chen et al.[5] 연구에서는 제안한 방법이 기존 기법 대비 성능 향상이 있는 이유에 대해 t-SNE 시각화 기법으로 결함과 비 결함 이미지의 분포 차이가 있음을 보였다. 본 연구에서는 결함과 비 결함 이미지의 분포 차

이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2019R1G1A1005047, NO.2020R1F1A1072039)

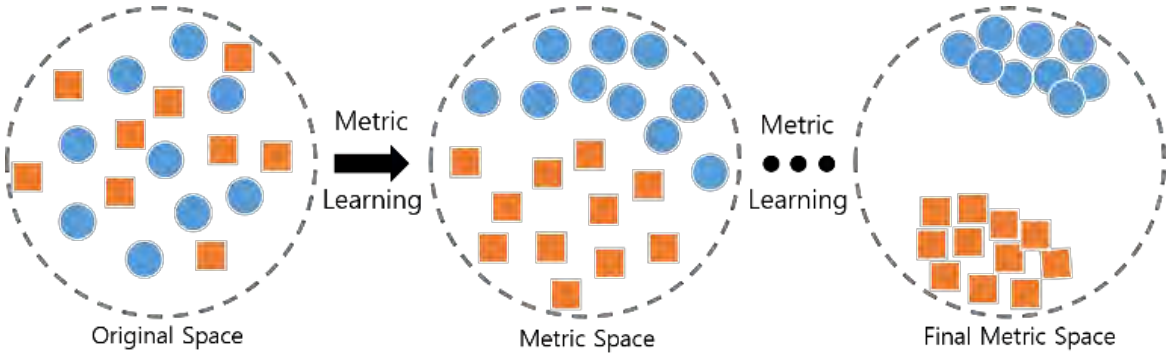


그림1. 딥 메트릭 러닝 과정

이를 다른 접근 방법으로 검증한다. 결함과 비 결함 이미지의 분포 차이를 확인해야 하는 이유는 두 이미지 간의 분포 차이가 존재하면 소프트웨어 결 함 예측 성능을 향상시키기 위해 다양한 이미지 기법들을 소프트웨어 결함 예측 연구에 적용할 수 있기 때문이다. 실험을 위해 소스코드를 이미지로 변환한다. 변환시킨 이미지의 RGB 값 순서를 조합해 이미지 데이터를 증강시킨다. 본 연구에서는 결함 데이터와 비 결함 데이터가 효과적으로 구분됨을 보이기 위해 딥 메트릭 러닝 기법 중 Large margin cosine loss[15]를 사용한다. 또한 컴퓨터 비전 기반의 이미지 유사도 측정 방법인 평균 제곱근 편차(Root Mean Square Error, RMSE)[19] 와 최대 신호 대 잡음비(Peak Signal-to-Noise Ratio), 구조적 유사성 측정(Structural Similarity, SSIM) 방법[16]을 사용한다. 결함과 비 결함 이미지의 픽셀 분포를 사용해 통계적인 기법으로 결함과 비 결함 이미지 간에 차이가 있음을 실험적으로 보인다. 딥 메트릭 러닝 적용 결과, 결함 이미지와 비 결함 이미지가 딥 메트릭 러닝 적용 전 보다 명확하게 구분되는 모습을 시각적으로 확인할 수 있다. 또한 컴퓨터 비전 기반의 이미지 유사도 측정 방법과 통계적인 기법을 적용한 결과, 결함과 비 결함 이미지 사이에 분포 차이가 있음을 유의적으로 보였다.

실험 결과를 통해 결함과 비 결함 데이터들이 컴퓨터 비전 기반의 방법으로 분류될 수 있음을 보였다. 이러한 실험 결과는 분류 및 클러스터링의 정확도를 향상시키는데 적용될 수 있을 것으로 기대된다.

2. 관련 연구

소프트웨어 결함 예측은 소프트웨어 소스코드의 잠재적인 결함 영역을 확인한다. 결함 예측을 통해 개발자는 출시 전에 소프트웨어 시스템의 영역에 초점을 맞출 수 있어 결함을 찾는 시간과 노력을 줄일 수 있다[4].

결함 예측 성능 향상을 위해 이전 연구들은 소스코드의 피처를 식별하고 평가하기 위한 많은 피처를 수동으로 설계했다. 예를 들어 Halstead 척도[10]는 연산자 및 피연산자 수를 기반으로 하고, McCabe 척도[11]는 함수 및 상속 수를 기반으로 하고, Chidamber and Kemerer (CK)척도[22]는 커플링 인자 등을 기반으로 한다. 하지만 시스템 전반적으로 문제가 있는 부분의 코드를 일관되게 식별하는 정적 코드의 피처는 없다[12]. 결함을 나타내는 코드의 피처가 시스템마다 다르고[13], 소스코드의 의미 있는 정보가 숨겨져 있지만, 전통적인 피처들은 종종 이를 포착하지 못한다. 따라서 결함이 있는 코드를 나타내는 피처들을 식별하는 방법을 찾아내는 것은 중요하다.

최근 딥 러닝은 자동화된 피처 생성을 위한 기법으로 사용되고 있다. 딥 러닝의 아키텍처는 고도로 복잡한 비선형 피처를 효과적으로 포착할 수 있다. Wang et al.[2]연구에서는 프로그램의 의미 표현을 자동으로 학습하기 위해 딥 러닝을 활용할 것을 제안했다. Shippey et al.[4] 연구에서는 프로그램의 AST에서 추출한 토큰 벡터에서 피처를 학습하기 위해 심층 신뢰 신경망(Deep belief network)을 활용했고, 결함 예측에서 전통적인 피처 기반 접근법보다 우수한 성능을 실험적으로 보였다.

소프트웨어 시각화는 소프트웨어 엔지니어링 연구 및 실무에서 오랜 기간 사용되어왔다. 대표적인 사례로 버그 저장소(repository)에도 적용되었고, 버그를 코드 구조와 연관시키는 데 도움을 주었다[14]. Chen et al.[5] 연구는 소프트웨어 결함 예측에 소스코드 시각화를 접목해 프로젝트 간의 이미지 및 구조적 유사성을 효과적으로 식별하고자 했다. 먼저 각 프로젝트의 소스코드 파일을 이미지로 변환시킨 후 제안한 기법인 DTL-DP(Deep Transfer Learning for Defect Prediction)를 사용한다. DTL-DP는 end-to-end 프레임워크이며, 새로운 소스코드 파일이 결함을 포함하고 있는지 예측한다. 제안한 기법은 프로젝트 내 결함 예측(WPDP) 보다 교차 프로젝트 결함 예측(CPDP)에서 성능이 우수함을 강조한다. 교차 프로젝트 결함 예측(CPDP)의 최신 연구와 성능을 비교한 결과 성능 개선 효과가 있음을 실험적으로 보였다. 연구에 따르면 코드를 이미지로 변환시키는 이점을 두 가지로 정리하고 있다.

첫째, 소스코드의 구조적인 특징뿐만 아니라 의미 있는 정보를 유지할 수 있다. 따라서 이미지 피처 정보를 추출할 수 있는 데 보다 최적화 되어있는 딥 러닝 네트워크 방법을 적용할 수 있다.

둘째, 실험에 가용할 데이터세트를 사용해 동일한 의미를 지니는 보다 큰 데이터세트를 생성할 수 있다. 특히 딥 러닝 모델의 경우 훈련 데이터가 많을수록 좋은 성능을 기대할 수 있다. 이미지 데이터세트의 일반적인 데이터 증강 방법에는 수직 및 수평으로 뒤집는 방법과 회전, 자르기, 가우스 노이즈 추가 등의 방법이 있다. Chen et al.[5] 연구는 이미지가 가진 R, G, B 3채널의 순서를 조합하여 새로운 이미지를 만들어내는 방법으로 데이터 증강을 이뤘다.

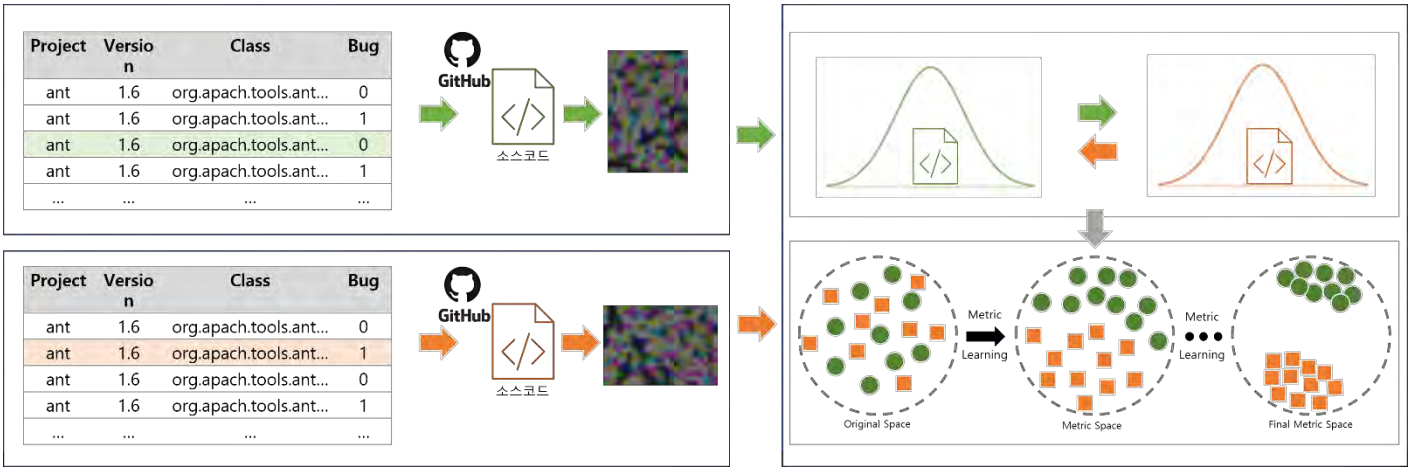


그림2. 소스코드 이미지 분석 방법

3. 배경 지식

3.1 메트릭 러닝 기법

메트릭 러닝은 객체 간의 유사성 또는 차이점을 확립하는 것을 목표로 하는 거리 측정법에 기반을 둔 학습 기법이다[20]. 이러한 방법은 데이터에서 적절한 메트릭을 자동으로 학습할 수 있어 기계 학습 및 데이터 마이닝에 널리 사용된다. 데이터 샘플 간 유사성을 측정하는 메트릭으로는 유클리디안(Euclidean)거리와 마할라노비스(Mahalanobis) 거리 측정방법, 쿨백 라이블러(Kullback-Leibler)[21] 등이 있다. 이러한 전통적인 방법은 비선형 정보를 반영하지 못하고 하나의 메트릭 공간으로 변환시킨다.

3.2 딥 메트릭 러닝 기법

3.1 절에서 언급한 전통적인 메트릭 러닝의 한계점을 극복하기 위해 딥 러닝을 접목한 연구가 활발히 진행되고 있다[20]. 그림1에서 확인할 수 있듯이 딥 메트릭 러닝은 주어진 데이터에서 적절한 메트릭을 자동으로 학습한다. 유사한 데이터 샘플은 가까워지고 유사하지 않은 데이터 샘플은 멀리 떨어지도록 학습하여 샘플을 잘 구분할 수 있는 메트릭 공간으로 변환시킨다. 비선형 피처를 표현하는 딥 러닝의 학습 능력과 메트릭 러닝의 특징을 활용하여 컴퓨터 비전 작업에 적용한다[15].

3.3 Large margin cosine loss

본 실험에서는 데이터를 목적 값에 따라 잘 구분되도록 변환해주기 위해 데이터 유사성 측정에 Large margin cosine loss[15]를 적용한다. 임베딩 벡터와 예측하고자 하는 클래스 간의 cosine 각도를 계산한다. 이때 margin 값을 더해 다른 클래스 간 거리를 벌려 주는 방향으로 모델을 훈련시키는 방법이다. 이를 통해 피처 벡터가 메트릭 러닝 기법의 목적에 맞게 동일 클래스 내에서는 더 잘 모여있고, 다른 클래스와는 더 멀리 떨어져 있을 수 있도록 임베딩 된다. 손실함수는 다음과 같이 정의된다.

$$L3 = \frac{1}{N}\sum_{i=1}^{N} -log\frac{e^{s(cos(\theta_{y_i}+m))}}{e^{s(cos(\theta_{y_i,i})-m)}+\sum_{j\neq y_i} e^{scos(\theta_{j,i})}} \quad \cdots\cdots (1)$$

(1)에서 m은 margin 크기를 조절해 주기 위한 고정된 파라미터이고, θ는 해당 샘플 데이터 클래스의 weight 벡터와 입력 데이터 벡터 사이의 각이다. s는 이를 scaling 해주기 위한 파라미터이다. 데이터 간의 각도에 정량적인 margin 값을 더함으로써 클래스 사이의 거리를 벌려준다.

3.2.3 이미지 유사도 비교 기법

이미지의 유사성을 평가하기 위해 주변 지점 또는 피처들의 관계를 설명하는 지표가 있다[17~19]. 본 연구에서는 컴퓨터 비전 기반의 이미지 유사성 기법을 사용해 결함 이미지와 비 결함 이미지 간의 차이가 있음을 보인다. 사용한 이미지 유사도 비교 기법은 평균 제곱근 편차와 최대 신호 대 잡음비, 구조적 유사성 측정 방법이다.

평균 제곱근 편차(Root Mean Square Error). 평균 제곱근 편차(RMSE)는 이미지 유사도 측정 지표의 일반적인 방법이다. 예측한 값과 실제 값 간의 차이를 측정하는데 사용되어 오류 크기를 평가한다. RMSE는 평균 제곱근 오차(Mean Square Error, MSE)의 제곱근의 형태를 취하고 있으며 추정치 θ에 대한 추정량 $\hat{\theta}$은 다음과 같이 정의된다.

$$\boldsymbol{MSE} = \frac{1}{MN}\sum_{n=0}^{M}\sum_{M=1}^{N}[\boldsymbol{I_1(m,n)} - \boldsymbol{I_2(m,n)^2}] \quad \cdots\cdots (2)$$

$$\boldsymbol{RMSE(\hat{\theta})} = \sqrt{\boldsymbol{MSE(\hat{\theta})}} \quad \cdots\cdots (3)$$

최대 신호 대 잡음비 (Peak Signal-to-Noise Ratio). 최대 신호 대 잡음비(PSNR)는 두 이미지 사이의 최대 신호 전력과 표현의 품질에 영향을 미치는 왜곡 노이즈 전력 간의 비율을 계산하는데 사용된다. 8비트 픽셀의 경우 픽셀의 채널 값은 0에서 255사이이다. 신호는 결함이 없는 이미지로 간주하고 노이즈는 오류가 있는 결함 이미지다. PSNR은 인간의 인식에 대한 대략적인 추정치로 일반적으로 30dB가 넘으면 두 이미지의 차이를 육안으로 감지하기 어렵다. PSNR은 다음과 같이 정의된다.

$$\boldsymbol{PSNR} = \boldsymbol{10log_{10}(peakval^2)/MSE} \quad \cdots\cdots (4)$$

여기에서 peakval (Peak Value)는 이미지 데이터의 최대

값으로 8비트 이미지 데이터일 경우 peakval은 255이다. 결함이 없는 이미지일 경우 작은 MSE 값을 가지며 더 큰 PSNR 값을 가진다.

구조적 유사성 측정(Structural Similarity Index). 구조적 유사성 측정(SSIM) 방법은 수치적인 에러가 아닌 인간의 시각적인 인식에 기반을 둔 유사도 평가 방법이다. 두 이미지는 동일한 크기로 비교되고 휘도, 대비 및 구조의 변화를 고려하여 구조적 유사성을 계산한다. 유사도 범위는 -1부터 1 사이의 값을 가지며 1에 가까울수록 비슷한 이미지임을 뜻한다. l, c, s 는 각각 휘도, 대비 및 구조를 나타내며 함께 아래와 같이 SSIM 이미지 유사도를 정의한다.

$$SSIM(A,B) = l(A,B) \cdot c(A,B) \cdot s(A,B) \quad \cdots\cdots (5)$$

표1. 파일 크기에 따른 이미지 사이즈

파일 크기	이미지 크기
<10 kB	32
10kB – 30kB	64
30kB – 60kB	128
60kB – 100kB	256
100kB – 200kB	384
200kB – 500kB	512
500kB – 1000kB	768
>1000kB	1024

4. 연구 방법

그림2는 전체적인 연구 방법을 나타낸다. PROMISE 저장소에 있는 데이터세트의 클래스 정보와 버그 유무를 추출해 Github 저장소에서 소스코드를 가져온다. 이후 소스코드를 이미지로 변환하여 실험을 진행한다.

이 절에서는 소스코드를 의미 및 구조적 기능을 내포할 수 있는 이미지로 변환하고 실험 방법에 관해 기술한다.

4.1 소스코드 이미지 변환

본 연구에서는 Chen et al.[5] 연구에서 제시한 방법과 동일하게 소스코드를 이미지로 변환한다. 소스코드의 3개의 문자 RGB 값으로 할당하여 이미지로 변환한다. 예를 들어, int main()은 9개의 character를 가지고 있으므로 3개의 픽셀이 만들어진다. 이를 모든 소스코드에 적용하여 RGB 3개의 채널을 갖는 이미지를 만들어낸다. 추가적으로 이미지 RGB 값의 순서를 조합하여 한 개의 소스코드에서 6개의 이미지를 만들어낸다. RGB 순서를 조합해 만들어진 데이터는 기존 데이터와 동일한 의미를 가진다. 이를 통해 데이터의 증강을 이룰 수 있다. 본 연구의 실험에서는 RGB 값의 순서를 조합한 데이터를 사용한다.

이미지의 크기는 소스코드 파일의 크기에 의해 결정된다. 소스코드 파일의 크기에 따른 이미지의 크기를 규격화 하기 위해 Xin Zhou et al.[23] 연구와 동일한 방법으로 진행한다. 표1은 소스코드 파일 크기에 따른 적합한 이미지 크기를 나타낸다. 본 실험에서는 표1의 내용과 동일하게 파일의 크기에 따른 이미지의 크기를 설정해주었다.

4.2 이미지 크기 조정

4.1절에서 만들어진 이미지의 크기는 결함과 비 결함 이미지를 비교하기에 적합하지 않다. 결함과 비 결함 이미지를 비교하기 위해 이미지의 크기를 동일하게 맞추었다. Chen et al.[5] 연구에서는 서로 다른 이미지의 크기를 동일하게 맞추기 위해 이미지의 중심점을 기준으로 224 * 224 사이즈로 크기를 변환한다. Chen et al.[5]은 이미지의 크기가 224*224 사이즈 보다 크면 이미지의 중앙을 중심으로 잘라낸다. 반면에 이미지가 224*224 사이즈보다 작으면 zero padding 해주어 이미지 사이즈를 동일하게 맞춘다.

본 실험에서는 Chen et al.[5] 방법과 다르게 이미지 보간 방법을 적용하여 이미지 크기를 224*224로 변환한다.

4.3 결함 및 비 결함 이미지의 분포 확인

결함 이미지와 비 결함 이미지 간의 분포 차이를 통계적으로 확인하기 위해 결함, 비 결함 그룹으로 분리한다. 이미지별 픽셀이 가진 0 ~ 255 사이의 값을 도수 분포로 나타낸다. 각 그룹이 가지는 도수 분포를 모두 합하여 그룹을 대표하는 분포를 만든다.

4.4 결함 및 비 결함 이미지 비교

결함과 비 결함 이미지를 명확하게 구분하기 위해 딥 메트릭 러닝을 사용한다. 딥 메트릭 러닝은 결함과 비 결함 이미지 데이터가 잘 구분될 수 있게 학습해준다. 또한 0~255 사이의 픽셀 빈도에 대한 분포를 바탕으로 카이 제곱 검정 기법을 사용한다 해당 평가 지표에 대한 자세한 설명은 5.3.1절에서 설명한다. 데이터가 이미지라는 점에서 컴퓨터 비전 기반의 이미지 유사도 비교 기법을 사용해 결함과 비 결함 이미지를 비교한다.

표2. 실험에 사용한 데이터세트 정보

Project	# of file	# of clean	# of buggy
ant-1.6	350	258	92
camel-1.4	839	695	144
poi-3.0	373	118	251
lvy-1.4	241	225	16
lucene-2.0	195	104	91
log4j-1.1	78	48	30
synapse-1.0	157	141	16
xalan-2.4	428	349	79
xerces-1.2	434	365	69

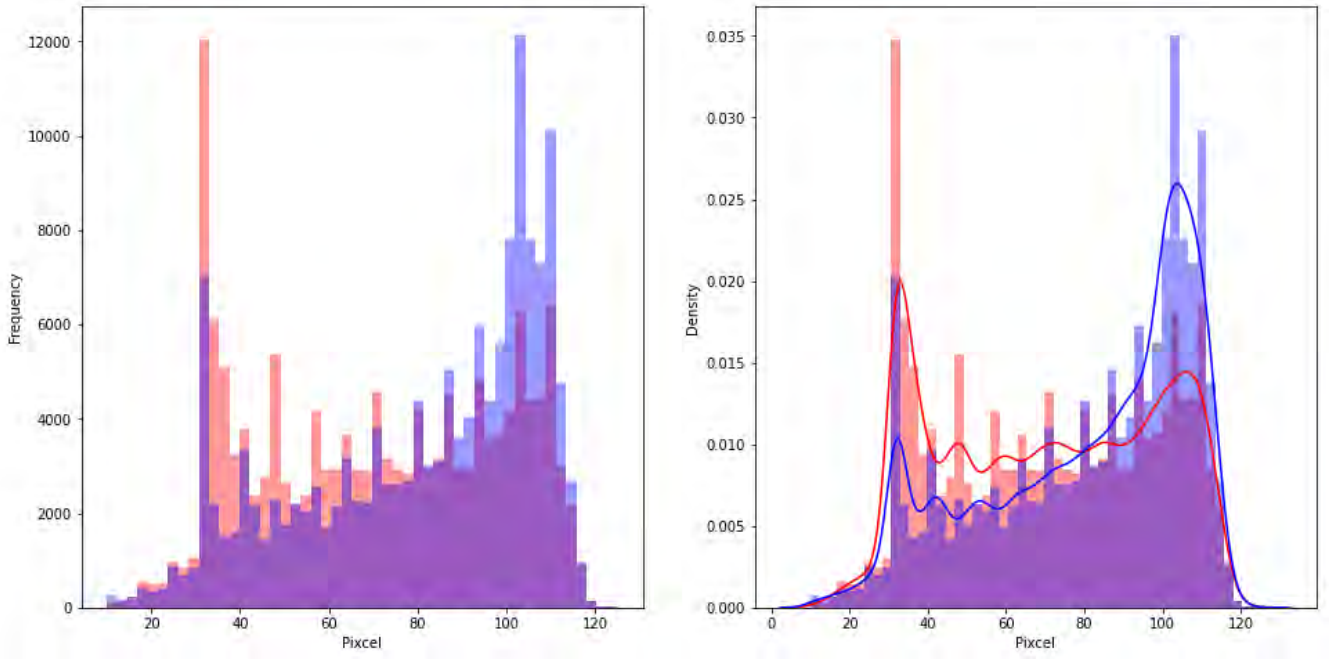
5. 실험 설정

5.1 연구 질문

소프트웨어 결함 예측을 위해 소스코드를 이미지로 변환했을 때 분포의 차이가 있는지를 확인하기 위해 다음과 같은 연구 질문을 갖는다.

RQ1. 결함 이미지와 비 결함 이미지의 분포에 차이가 있는가?

$-H_0$: 결함 이미지와 비 결함 이미지의 분포 차이가 없다.
$-H_A$: 결함 이미지와 비 결함 이미지의 분포 차이가 있다.



(a) (b)

그림3. 결함과 비 결함 이미지 픽셀 분포 비교

RQ2. 컴퓨터 비전 기반의 이미지 유사도 비교 기법을 활용해 결함 이미지와 비 결함 이미지의 분포 차이를 확인할 수 있는가?

RQ3. 딥 메트릭 러닝을 활용해 결함 이미지와 비 결함 이미지가 효과적으로 구분 지을 수 있는가?

5.2 데이터세트

PROMISE 저장소에서 총 9개의 프로젝트를 사용했다. PROMISE 데이터세트는 소프트웨어 결함 예측의 많은 연구에서 사용되는 데이터세트 이다.

PROMISE 저장소에서 우리는 ant, camel, poi, ivy, lucene, log4j, synapse, xalan, xerces 프로젝트를 선택해 실험에 사용했다. 우리는 PROMISE 저장소에서 프로젝트 버전과 클래스 이름, 결함 라벨만을 수집했다. 버전 정보와 클래스 이름을 통해 Github 저장소에서 자바 소스코드를 식별 및 수집하여 이미지로 변환했다. 표2는 실험에 사용한 프로젝트와 파일 수와 비 결함 파일 수, 결함 파일의 수를 보여준다. 프로젝트별로 가지고 있는 파일의 수가 적어 이미지 변환 시 R, G, B 순서를 조합하여 추가적인 데이터세트로 사용했다. 일부 소스코드 파일은 PROMISE 저장소의 클래스 이름과 Github 저장소의 클래스 이름이 동일하지 않아 데이터세트에서 제외했다.

5.3 평가 지표

5.3.1 통계적 검증 기법

카이 제곱 검정 기법은 관찰된 빈도가 기대되는 빈도와 유의하게 차이가 있는지를 검증하는 방법이다. 카이 제곱 검정 기법의 종류는 크게 세 가지로 동질성 검정과 독립성 검정, 적합도 검정 방법이 있다. 동질성 검정은 각각의 모집단에서 독립적으로 뽑은 표본들의 분포가 동일한지 검정한다.

본 연구에서는 결함 이미지와 비 결함 이미지의 분포가 동일한지 검증해주는 동질성(homogeneity) 검정을 사용하여 분포를 차이를 확인한다. 카이 제곱 검정량은 관측 빈도와 기대 빈도 차이의 변동을 정량화한 통계량이다. 자유도(df)는 해당 프로젝트의 이미지에서 사용된 픽셀의 수를 의미한다. 유의 수준 값이 0.05보다 작으면 귀무가설이 기각된다.

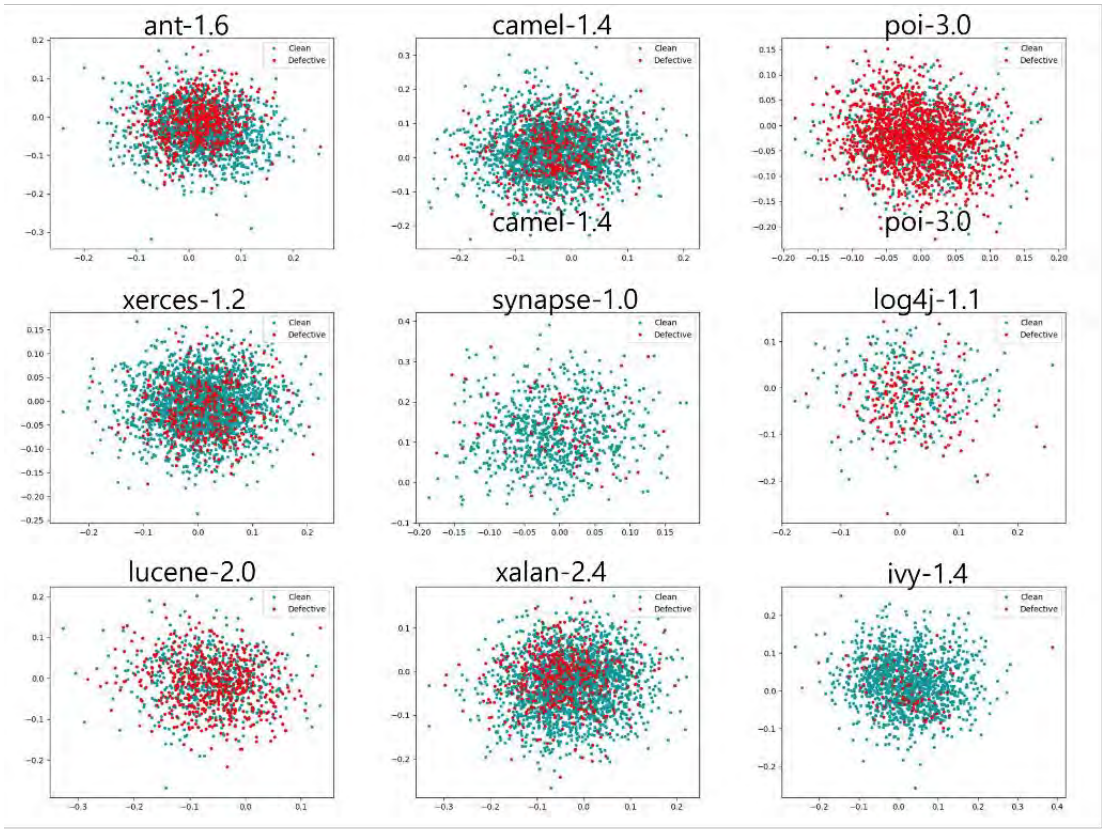
표3. 카이 제곱 검정 결과

Project	X-squared	df	p-value
ant-3.0	16138.697	115	0.0
camel-1.4	747.308	115	2.5746974e-93
poi-3.0	12380.333	115	0.0
Ivy-1.4	6670.738	254	0.0
lucene-2.0	3888.144	210	0.0
log4j-1.1	33955.959	252	0.0
synapse-1.0	6834.350	245	0.0
xalan-2.4	12232.023	254	0.0
xerces-1.2	24550.456	213	0.0

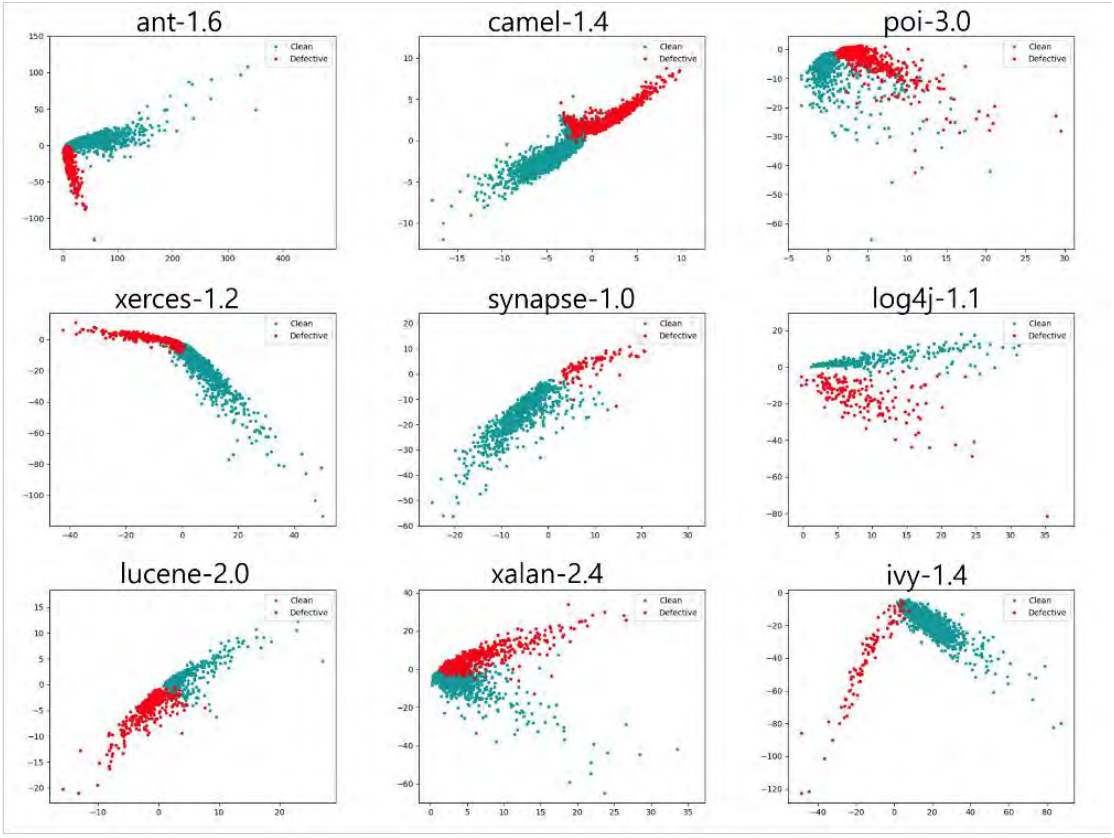
6. 실험 결과

RQ1. 결함 이미지와 비 결함 이미지의 분포에 차이가 있는가?

Chen et al.[5] 연구에서는 프로그램의 소스코드를 이미지 형태로 변환시켰고, end-to-end 방식의 프레임워크를 사용해 소프트웨어 결함 예측의 성능을 향상시켰다.



(a)



(b)

그림4. 딥 메트릭 러닝 적용 결과

표4. 프로젝트별 기대 빈도

Project	# of pixel (n < 5)	n < 5 (%)
ant-3.0	2	1.72
camel-1.4	3	2.60
poi-3.0	2	1.72
Ivy-1.4	18	7.05
lucene-2.0	17	8.05
log4j-1.1	27	10.67
synapse-1.0	23	9.34
xalan-2.4	29	11.37
xerces-1.2	19	8.87

표5. 이미지 유사도 비교 기법 적용 결과

Project	SSIM	PSNR	RMSE
ant-3.0	0.1744	16.3549	38.7969
camel-1.4	0.1577	18.1010	31.7315
poi-3.0	0.1840	16.8166	36.7884
Ivy-1.4	0.0775	13.5768	53.4197
lucene-2.0	0.1092	14.6260	47.3413
log4j-1.1	0.1033	13.4271	54.3480
synapse-1.0	0.1077	14.5945	47.5134
xalan-2.4	0.0668	13.2436	55.5088
xerces-1.2	0.1333	14.3821	48.6893

본 실험에서는 결함과 비 결함을 구분 짓는 소스코드의 의미 있는 정보들이 이미지에 담겨있어 소프트웨어 결함 예측 성능 향상에 기인함을 가정한다. 이 연구 질문에서는 통계적 검정 방법인 카이 제곱 검정 방법을 사용해 결함 이미지와 비 결함 이미지의 분포 차이가 있음을 보인다. 또한 결함과 비 결함 이미지의 분포 차이를 시각적으로 확인한다.

표3는 프로젝트별 결함 이미지와 비 결함 이미지 간 픽셀 빈도 분포의 카이 제곱 검정 결과를 보여준다. ant 프로젝트는 카이제곱=16138, 자유도=115, 유의확률 0.000이다. 따라서 픽셀 분포의 차이가 없다는 귀무가설을 기각하고 대립가설을 채택한다. 결함 이미지와 비 결함 이미지의 분포 차이가 있음을 통계적으로 검증하였다. 다른 8개의 프로젝트도 마찬가지로 유의확률이 0.000이므로 귀무가설을 기각한다. 표3의 유의확률 값은 1E-16보다 작아 0.0으로 표현한다.

카이 제곱 검정을 수행할 때 기대 빈도가 5 이하인 변수의 범주가 전체의 20% 미만이여야 한다. 표4는 프로젝트별 기대 빈도가 5 이하인 픽셀의 수를 나타낸다. 모든 프로젝트에서 기대 빈도가 5 이하인 변수의 범주가 전체의 20% 미만이므로 카이 제곱 검정을 수행한 검정 결과에 유의성이 있음을 보인다.

결함과 비 결함 이미지의 분포 차이를 확인하기 위해 대표적으로 ant-3.0 프로젝트를 선정한다. 그림3는 ant-3.0 프로젝트의 결함과 비 결함 이미지의 픽셀 분포를 나타낸다. ant-3.0 프로젝트가 가지고 있는 픽셀 값은 10부터 125이며, 그림3의 x축은 해당 프로젝트가 가지고 있는 픽셀 값의 범위를 나타낸다. 그림3(a)의 y축은 픽셀의 빈도수를 나타내고, 그림3(b)의 y축은 확률 밀도 값을 나타낸다. 그림3의 빨간색은 결함 이미지 분포를 나타내고, 파란색은 비 결함 이미지의 분포이다. 그림3(a)에서 결함 이미지는 32 픽셀 값을 6555개 가지고 있으며, 그림3(a)에서 비 결함 이미지는 110 픽셀 값을 2269개 가지고 있다. 그림3(a)에서 확인할 수 있듯이 ant 프로젝트의 결함과 비 결함 이미지는 10부터 125까지의 픽셀 값을 가지고 있다. 하지만 결함 이미지는 20부터 40까지의 픽셀 값을 많이 가지고 있었으며, 비 결함 이미지는 100부터 120까지의 픽셀 값을 보다 많이 가지고 있음을 확인할 수 있다. 그림3(b)에서의 곡선 그래프에서도 결함과 비 결함 이미지의 픽셀 분포에 차이가 있음을 보인다. 따라서 그림3의 분포 비교를 통해 결함과 비 결함 이미지의 분포에 차이가 있음을 시각적으로 확인할 수 있다.

RQ2. 컴퓨터 비전 기반의 이미지 유사도 비교 기법을 활용해 결함 이미지와 비 결함 이미지의 분포 차이를 확인할 수 있는가?

RQ2에서는 컴퓨터 비전 기반의 이미지 유사도 비교 기법 중 기본적이고 보편적으로 사용되는 기법을 채택해 결함 이미지와 비 결함 이미지의 분포 차이를 확인한다. 이미지 분포 차이를 확인하기 위해 우리는 SSIM와 PSNR, RMSE 기법을 모든 프로젝트에 적용하여 실험했다.

표5는 이미지 유사도 비교 기법을 적용한 결과이다. SSIM 결과값은 0에서 1 사이의 값을 가지며, 1에 가까울수록 결함과 비 결함 이미지가 유사함을 나타낸다. 표5의 결과에서 확인할 수 있듯이 모든 프로젝트의 SSIM 결과값이 0에 가까운 수치를 보인다. 따라서 결함과 비 결함 이미지가 유사하지 않음을 보였다. PSNR과 RMSE 결과값은 결함과 비 결함 이미지의 픽셀 차이에 대한 측정값을 나타낸다. 표5의 PSNR과 RMSE의 결과값을 통해 결함과 비 결함 이미지 간에 차이가 있음을 확인할 수 있다.

RQ3. 딥 메트릭 러닝을 활용해 결함 이미지와 비 결함 이미지가 효과적으로 구분 지을 수 있는가?

RQ3에서는 결함 이미지와 비 결함 이미지를 보다 잘 구분하기 위해 딥 메트릭 러닝을 적용한다. 딥 메트릭 러닝을 적용하면 딥 메트릭 러닝을 적용하기 전보다 결함과 비 결함 이미지가 보다 잘 구분될 수 있도록 도와준다. 더 나아가 결함과 비 결함의 두 분포의 차이를 시각적으로 분석한다. 딥 메트릭 러닝 방법의 도입은 분류 및 클러스터링의 정확도를 향상시킬 수 있을 것으로 기대된다.

실험에 사용된 결함 예측 모델은 DTL-DP(Deep Transfer Learning for Defect Prediction)[5]를 사용한다. 소스코드 파일을 이미지로 변환한 후 AlexNet을 기반으로 한 DTL-DP 프레임워크에 제공한다. 이후 Attention 네트워크에 입력되어 분류에 도움이 되는 피처 정보들을 강조하고 Maximum-Mean Discrepancy 손실 함수로 분포 차이를 계산한다. 우리는 결함 이미지와 비 결함 이미지의 거리에 초점을 맞추고 소스코드를 이미지화 한 결함과 비 결함 이미지 데이터에 딥 메트릭 러닝을 적용한다.

그림4은 9개의 프로젝트에 대해 딥 메트릭 러닝 기법을 이미지 데이터에 적용한 결과를 보여준다. 파란색 점은 결함이 없는 이미지, 빨간색 점은 결함이 있는 이미지다. 그

림4(a)는 소스코드를 이미지로 변환하여 2차원 벡터공간으로 정의한 결과를 보여준다. 그림4(b)는 딥 메트릭 러닝을 적용하여 실험한 결과를 나타낸다.

실험 결과 9개의 프로젝트에 대해 동일한 클래스의 데이터는 거리가 가깝지만 다른 클래스의 데이터는 거리가 떨어져 있어, 딥 메트릭 러닝의 적용으로 결함과 비 결함의 데이터가 명확하게 구분될 수 있음을 보여준다. 더 나아가 딥 메트릭 러닝 방법은 분류와 클러스터링 문제를 푸는데 사용될 수 있을 것으로 기대된다.

7. 위협 요소

내부 유효성에 대한 위협은 딥 메트릭 러닝을 위해 사용한 손실 함수로 Large margin cosine loss 단 하나의 함수만 사용해 실험했다. Large margin cosine loss 함수를 사용해 결함 이미지와 비 결함 이미지를 잘 구분 지을 수 있었으나 실험 결과가 본 연구에서 사용된 함수인 large margin cosine loss 함수 에만 유효할 수 있다.

외부 유효성에 대한 위협은 9개의 프로젝트에서 각각 한 개의 버전만을 사용한 것이다. 향후 연구로 추가 데이터세트를 사용해 실험을 진행할 예정이다.

8. 결론 및 향후 연구

소프트웨어 코드 영역에서 결함 여부를 예측하기 위해 소스코드의 피처를 추출한 후 이를 평가하는 많은 연구가 진행되어 왔다[4]. 대부분의 소프트웨어 소스코드 기반 연구는 추상 구문 트리(AST)를 기반으로 하지만, 일부 프로젝트에서 AST가 숨겨진 피처와 같은 의미 있는 코드 정보가 표현되지 않는 경우가 종종 발생한다[5].

최근 딥 러닝 네트워크는 이미지 분류 및 피처 추출에 매우 효과적임을 보인다[7,8]. Chen et al.[5] 연구는 AST와 같은 중간 표현을 피하고 소스코드를 이미지화하여 소스코드의 구조적 정보뿐만 아니라 의미 있는 정보를 직접 얻는 방법을 제안했다. 소스코드를 이미지로 변환했을 때 이점은 소스코드의 의미 있는 정보가 보존되며, 동일한 의미를 가지고 있는 데이터를 생성할 수 있다는 점이다.

본 연구에서는 소스코드를 이미지로 표현했을 때 결함 이미지와 비 결함 이미지 분포 차이를 분석했다. 딥 메트릭 러닝을 사용해 소스코드를 이미지로 만든 소프트웨어 결함 데이터에도 메트릭 러닝의 특징이 잘 들어나 결함 이미지와 비 결함 이미지를 더 잘 구분해 낼 수 있음을 확인했다. 또한 컴퓨터 비전 기반의 이미지 유사도 비교 기법과 통계적인 방법을 활용해 결함과 비 결함 이미지의 분포 차이가 유의미함을 확인했다. 실험 결과는 분류와 클러스터링의 정확도를 향상시킬 수 있을 것으로 기대된다.

향후 연구로 보다 많은 데이터세트를 사용해 실험을 수행하고 다양한 메트릭 러닝 방법을 적용할 계획이다.

참고문헌

[1] Gong, Lina, et al. "A novel class-imbalance learning approach for both within-project and cross-project defect prediction." IEEE Transactions on Reliability 69.1 (2019): 40-54.

[2] Wang, Song, et al. "Deep semantic feature learning for software defect prediction." *IEEE Transactions on Software Engineering* (2018).

[3] Shi, Ke, et al. "PathPair2Vec: An AST path pair-based code representation method for defect prediction." *Journal of Computer Languages* (2020): 100979.

[4] Shippey, Thomas, David Bowes, and Tracy Hall. "Automatically identifying code features for software defect prediction: Using AST N-grams." *Information and Software Technology* 106 (2019): 142-160.

[5] Chen, Jinyin, et al. "Software visualization and deep transfer learning for effective software defect prediction." *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 2020.

[6] Elish, Karim O., and Mahmoud O. Elish. "Predicting defect-prone software modules using support vector machines." *Journal of Systems and Software* 81.5 (2008): 649-660.

[7] Liu, Yi, Yu Fan, and Junghui Chen. "Flame images for oxygen content prediction of combustion systems using DBN." *Energy & Fuels* 31.8 (2017): 8776-8783.

[8] Liu, Yi, et al. "Ensemble deep kernel learning with application to quality prediction in industrial polymerization processes." *Chemometrics and Intelligent Laboratory Systems* 174 (2018): 15-21.

[9] Ball, Thomas, and Stephen G. Eick. "Software visualization in the large." *Computer* 29.4 (1996): 33-43.

[10] Halstead, Maurice Howard. *Elements of software science*. Vol. 7. New York: Elsevier, 1977.

[11] McCabe, Thomas J. "A complexity measure." *IEEE Transactions on software Engineering* 4 (1976): 308-320.

[12] Hall, Tracy, et al. "A systematic literature review on fault prediction performance in software engineering." *IEEE Transactions on Software Engineering* 38.6 (2011): 1276-1304.

[13] 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.

[14] Langelier, Guillaume, Houari Sahraoui, and Pierre Poulin. "Visualization-based analysis of quality for large-scale software systems." *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*. 2005.

[15] Wang, Hao, et al. "Cosface: Large margin cosine loss for deep face recognition." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[16] Wang, Zhou, Eero P. Simoncelli, and Alan C. Bovik. "Multiscale structural similarity for image quality assessment." *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*. Vol. 2. Ieee, 2003.
[17] Thompson, Geoffrey Z., and Ranjan Maitra. "CatSIM: A Categorical Image Similarity Metric." *arXiv preprint arXiv:2004.09073* (2020).
[18] Sara, Umme, Morium Akter, and Mohammad Shorif Uddin. "Image quality assessment through FSIM, SSIM, MSE and PSNR—a comparative study." *Journal of Computer and Communications* 7.3 (2019): 8-18.
[19] Liu, Zheng, and Robert Laganière. "Phase congruence measurement for image similarity assessment." *Pattern Recognition Letters* 28.1 (2007): 166-172.
[20] Kaya, Mahmut, and Hasan Şakir Bilge. "Deep metric learning: A survey." *Symmetry* 11.9 (2019): 1066.
[21] Elgammal, Ahmed, Ramani Duraiswami, and Larry S. Davis. "Probabilistic tracking in joint feature-spatial spaces." *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*. Vol. 1. IEEE, 2003.
[22] Chidamber, Shyam R., and Chris F. Kemerer. "A metrics suite for object oriented design." *IEEE Transactions on software engineering* 20.6 (1994): 476-493.
[23] Zhou, Xin, Jianmin Pang, and Guanghui Liang. "Image classification for malware detection using extremely randomized trees." 2017 11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID). IEEE, 2017.

스위치 모드 파워 컨버터에 적용된 다양한 지능형 제어기에 대한 설계 및 성능 비교

장진행 O 최호진

카이스트 소프트웨어 대학원 카이스트

Jinhaeng.jang@kaist.ac.kr hojinc@kaist.ac.kr

Comparative study on intelligent controllers for switch-mode power converter

Jin-Haeng Jang O Ho-Jin Choi

KAIST software graduate program KAIST

요 약

본 연구에서는 가전용 디스플레이 장치에 적용되는 넓은 입출력 범위에서 동작하는 직류-직류 전력 변환 회로(Dc-to-Dc Power Converter)에 다양한 지능형 제어기(Intelligent controller)를 적용하여, 시간 영역과 주파수 영역에서의 응답 특성에 대한 해석 및 제어기 설계 방법을 다루었으며, 각 제어기의 장단점 및 성능 비교를 하고자 하였습니다. 현대 제어 방식으로 분류되는 세가지 종류의 지능형 제어기인 퍼지 로직 제어기, 인공 신경망 제어기, 그리고 유전 알고리즘을 적용한 제어기를 각각 전력 변환 회로에 적용하여 정상 상태 및 동 특성 응답을 시뮬레이션을 통하여 평가하고, 대표적인 고전 제어 방식으로 분류되는 PID 제어기와의 성능을 비교하여, 지능형 제어기의 고유한 특성과 우수한 성능을 확인하였습니다.

1. 서 론

요즈음 가정용 디스플레이 장치는 우수한 화질을 구현하기 위하여 backlight 및 panel의 빠른 응답 특성을 요구하고 있다. 또한, 최근 음성 인식 등 인공지능 기능 내장을 위하여, 고속 연산의 CPU와 GPU 등 복잡한 연산을 수행하는 회로로 구성되어 있다. 따라서, 이러한 고속으로 동작하며 넓은 범위로 변화하는 입력 전압과 출력 부하 변동 조건에서 동작하는 회로에 전원을 공급하는 직류-직류 전력용 컨버터(dc-to-dc power converter)에 대한 우수한 동특성이 매우 중요해지고 있다. 이러한 가정용 디스플레이 장치에는 주로 높은 입력 전압을 낮은 출력 전압으로 변환하는 강압형 직류-직류 전압 변환 전력용 컨버터가 이용되고 있다. 전력 변환 컨버터는 전력 변환을 위한 블록과 제어 블록으로 나눌수 있으며, 전력 변환 블록은 반도체 스위치, 에너지 저장 소자인 인덕터와 커패시터로 이루어져 있다. 더불어, 안정적인 출력 전압을 공급하기 위한 제어기로는 주로 PID 방식이 많이 활용되어져 왔다. 이러한 PID 제어기는, 제어기 전달함수의 각각 고정된 비례, 적분, 미분 게인을 가지고 동작하므로서, 특정 동작점에서는 우수한 특성을 보이나, 넓은 범위로 변화하는 입력 및 출력 부하 변동 조건에서는 응답 성능의 한계를 가지고 있다.

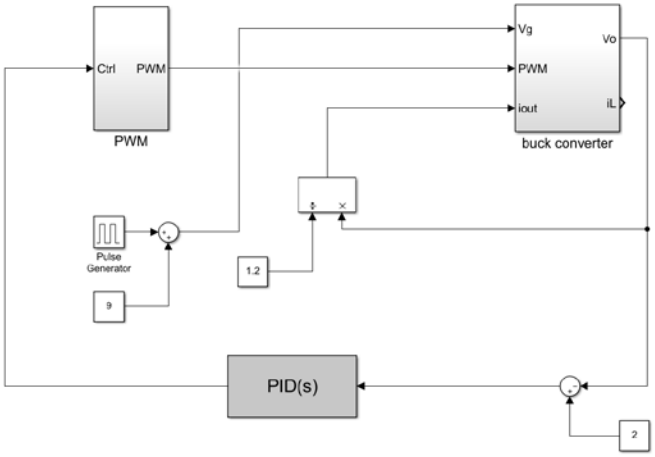


그림 1. 강압형 직류-직류 전력 변환 장치. 입력 전압: 9 [V], 출력 전압: 2 [V], 출력 전류: 1.6 [A]

또한, 고효율 전력 변환을 구현하기 위한 스위칭 동작 방식에 의한 비선형 특성에 의해 그 동특성에 대한 모델링이 매우 복잡하여, 제어기 설계 역시 까다롭다는 단점을 가지고 있다 [1]. 이러한 복잡한 제어기 설계와 동특성의 한계를 극복하기 위하여, 각 전력 변환 회로 방식에 맞는 여러 제어 방식이 제안되어져 왔다 [2].

	Input voltage	Output current
Condition A	9 V	1.6 → 0.3 A
Condition B	9 V	0.3 → 1.6 A
Condition C	9 →10 V	1.6 A
Condition D	10 → 9 V	1.6 A

표 1. 넓은 범위로 변화하는 입출력 동작 조건 사양

본 연구에서는, 현대 제어 방식으로 분류되는 지능형 제어기의 대표적인, 퍼지 로직 제어기 [3,4], 인공 신경망 제어기 [5-9], 그리고, 유전 알고리즘[10-14]을 적용한 제어기를 각각 전력 변환 회로에 적용하여, 정상 상태 및 동 특성 응답을 시뮬레이션을 통해 평가하여, 종래 고전 제어 방식인 PID 제어 방식 대비 우수한 동특성을 확인하고자 하였다. 본 연구에 사용된 높은 입력 전압 9V를 낮은 출력 전압 2V로 변환하는 강압형 직류-직류 전력 변환 장치의 회로 구성을 그림 1에 나타내고 있다 [16]. 전력 변환 블록과 부궤환의 출력과 입력 사이에 연결된 PID 제어기와 제어 신호를 펄스폭 신호로 변환하는 PWM 블록으로 구성되어 있다. 제어기는 출력 전압과 목표 전압간 오차 신호의 차이를 입력으로 하여, 제어기의 전달함수로 입력시키고, 제어량에 해당하는 제어 신호를 출력한다. 이때, 제어기의 설계에 따라, 컨버터의 정상 상태 응답과 과도 응답 동 특성이 크게 좌우된다. 펄스폭 변조 블록은 이러한 제어 신호를 받아 필요한 변조폭만큼 도통 구간을 변조하므로서, 출력 전압을 항상 목표 전압을 추종하도록 제어하고 있다. 이러한 컨버터가 장치내에서 동작하는 사양은 입출력 조건이 매우 폭 넓게 변화하게 된다. 이러한 변화하는 입력 전압과 출력 부하 전류의 조건을 테이블 1에 나타낸다. 각 입력과 출력 변동 조건을 구분하여 A, B, C, 그리고 D의 4개 조건으로 나눌 수 있다. 조건 A와 B는 입력 전압을 고정시키고, 출력 부하 전류가 변화하는 조건을 나타내고 있으며, 조건 C와 D는 출력 부하 전류는 고정시키고, 입력 전압이 변화하는 조건을 나타내고 있다. 본 연구에서는 이러한 폭넓게 변화하는 조건에서 동작하는 컨버터에 지능형 제어기를 적용하여 우수한 동특성을 확인하고자 하였다.

.

2. 제어 이론의 분류

제어 이론은 그림 2에 나타낸 바와 같이 크게 고전 제어와 현대 제어로 분류될 수 있다 [2]. 고전 제어 방식은 수학적으로 잘 증명된 모델링으로 인해 정확한 해석 결과를 얻을 수 있다는 가장 큰 장점이 있다. 하지만, 전력 변환기의 응용 회로와 동작 방식에 따라 다른 복잡한 수학적 모델링은 제어기 설계 및 응용의 큰 제약이 되었다. 따라서, 회로 해석 시뮬레이션 툴을 활용하여 근사화 및 최적화 기법을 활용하여 제어기를

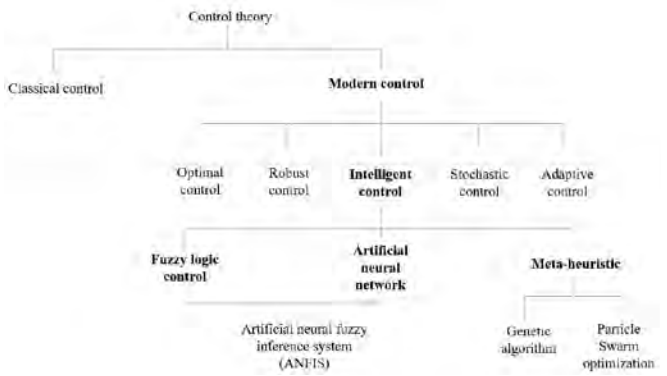


그림 2. 제어 이론의 분류

설계하였다 [6]. 반면, 고전 제어 방식과 대비되는 현대 제어 방식으로서 여러 제어기가 제안 및 활용되어져왔다. 이러한 현대 제어 방식은 기존의 고전 제어 방식인 PID 제어 방식의 단점을 보완하기 위한 설계 기법을 제안하였다. 이러한 다양한 제어 방식은 사용되는 컨버터의 종류에 따라, 최적화 및 성능의 차이를 보여주었다. 본 연구에서는, 기본적인 강압형 직류-직류 전력 변환기에 지능형 제어 방식을 적용하여 종래 고전 제어 방식인 PID 제어 방식과의 해석 및 설계 방법, 그리고, 성능의 차이를 확인하고자 하였다. 대표적인 지능형 제어 방식으로는 퍼지 로직 제어 방식이 있다. 이 방식은 언어적 규칙을 적용하는 직관적인 설계 방법을 제공하므로서, 비교적 설계하기 쉬운 제어기 설계 방법으로서 활용되어져 왔다. 두번째는 근래 자연어 처리 분야와 이미지 인식 등 다방면에 널리 활용되고 있는 인공신경망 제어기를 적용하고자 하였다. 이러한 인공신경망 제어 방식의 경우, 수십년 전 제안 되어졌으나, 연산 능력과 기억 저장 장치의 리소스의 제약 등으로 인해 그 성능을 발휘할 수 없었다 [5].

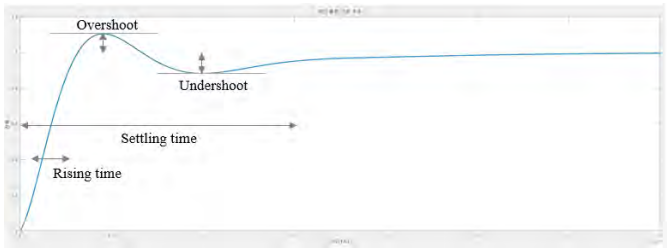


그림 3.1 시간 영역에서의 최적화

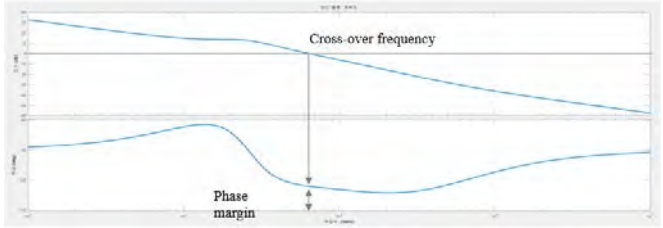


그림 3.2 주파수 영역에서의 최적화

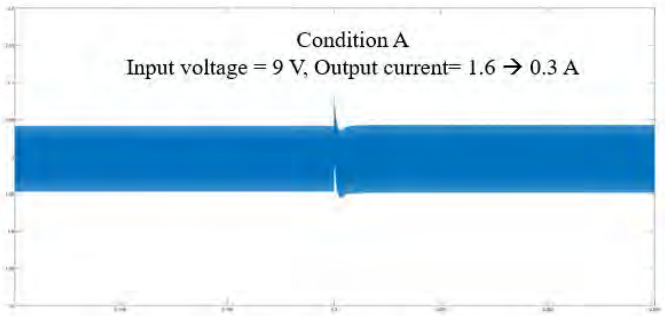


그림 4.1 동작 조건 A에서의 과도 응답 특성

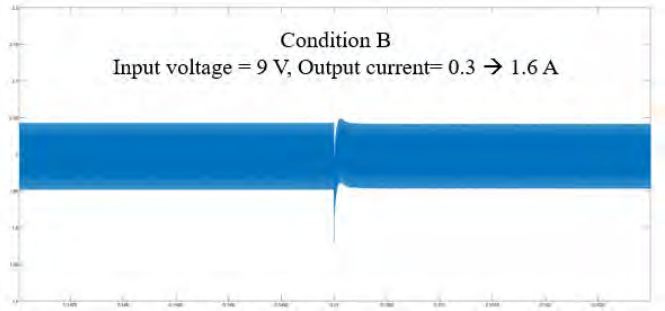


그림 4.2 동작 조건 B에서의 과도 응답 특성

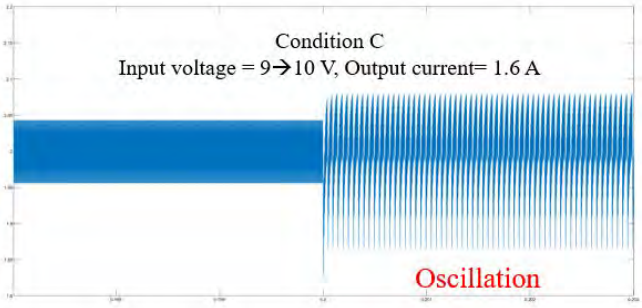


그림 4.3 동작 조건 C에서의 과도 응답 특성

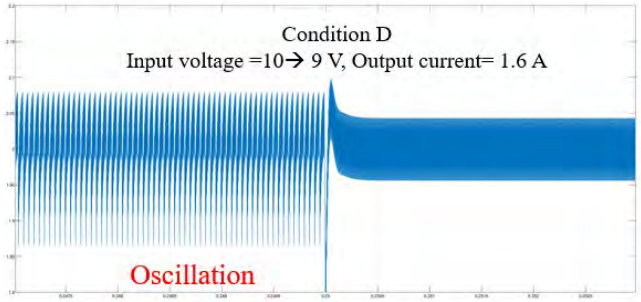


그림 4.4 동작 조건 D에서의 과도 응답 특성

본 연구에서는 동일한 전력 변환단 회로에 인공신경망 제어기를 구현하여, 동일 조건에서 과도 응답 특성을 확인하므로서, 지능형 제어기의 성능을 확인하고자 한다. 마지막으로 적용된 지능형 제어 방식은 메타 휴리스틱 방식의 유전 알고리즘을 적용한 제어기이다. 이러한 세가지 종류의 지능형 제어 방식을 전력 변환단 컨버터에 적용하여 성능을 시뮬레이션을 통해 비교 평가하였다 [16-19].

3. PID 제어기의 설계 및 성능

그림 1에 나타낸 직류-직류 전력용 컨버터의 PID 제어기의 전달 함수는 다음 식 1과 같이 나타낼 수 있다.

$$P + I \cdot \frac{1}{S} + D \cdot \frac{N}{1+N\frac{1}{S}} \qquad (1)$$

식 1은 비례(P), 적분(I), 그리고 미분(D) 항으로 이루어져 있으며, 비례 제어항과 적분 제어항은 정상 상태 출력 전압의 오차를 최소화하도록 설계되며, 미분항은 과도 응답 조건에서의 빠른 응답 성능을 나타내도록 각 항의 계수가 선정된다. PID 최적화 툴을 활용하여, 설계 목표에 최적화된 각 항의 게인을 구할 수 있다. 설계 목표는 그림 3.1에 나타낸 바와 같이 과도 응답에 대해서, 출력 전압의 최대치와 최소치가 목표 전압인 2V의 10% 이내와 출력 전압의 상승 시간을 0.5초 이내가 되도록 설계하였다. 또한, 컨버터의 빠른 응답 특성과 안정도를 확보하기 위하여, 넓은 대역폭을 확보하고 위상 여유를 50도 이상으로 안정도를 확보하고자 하였다.

이러한 설계 목표 사양에 대한 최적화를 통하여 빠른 응답 특성과 안정도의 trade-off를 통하여, 식 1의 비례 게인 P=0.987, 적분 게인 I=12708, 미분 게인 0.0000004의 각 계수를 얻을 수 있었다. 넓은 범위로 변화하는 입력과 출력 부하 변동 조건 A부터 D까지의 각각 시뮬레이션 결과를 그림 4에 나타내었다. 그림 4.1과 2에서 알 수 있는 바와 같이, 부하가 변동하는 조건에서는 출력 전압의 최대치와 최소치가 목표 전압 2 V로 수렴하고 있는 것을 확인할 수 있다. 그러나, 그림 4.3과 4.4에 나타낸 바와 같이, 입력 전압이 변화하는 순간에는 출력 전압이 정상적으로 출력되는 것이 아니라 발진하는 파형을 보여주고 있다. 위 실험 결과에서 알 수 있는 바와 같이, 설계 목표를 만족하기 위하여 최적화 툴을 통하여 얻어진 게인들로 설계된 PID 제어기를 컨버터에 적용하여 넓은 범위로 변화하는 입력 전압과 출력 전류 조건에서 동작시키는 경우, 특정 동작 조건에서 발진하는 원치 않는 응답을 가질 수 있다. 이것은 특정 동작점에 대하여 최적화된 제어기의 고정된 게인값에 기인한다. 바로 이러한 점이 종래 PID 제어 방식의 과도 응답 성능의 한계를 잘 나타내고 있다.

4. 지능형 제어기

본 장에서는 지능형 제어기를 설계 및 적용하여 앞장에서 살펴본 PID 제어 방식의 단점을 보완하고 우수한 성능을 보일 수 있는지에 관하여 다루어 보고자 한다.

4.1 퍼지 로직 제어기

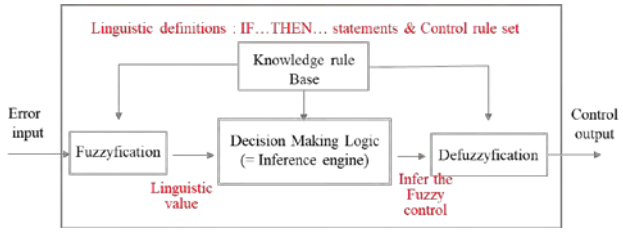


그림 5. 퍼지 로직 제어

그림 5는 퍼지 로직 제어의 제어 방법을 나타내고 있다 . 퍼지 로직 제어기는 언어적 정의를 기반으로 한 일련의 규칙 기반의 세트로 된 룩-업 테이블과 추론 엔진으로 구성되어 있다. 퍼지 제어기는 입력의 오차 전압 신호를 언어적 값으로 퍼지화 변환하고, 여러 제어 규칙의 세트로 된 룩-업 테이블로부터 추론 값을 결정하여 얻고, 제어 신호의 출력을 원래의 입력 신호인 전압 신호로 변환하여 출력한다. 퍼지 제어의 장점은 복잡한 수학적 모델이 아닌, 언어적 규칙을 기반으로 하기 때문에, 매우 직관적으로 추론을 할 수 있다 [17]. 예를 들면, 만일 출력 전압 오차 신호가 음수이고, 출력 오차 신호의 변화율 또한 음수라면, 에너지 공급이 최소화 되도록 펄스폭 변조량을 대폭 줄여라와 같은 직관적인 방법을 이용하여 제어기를 설계 할 수 있다. 그림 6에 퍼지 로직 제어기를 적용한 직류-직류 전력 변환기의 전체 모델을 보여주고 있다. 그림 1에 나타낸 전력용 컨버터에 적용된 PID 제어기로부터 퍼지 로직 제어기로 변경되었다. 이때 제어기의 입력으로는 오차 신호와 오차 신호의 변화율에 해당하는 두 개의 입력 신호가 주어지고 있다.

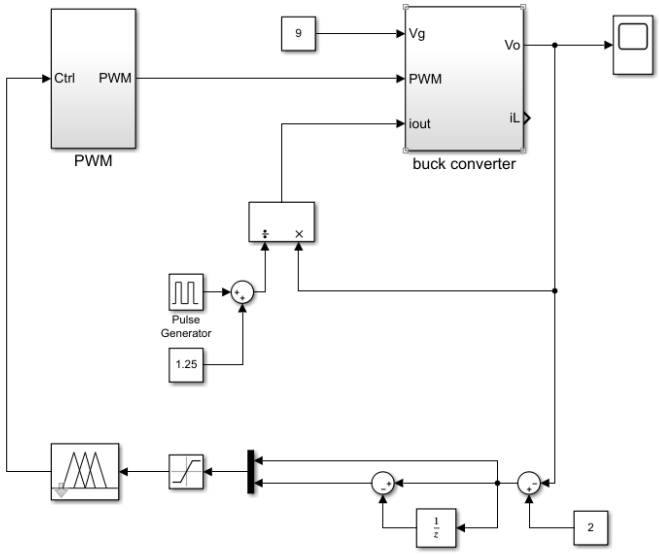


그림 6. 퍼지 로직 제어 방식 전력용 컨버터

e(n-1) \ e(n)	NB	NS	ZE	PS	PB
PB	ZE	PS	PB	PB	PB
PS	NS	ZE	PS	PB	PB
ZE	NB	NS	ZE	PS	PB
NS	NB	NB	NS	ZE	PS
NB	NB	NB	NB	NS	ZE

N: Negative, P: Positive, B: Big, S: Small, Z: Zero, E: Error

표 2. 5행 5열의 규칙 기반 룩-업 테이블

표 2는 제어 특징의 규칙 셋을 오차 신호와 오차 신호의 변화율로 하여 나타낸 제어의 규칙에 대한 테이블이다. 표에 나타낸 바와 같이, 오차 신호의 크기와 오차 신호 변화율의 크기에 따라 25가지의 맴버쉽 함수라고 부르는 제어량으로 나눌 수 있다. 그림 7.1과 7.2는 은 오차 신호와 오차 변화량에 대한 각 멤버쉽 함수를 보여주며, 그림 7.3은 해당 조건에서의 결과인 출력 신호를 보여주고 있다. 그림의 조건은, 오차 신호가 큰 음의 량이고, 오차 변화량 또한 큰 음의 량이라면, 출력 신호를 크게 줄여서 제어해라와 같은 언어적 규칙을 기반으로 한 맴버쉽 함수를 나타내는 그림으로, 쉽게 제어의 규칙을 확인할 수 있다.

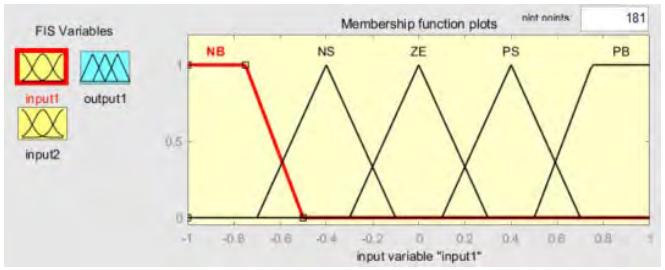


그림 7.1 오차 신호에 대한 맴버쉽 함수

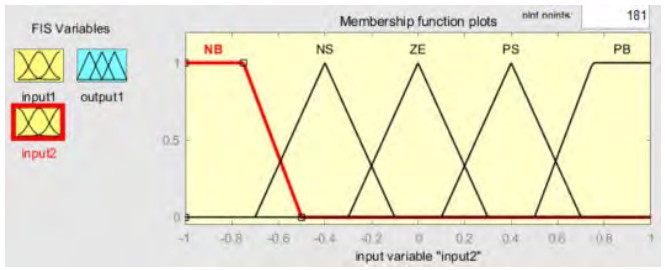


그림 7.2 오차 변화량에 대한 맴버쉽 함수

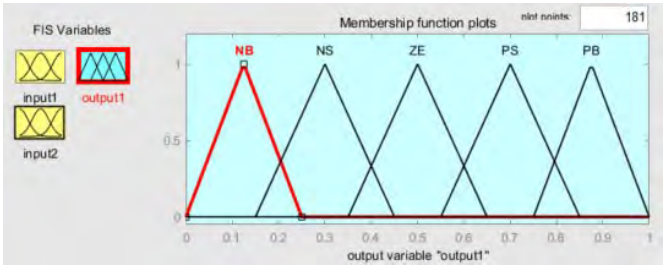


그림 7.3 출력 맴버쉽 함수

그림 8. 실시간 퍼지 로직 제어량 모니터링

그림 8은 퍼지 로직 제어기의 두가지 오차 입력 신호에 대한 출력 제어량의 변화의 움직임을 실시간으로 관측할 수 있는 퍼지 로직 제어 방식의 모니터링 그림이다. 입력과 출력의 색이 칠해진 번호의 삼각형 규칙들이 현재 변화하는 오차 신호에 해당하는 순시 조건에서의 해당 출력 신호이다. 그리고, 아래 그림 9는 3차원 공간에 나타낸 입력 오차 신호와 오차 변화량에 대한 출력 제어량을 나타낸다. 이 그림으로부터 퍼지 제어의 변화량을 시각적으로 파악할 수 있지만, 다른 한편으로는 퍼지 제어가 가지는 제약 사항을 알 수 있다. 즉, 제어 범위의 양 끝 부근에서 출력이 로컬 미니마 트랩에 갇혀 있는 것을 볼 수 있다. 이는 과도 부하 응답 시 출력 전압 제어에 대한 제한을 나타내는 것을 알 수 있다.

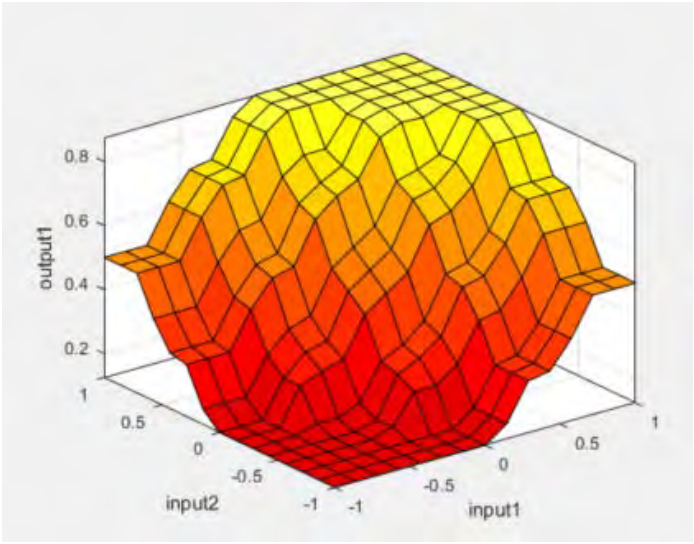


그림 9. 3차원 공간 퍼지 로직 제어 표현

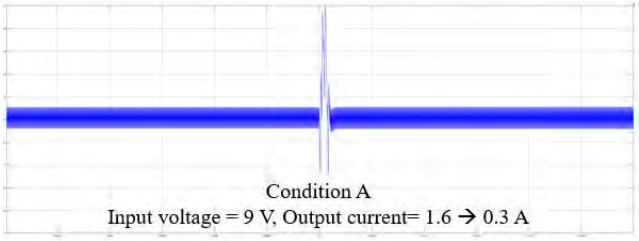


그림 10.1 동작 조건 A에서의 과도 응답 특성

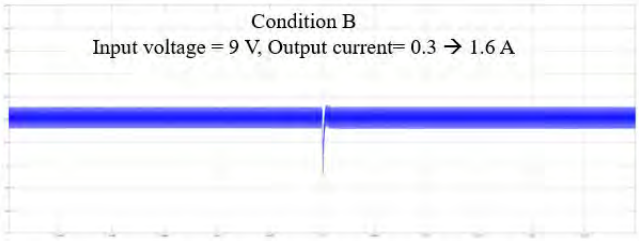


그림 10.2 동작 조건 B에서의 과도 응답 특성

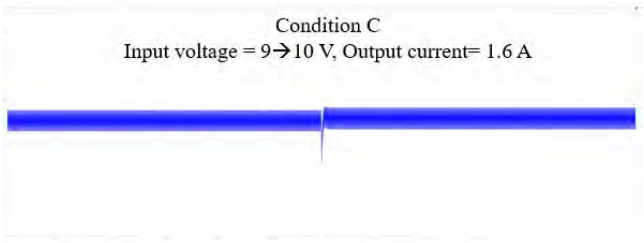


그림 10.3 동작 조건 C에서의 과도 응답 특성

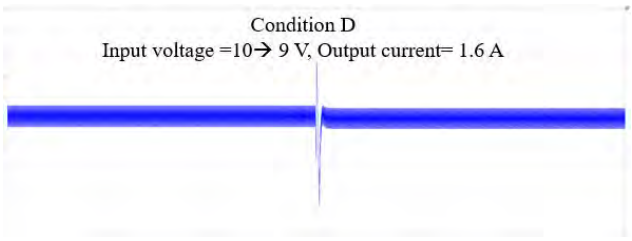


그림 10.4 동작 조건 D에서의 과도 응답 특성

그림 10에 퍼지 로직 제어기를 적용한 직류-직류 전력 변환기 출력의 과도 응답 특성의 시뮬레이션 결과를 보여주고 있다. 그림 4에서 보여준 기존 PID 제어기를 적용한 전력 변환기의 입력 전압이 변화하는 조건에서 발생한 출력의 발진 현상이, 동일한 조건에서 그러한 발진하는 현상이 없는 것으로 보아, 퍼지 로직 제어기가 넓은 범위의 입력과 출력 부하 변동 조건에서 우수한 제어기임을 나타내고 있다. 이러한 점은 종래 PID 제어기가 고정된 상수의 PID 게인으로 인해 넓은 범위에서 변화하는 입력과 출력 조건에서 동작하는데 제약 사항이었다면, 퍼지 로직 제어기를 적용한 경우, 변화하는 입력과 출력 조건에 맞는 규칙 기반의 퍼지 로직 제어 방식으로 동작하여 우수한 성능을 나타낼 수 있다는 것을 보여주고 있다. 그러나, 그림 10.1의 조건에서 PID 제어기보다 큰 크기의 출력 신호의 최대치를 보여주고 있는 것을 알 수 있으며, 이러한 것은 그림 9에서 나타낸 바와 같이 제어 범위의 끝단에서 보여준 제어 신호의 클램핑과 관련이 있다.

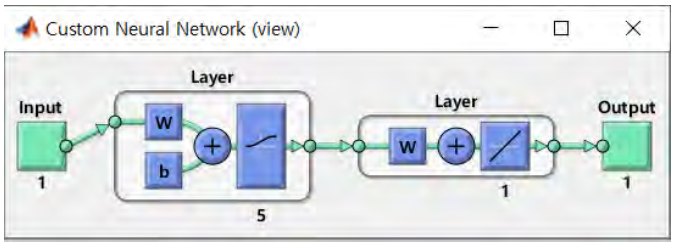


그림 11. 인공 신경망 제어기

4.2 인공 신경망 제어기

인공지능과 머신 러닝 기술에 대한 관심이 높아지고 응용 기술이 발전함에 따라 제어 분야에도 적용되기 위한 여러 시도들이 있다 [4,5]. 이번 절에서는 지능형 제어기로서 인공 신경망 제어기를 설계하여 성능을 확인하고자 한다. 그림 11은 매트랩 소프트웨어로 구현한 피드 포워드 뉴럴 네트워크의 구조를 보여주는 그림이다 [18]. 입력층과 다섯 개의 뉴런으로 구성된 히든층과 출력층으로 구성되어 있다. Back propagation 알고리즘으로 학습하였으며 [8], 손실 함수로서 평균 제곱 오차 방식을 이용하였다. 입력 데이터로는 종래 컨버터의 오차량에 대한 컨버터의 펄스폭 변조의 구간 변화량에 대한 일정 패턴을 적용하였다.

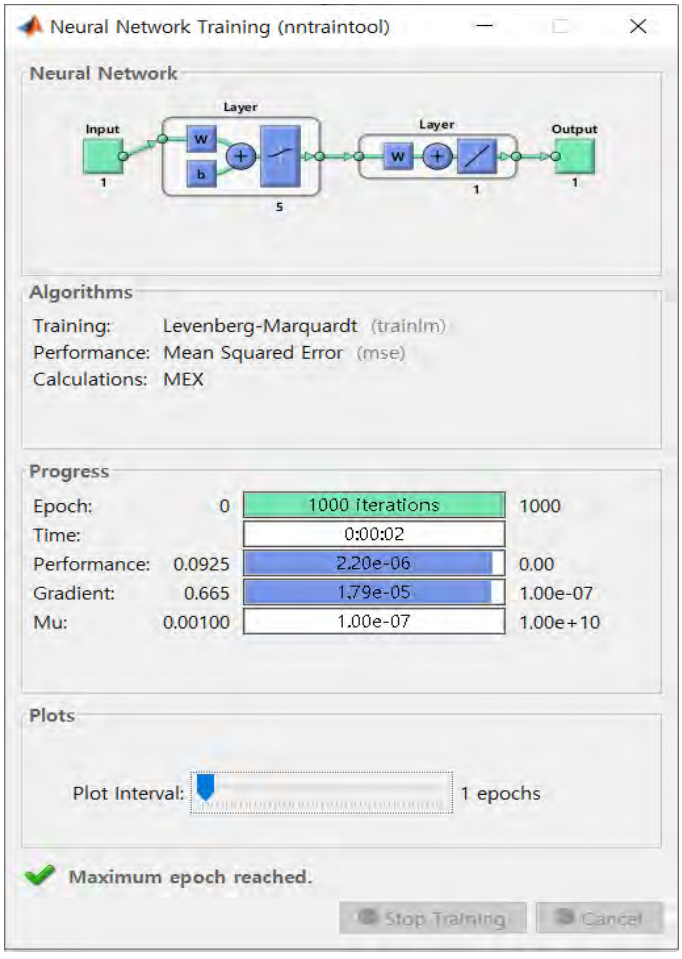


그림 12. 인공신경망 학습 결과

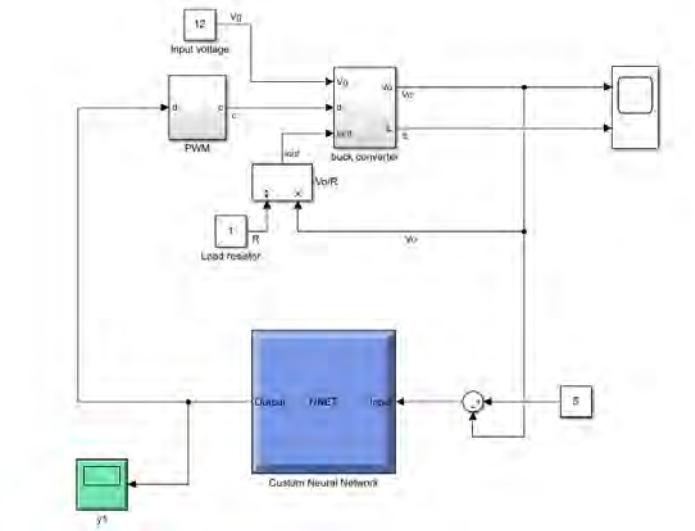


그림 13. 인공신경망 제어기를 적용한 컨버터 모델

그림 12 는 인공 신경망의 학습 과정과 결과를 나타내고 있다. 간단한 신경망에 대한 학습이므로 짧은 반복 학습으로부터 최소의 손실 값에 도달한 것을 알 수 있다. 그림 13 은 인공신경망 제어기를 적용한 직류 변환 컨버터 모델을 나타내고 있다. 출력 전압과 목표 전압과의 오차 전압을 신경망의 입력 신호로 입력하고 신경망의 출력을 제어 신호의 출력으로 발생시키고 있다. 아래 그림 14 는 인공 신경망 제어기를 적용한 컨버터의 응답 특성의 시뮬레이션 결과이다.

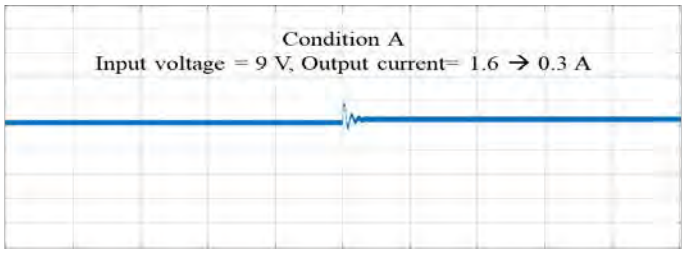


그림 14.1 동작 조건 A에서의 과도 응답 특성

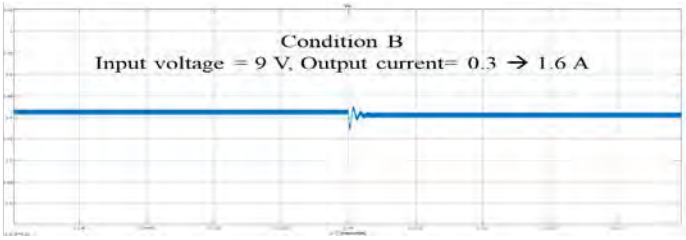


그림 14.2 동작 조건 B에서의 과도 응답 특성

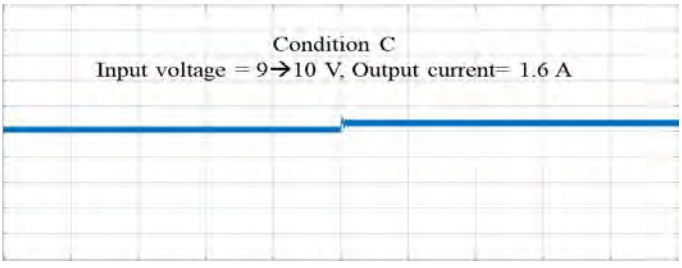


그림 14.3 동작 조건 C에서의 과도 응답 특성

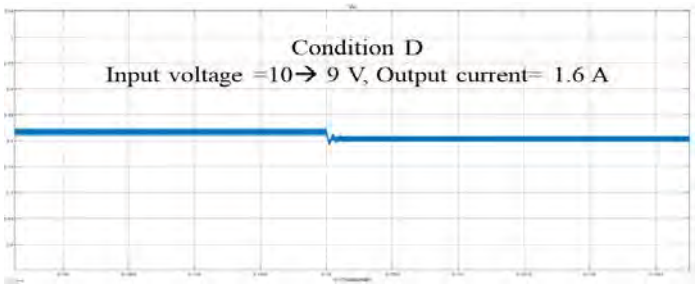


그림 14.4 동작 조건 D에서의 과도 응답 특성

그림 14 의 결과로부터 입력 전압과 출력 부하 전류가 변화하는 모든 동작 조건에서 출력 신호의 발진 현상 없이 안정적으로 동작하며, 최대값 또는 최소값이 매우 적은 양으로 빠르게 수렴 것을 확인 할 수 있다. 이러한 시뮬레이션의 결과로부터 넓은 범위로 변화하는 입력 전압과 출력 전류 조건에서 인공신경망의 우수한 제어 특성을 확인할 수 있다.

4.3 유전 알고리즘

마지막으로, 탐색 기반의 오픈 소스 유전 알고리즘[15]을 전력 변환단의 제어기로 적용하여 전력용 컨버터의 동특성을 확인하였다. 그림 1 에서 보여준 전력용 컨버터의 전력 변환단에 대한 소 신호 전달함수는 식 (2)와 같이 표현될 수 있다.

$$G_{vd}(s) = \frac{\hat{v}(s)}{\hat{d}(s)} = \frac{26\times10^3 s + 2817\times10^6}{s^2 + 1.2\times10^3 s + 313\times10^6} \quad (2)$$

유전 알고리즘은 총 20 세대를 반복하면서 성능이 점차 진화하도록 되어 있다 [15]. 또한, 하나의 세대는 40 개의 독립적인 개체들로 구성된다. 비트 연산을 위해 각 개체에는 비례, 적분, 그리고 미분 제어항에 각각 8 비트씩 할당된다. 비트 연산을 통하여 우수한 유전자를 선택하고, 자손 개체 간 유전자에 대한 크로스 오버 연산, 그리고, 돌연 변이에 대한 연산을 수행하며 세대를 거듭할수록 개체들이 진화하게 된다. 그림 15 는 이러한 유전 알고리즘을 적용하여 얻어진 게인들을 적용한 컨버터의 전체 모델을 나타낸다.

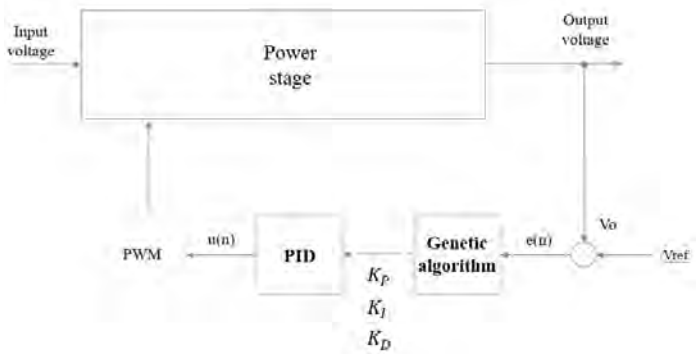


그림 15. 유전 알고리즘을 적용한 컨버터

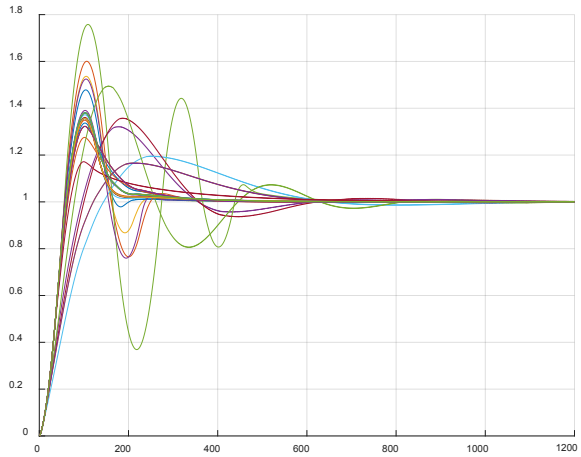


그림 16. 첫번째 세대의 계단 응답 특성

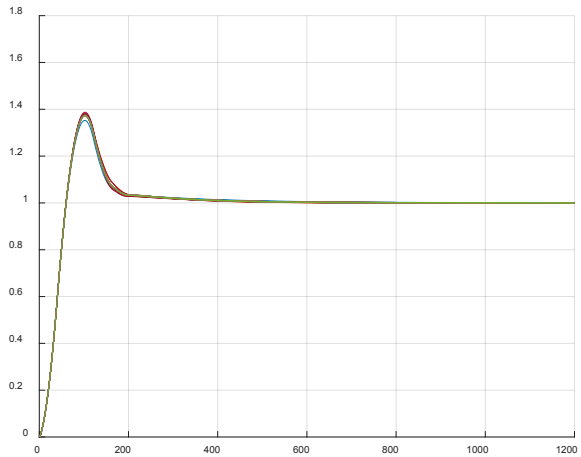


그림 17. 마지막 최종 세대의 계단 응답 특성

이러한 각 게인들은 사전 학습을 통해 얻어지며, 전이 학습 방법으로 실시간으로 동작하는 컨버터에 적용하여 구현할 수 있다. 본 연구에서는 시뮬레이션을 이용하여 유전 알고리즘으로부터 얻어진 결과를 전력용 컨버터에 적용하여 그 동특성을 확인하였다 [19]. 그림 16 에 나타난 바와 같이 최초의 세대에서는 40 개의 개체들이 각각 다른 응답을 보여주고 있으나, 그림 17 에서는 모든 40 개의 개체들이 개선된 특성의 동일한 응답을 나타내며 수렴하는 응답 특성을 보여주고 있다.

5. 결론

본 연구에서는, 기본적인 강압형 직류-직류 전력 변환기에 지능형 제어 방식을 적용하여 종래 고전 제어 방식인 PID 제어 방식과의 해석 및 설계 방법, 그리고, 성능의 차이를 확인하고자 하였다. 실험으로부터 PID 제어기의 경우, 특정 동작점에 한해서 좁은 범위에서 최적화된 성능을 보여주며, 넓은 범위로 변화하는 동작 조건에서는 발진 등 불안정한 동작을 보여주었습니다.

반면, 본 연구에서 적용한 세 가지 지능형 제어기 중 퍼지 제어기의 경우, 설계가 매우 직관적이며, 전체 동작 영역에서 설계 사양 내의 적합한 수준의 성능을 보여주었다. 그러나, 양 극단의 동작 범위에서 제어 신호가 클램핑되는 특성을 나타내었다. 두번째 지능형 제어기인 인공 신경망의 경우, 넓은 범위의 동작 조건에 대해서, Fuzzy 제어 방식보다 우수한 성능을 나타내었다. 마지막으로 적용된 유전 알고리즘 제어기의 경우, 세대 간 연산을 통한 진화과정을 통해 빠른 응답 특성과 안정도 측면에서도 우수한 성능을 보여주었다. 따라서, 본 연구에서 구현한 세가지 지능형 제어기의 우수한 응답 특성을 확인할 수 있었다. 본 프로젝트의 결과를 향후 현업에서 개발중인 DSP를 이용한 디지털 제어 방식에 적용한다면, 개선된 성능 향상과 함께 우수한 품질의 제품 개발로 이어질 수 있을 것으로 기대해 볼 수 있다.

참고 문헌

[1] V. F. Pires and J. F. A. Silva., "Teaching Nonlinear Modeling, Simulation, and Control of Electronic Power Converters using MATLAB/simulink," IEEE Trans. on Educ., vol.45, no.3, p253-261, Aug. 2002.

[2] M. J. Blondin et al., "Computational intelligence and optimization methods for control engineering," Springer Nature Switzerland, 2019.

[3] P. Mattavelli, L. Rossetto, G. Spiazzi, and P. Tenti, "General-Purpose Fuzzy Controller for DC-DC Converters," IEEE Trans. Power Electron., vol.12, no.1, pp79~86, Jan. 1997.

[4] B. k. Bose, "Fuzzy logic and Neural Network," IEEE industry applications magazine, pp57-63, May 2000.

[5] T. Dragicevic, P. Wheeler, and F. Blaabjerg, "Aritificial Intelligence Aided Automated Design for Reliability of Power Electronic Systems," IEEE Trans. Power Electron., vol.34, no.8, pp7161~7171, Aug. 2019.

[6] N. Kodner, D. Adar, S. B. Yaakov, "Neural Network control of switch mode systems: off line training by an ideal controller data set," ICPE, pp 56-61, 1995.

[7] B. k. Bose, "Neural Network Applications in Power Electronics and Motor Drives," IEEE Trans. on Ind. Elect., vol.54, no.1, Feb. 2007.

[8] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning representations by back-propagating errors," Nature publishing group., vol.323, Oct. 1986.

[9] X. H. Yu, W. L. Taufik, "Design and implementation of a neural network controller for real-time adaptive voltage regulation," Neuro-computing, pp 531-535, 2009.

[10] A. Arslan, M. Kaya, "Determination of fuzzy logic membership functions using genetic algorithm," Fuzzy sets and systems., vol. 118, pp 297-306, 2001.

[11] S. Aldaco, H. Calleja, "Metaheuristic optimization methods applied to power converters: a review," IEEE Transactions on power electronics., vol. 30, No.12, Dec. 2015.

[12] A. Endre, E. R. Hinterding, and Z. Michalewicz, "Parameter control ′ in evolutionary algorithms," IEEE Trans., Evol. Comput., vol. 3, no. 2, pp 124–141, Jul. 1999.

[13] H. Visairo, M. A. Medina, and J. M. Ramirez, "Use of evolutionary algorithms for design optimization of power converters," in Proc. 22nd Int. Conf. Electr. Commun. Comput., pp 268–272., 2012.

[14] S. L. Brunton and J. N. Kutz, "DATA-DRIVEN SCIENCE AND ENGINEERING: Machine Learning, Dynamical Systems, and Control," CAMBRIDGE, 2019.

[15] I. Uzunlar (2017) PID_genetic_algorithm [Source code]. https://github.com/ismailuzunlar?tab=repositories

[16] Math Works Inc., Simulink control design User's Guide R2020a, 2020.

[17] Math Works Inc., Fuzzy logic toolbox User's Guide R2020a, 2020.

[18] Math Works Inc., Deep learning toolbox User's Guide R2020a, 2020.

[19] Math Works Inc., Global optimization toolbox User's Guide R2020a, 2020.

감사의 글

본 연구는 한국전력공사의 2018년 착수 에너지 거점대학 클러스터 사업에 의해 지원되었음
(과제번호: R18XA05)

유스케이스를 이용한 키워드 기반 테스팅 방안

최승훈 ○ 최은만

동국대학교 대학원 컴퓨터공학과

sehn47@dongguk.edu, emchoi@dongguk.edu

Keyword-Driven Testing Method Using Use Cases

Seung Hun Choi ○ Eun Man Choi

Dept. of Computer Science and Engineering, Dongguk University

요 약

키워드 기반 테스트는 값을 기반으로 하는 테스팅 방법과는 달리 키워드를 이용하여 개발자가 아니더라도 쉽게 테스트 케이스를 작성하고 재사용성이 높다는 면에서 최근 관심을 끌고 있다. 하지만 키워드를 추출하기에 적합한 힌트가 되는 것이 불분명하고, 실행된 소프트웨어를 사용하여 키워드 테스트 케이스를 작성하는 것은 불규칙적이다. 이러한 불편함을 최소화 하기 위해 본 연구에서는 유스케이스에서 키워드를 추출하여 키워드 기반 테스트 케이스를 작성하기 쉽게 만드는 방안을 제안한다. 키워드 기반 테스트 케이스는 오픈소스 기반인 로봇프레임워크를 기준으로 작성되었으며, 유스케이스 다이어그램과 명세를 확인 후 본 논문에서 주장하는 방법과 규칙에 따라 절차대로 작성하여 키워드 기반 테스팅의 단점을 줄이고 장점을 극대화할 수 있게 하였다.

1. 서론

키워드 기반 테스트가 최근 관심을 모으고 있는 이유는 테스트 케이스를 비전공자도 작성하기 쉽고 자동화와 재사용성이 높기 때문이다. 하지만 키워드가 제한되거나 관련 기능을 개발하는데 오래 걸릴 수 있어 문제가 될 수 있다[1,6]. 이와 같은 키워드 기반 테스트의 핵심은 키워드의 정의 방법이다. 테스터들이 작성하는 테스트 케이스나 선언하는 키워드가 사람의 성격이 다르듯이 제각기 다르기 때문에 키워드 기반 테스트 케이스를 작성하지 않았던 다른 테스터가 테스팅 프로그램을 인수인계 받을 경우 문제가 생긴다. 또한 키워드의 계층이 일정하지 않아 오랜 시간이 걸리며, 키워드 기반으로 작성하기위해 키워드를 추출하려면 어려움을 겪는다.

본 논문의 연구 주제는 프로그램 개발 과정에 키워드를 정확히 추출하는 방법을 연구하는 것이다. 애자일 개발 방법은 유스케이스(Use Case)나 유저스토리(User Story)를 사용한다. 이 중 제안하는 방법은 유스케이스에서 키워드를 추출하는 방법을 연구하고자 한다. 유스케이스에서 주요 키워드를 추출하였으면, 추출된 키워드를 바탕으로 키워드 기반 테스트 케이스를 작성하는 것이다. 키워드 기반 테스트 케이스는 오픈소스로 작성된 로봇프레임워크(Robot Framework)를 기준으로 작성한다. 위와 같은 방법으로 절차, 방법, 규칙을 작성하여 사례 연구(Case Study)로 ATM프로그램을 이용하여 적용해본다.

이 논문은 총 5장으로 구성되어 있다. 2장에서는 키워드 기반 테스팅과 키워드 추출 기술들을 살펴보고 3장에서는 본 논문에서 제안하는 유스케이스에서 테스트 키워드를 추출하는 방법과 테스트 케이스를 라이브러리화 하여 자동화 하는 방법에 대한 기술을 설명한다. 4장에서는 3장에서 언급한 기법을 사용하여 적용한 방법을 정리하여 설명한다. 마지막으로 5장에서 논문의 결론을 이야기한다.

2. 관련 연구

프로그램을 개발하고 테스트할 때 목적에 따라 여러 가지 테스팅이 사용된다. 하지만 대부분의 테스팅은 개발 경험이 있어야 데이터 중심의 테스트 스크립트를 작성할 수 있다. 본 논문에서 다루는 키워드 기반 테스트는 테스트를 구동시키는 여러 과정을 모르더라도 테스트 케이스를 만들고 이를 작성할 수 있다. 먼저 키워드 기반 테스팅과 키워드 추출 연구를 이번 장에서 설명한다.

2.1 키워드 기반 테스팅

2.1.1 자동화

키워드 기반 테스트란 국제 표준 ISO 29119 파트5에 기록이 되어 있으며, 명세 기반 테스트 설계 기법을 이용하여 테스트를 수행하는 접근법으로 테스트 자동화 및 테스트 자동화 프레임워크 개발을 지원하는 일반적인 방식이다. 모든 테스트 레벨과 다양한 유형의 테스트(기능, 안정성 테스트 등)에 적용할 수 있다[1].

키워드 테스팅을 위한 테스트 케이스 작성은 도구, 즉 테스트 스크립트와 관련되지 않은 프론트 단계와

테스트 스크립트 및 테스트 프레임워크에 의존하는 백엔드 단계로 이루어진다. 프론트 단계는 테스트 키워드 작성과 수행 문장 정의로 구성된다.

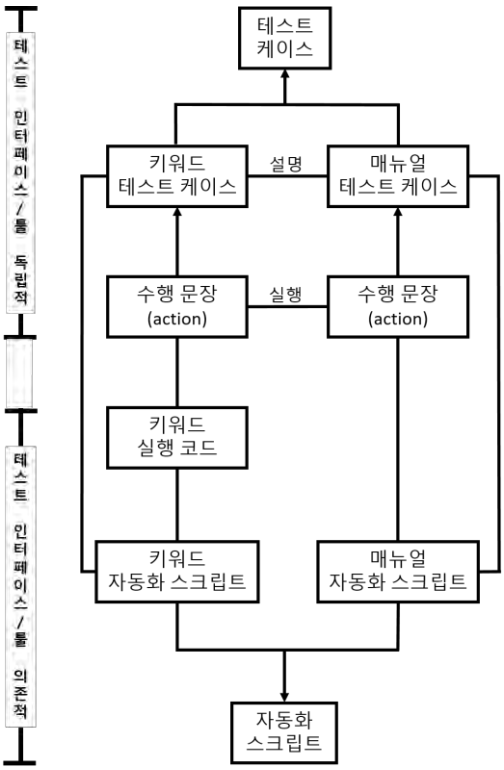


그림 1 키워드 테스트 케이스 작성 단계와 도구의존성[1]

그림 1에서 의존성이 높은 '키워드 실행 코드'와 '자동화 스크립트'는 키워드 테스트 케이스가 실행될 때 내부적으로 실행된다.

키워드 기반 테스트 자동화를 최종적으로 실행하기 위해서 키워드 기반 테스트를 실행할 backend 코드가 필요할 것이다. 해당하는 테스트 자동화를 쉽게 작성하기 위해서 로봇 프레임워크(Robot Framework)를 사용한다.

로봇 프레임워크는 Python 기반 도구로 작성되었다. 로봇 프레임워크에는 정의된 라이브러리가 있지만 Python 또는 Java로 작성된 라이브러리를 쉽게 작성할 수 있다. 필요한 라이브러리가 있다면 함수를 작성하는 것이 가능하다[2, 10]. 예를 들어 그림 2와 같이 기본으로 제공하는 라이브러리가 아닌 작성된 라이브러리를 사용하여 코드 작성에 들어가는 시간과 비용을 줄일 수 있다.

```python
from calculator import Calculator, CalculationError

class CalculatorLibrary(object):
    def __init__(self):
        self._calc = Calculator()
        self._result = ''

    def push_button(self, button):
        self._result = self._calc.push(button)

    def push_buttons(self, buttons):
        for button in buttons.replace(' ',''):
            self.push_button(button)

    def result_should_be(self, expected):
        if self._result != expected:
            raise AssertionError('%s != %s' % (self._result, expected))

    def should_fail(self, expression):
        try:
            self.push_buttons(expression)
        except CalculationError, err:
            return str(err)
        else:
            raise AssertionError("'%s' should have failed" % expression)
```

그림 2 로봇 프레임워크 라이브러리 예제

2.1.2 키워드 유형

키워드 기반 테스트는 키워드의 집합이다. 키워드를 계층별로 도메인 계층, 중간 계층, 테스트 인터페이스 계층으로 구성한다. 도메인 계층의 키워드는 각 소프트웨어 특성에 맞게 사용하는 용어로 구성하고 키워드 간 독립성을 갖도록 한다. 중간 계층은 로그인 과정을 예로 들면 로그인에 필요한 사용자, 암호 등의 키워드를 분리하여 구성하는 것이다. 이 키워드를 상위 레벨 키워드라고 한다. 상위 레벨 키워드는 의미를 가진 단위로 하위 레벨 키워드와 매개변수를 이용하여 테스트 케이스의 목적을 수행하도록 작성되는 키워드다. 마지막으로 테스트 인터페이스 계층의 키워드로는 테스트 워드와 매개변수를 이용하여 테스트 케이스를 상세하게 작성한다. 매개 변수로는 ID, PW 같은 변수를 이용하여 로그인 테스트를 수행한다. 이 키워드를 하위 레벨 키워드라고 한다. 하위 레벨 키워드는 메뉴나 버튼 선택을 위한 마우스 클릭이나 키보드 입력 등의 각 단계의 액션을 수행하는 역할을 한다.

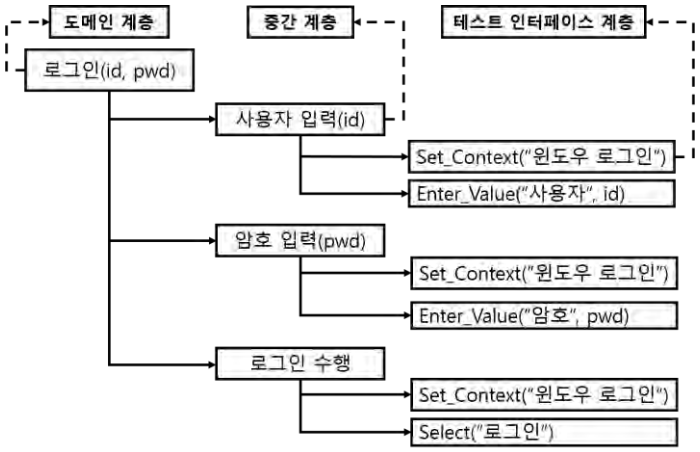


그림 3 계층을 이용한 복합 키워드[3]

키워드는 크게 단일 키워드와 복합 키워드로 나뉜다.

단일 키워드는 테스트 인터페이스 계층에 해당하는 키워드로 그림 3에 해당하는 테스트 인터페이스 계층에 작성된 “Set_Context”와 “Enter_value”를 뜻하고 이 키워드는 ‘사용자 설정’과 ‘암호 설정’ 키워드에서 다른 매개변수를 입력 받아 반복해서 사용된다.

복합 키워드는 여러 키워드를 조합하여 만든 키워드로 그림 3에서 ‘로그인’ 키워드가 단일 키워드인 ‘사용자 입력’, ‘암호 입력’, ‘로그인 수행’로 구성된다. 복합 키워드는 도메인 계층에서 이해할 수 있는 언어를 사용하여 상세한 정보를 몰라도 모두가 이해하는 수준으로 작성된다.

결국 키워드 유형을 정리하면 그림 4와 같이 나타낼 수 있다.

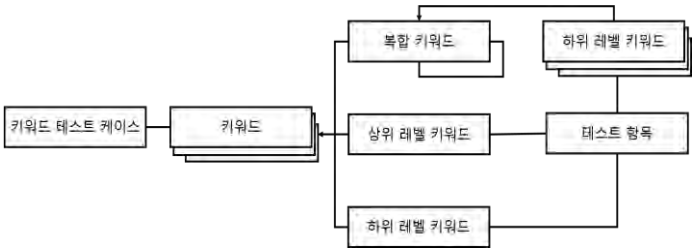


그림 4 복합 키워드 유형 구조[4]

키워드 기반 테스트의 장점은 첫째로, 테스트 케이스의 명확성과 이해도가 높아진다. 둘째로, 프로그래밍 지식 없이도 테스트 개발이 가능하다. 셋째로, 특정 도구 또는 프로그래밍 언어에 종속되지 않는다. 넷째로, 높은 재사용성을 가진다. 위와 같은 장점이 키워드 기반을 사용하는데 이점이 된다. 하지만 테스트 키워드를 어떤 근거로 작성하는지 무엇이 적합한 테스트 키워드인지는 정의되지 않고 있다.

2.2 데이터 기반 테스트

데이터 기반 테스트는 테스트 데이터가 테이블 또는 스프레드시트(엑셀) 형식으로 저장되는 소프트웨어 테스트 방법이다. 데이터 기반 테스트를 통해 테스터는 테이블의 모든 테스트 데이터에 대한 테스트를 실행하고 동일한 테이블에서 테스트 출력을 기대할 수 있는 단일 테스트 스크립트를 입력할 수 있다[11]. 이를 테이블 기반 테스트 또는 매개 변수 테스트라고도 한다.

데이터 기반 프레임워크는 입력 값을 데이터 파일에서 읽고 테스트 스크립트의 변수로 저장되는 자동화 테스트 프레임워크로써 테스터는 양수 및 음수 테스트 사례를 단일 테스트로 빌드 할 수 있으며, 입력데이터는 .xls, .xml, .csv 및 데이터베이스와 같은 데이터에 저장이 가능하다. 데이터 기반 테스트를 하는 이유는 테스터는 단일 테스트에 대해 여러 데이터 집합을 자주 가지고 있고 각 데이터 집합에 대한 개별 테스트를 만드는 데 시간이 많이 걸릴 수 있어 데이터 기반 테스트가 중요하다. 회귀 테스트 중에 여러 데이터 값 집합으로 응용 프로그램을 테스트할 수 있다. 하지만 단점으로는 구현 팀의 자동화 기술에 의존한다는 것이다.

위와 같은 차이로 인해 키워드 자동화 테스트가 데이터 기반 테스트보다는 테스트 케이스 구성이 더 체계적이고 재사용 라이브러리화 하기 쉽다. 데이터 기반 테스트 같은 경우에는 데이터를 기반으로 하여 테스트를 진행하므로 개발자에게 좋은 테스팅 도구 중 하나이지만, 개발 초기에 비개발자들도 이해할 수 있고 리그레션 테스팅에서 사용 가능한 키워드 기반 테스트가 테스트 주도 개발에 더 실용적이므로 테스트 키워드를 추출하는 방법을 제안하여 더 나은 키워드 기반 테스트를 소개한다.

2.3 키워드 추출 연구

자연어 처리(NLP) 기법을 사용하여 자연어로 표현된 테스트 케이스를 도출하는 논문도 있지만, 이러한 방법은 테스트 케이스 자동화 생성을 완전히 지원하지는 않는다[5]. 테스트 데이터 없이 자연어로부터 테스트 케이스를 생성하는 연구[6]와 테스트 케이스를 생성할 수 있지만 생성된 테스트 케이스에 입력 값을 넣기 위해 수동으로 입력해 주는 연구도 있다[7]. 최근에는 웹 기반 기능 테스트를 위한 자동화된 유저스토리 기반 접근 방식을 연구한 논문도 작성되었다[8].

하지만 위와 같은 논문들은 문장에서 글자를 추출하여 테스트 케이스로 작성하는 방법들이라면, 연구가 필요한 부분은 유스케이스에서 중요 키워드를 추출하여 키워드 기반 테스팅에 필요한 키워드를 사용하여 테스트 프로그램을 만드는 것이다. 다음 3장에서 바로 키워드를 추출하는 방법을 설명한다.

3. 유스케이스에서 키워드를 추출하는 방안

일반적으로 테스트 케이스를 작성할 때 코드를 보고 테스트 케이스를 제작하는 화이트박스 형식으로 작성하는 방법과 프로그램을 전체적으로 실행해보고 코드를 보지 않은 채 테스트 케이스를 제작하는 블랙박스 형식으로 이루어진다.

일반적인 시스템 수준의 블랙박스 테스트 케이스 작성 방법은 유스케이스를 자세히 살펴본 후 제시된 흐름에 맞게 테스트 케이스를 작성한다. 그림 5와 같이 주어진 유스케이스에 따라 테스트 케이스를 작성 후 테스트 케이스를 작성한 대로 잘 흘러가는지 다시 확인하는 과정이다.

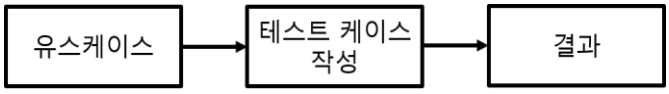


그림 5 유스케이스를 이용한 일반적인 테스트 케이스 작성 흐름도

이와 같은 방법은 유스케이스를 구성하는 각 이벤트 중 입출력이 출현하는 곳에서 테스트를 위한 입력 값과 예상 출력 값을 구함으로 테스트 케이스를 작성할 수 있다. 하지만 유스케이스에 표현된 테스트 단계를 진행하기 위하여 실제 값을 주고 예상 결과를 확인하는 것은 수동으로 하거나 Play-and-record 도구에 의존할 수밖에 없다.

본 논문에서는 위와 같은 작성 방안을 참고하여 그림 6과 같은 순서로 키워드 기반 테스트 케이스를 작성할 때 유스케이스에서 키워드를 추출하여 알맞은 테스트 케이스를 작성 후 결과를 시험할 수 있는 과정을 제안한다. 키워드를 추출하는 방법은 테스트 케이스를 입력값과 예상 결과값에서 분리하여 재사용 가능한 테스트 단위로 만들어 추후 리그레션 테스트에 많은 도움을 준다.

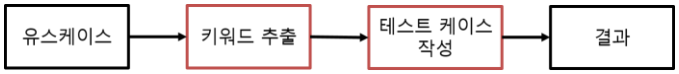


그림 6 유스케이스를 이용한 키워드 추출을 추가한 테스트 케이스 작성 흐름도

일반적인 테스트 케이스 작성 방법은 유스케이스를 작성하고 이를 기초로 입력부분과 테스트 입력 값을 찾아 테스트 케이스를 작성하고 예상 결과를 도출해낸다.

이런 방법으로는 키워드 기반 테스트에 적합하지 않을 뿐 더러 테스트 케이스를 작성하는 테스터마다 다른 키워드를 추출하여 작성할 가능성이 높다. 따라서 모든 테스터가 키워드를 보고 테스트 케이스를 작성할 때 기준이 있어야 유스케이스에서 주요 키워드를 추출한 것이 정확하고 통일성이 있으며 추출한 테스트 키워드를 이용하여 테스트 케이스를 작성 한 뒤 테스트를 위한 스크립트를 더 효과적으로 작성할 수 있다.

3.1 유스케이스 키워드 추출

본 논문에서 제시할 키워드 추출 방법을 소개한다. 가장 처음은 유스케이스 안에 이벤트를 관찰하는 것이다. 두번째로는 유스케이스 안에 있는 명사를 추출하는 것이다. 다음 각 추출한 키워드를 나열하여 후보를 만든다. 키워드 후보를 보고 어디에 무엇을 쓸지 결정을 한다. 모든 키워드가 결정되어 테스트 케이스가 작성되었으면 예시가 될 유스케이스를 하나 만들어서 주어진 입력 값을 넣어 키워드가 잘 적용됐는지 검증을 한다. 마지막으로 추출한 유스케이스의 키워드가 중복되거나 잘못 설정한 경우가 있을 수 있으므로 확인하고 중복되었으면 적합한 키워드를 찾아 대체한다.

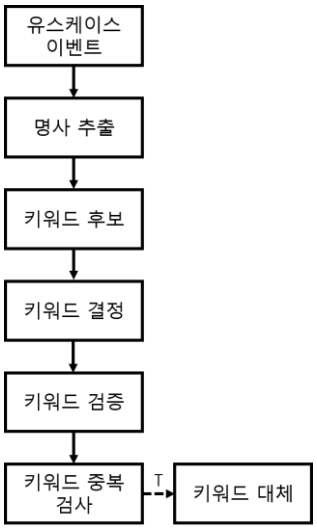


그림 7 제안하는 유스케이스 키워드 추출 방법

우선 키워드를 추출하기 전에 유스케이스에 해당하는 부분을 살펴보고 키워드를 추출할 준비를 한다. 일단 시나리오 이름은 유스케이스 단위의 테스팅을 위한 키워드로 띄어쓰기를 포함하여 추출한다.

다음과 같은 키워드는 복합 키워드 유형 중 도메인 계층이라고 볼 수 있다. 즉 액터는 유스케이스를 작동하는 작동자로서 만족 조건에는 꼭 필요한 키워드이다. 이 키워드를 가지고 만족조건 테스트 케이스 제작을 효율성 있게 할 수 있다. 또한 사전 조건은 유스케이스를 보고 시작하기 전 완료된 작업들을 의미한다. 사전 조건에 나오는 키워드는 액터가 어떠한 행동을 하였는지 잘 알려주는 키워드이다. 마지막으로 만족 조건은 테스트 케이스로 제작되는 부분이며, 유스케이스 키워드 추출의 가장 중요한 곳이다. 만족 조건에서 액터가 포함된 조건들을 추출하고 키워드로 제작하는 것이 가장 큰 목적이다. 액터가 포함된 만족 조건은 테스트 케이스를 만들기 전 단계로 복합 키워드 중 중간 계층 즉 상위 레벨 키워드라고 볼 수 있다. 이러한 상위 레벨 키워드를 기반으로 테스트 인터페이스 계층 즉 하위 레벨 키워드 작성이 가능하다.

앞서 설명한 키워드 추출 준비를 맞추었으면 각 키워드를 추출할 예정인 줄마다 명사를 추출한다. 시나리오 이름은 고유 명사이므로 띄어쓰기를 무시한 채 전체 문장을 추출한다. 액터는 사용자 자체를 의미하며 이 액터에서 언급한 작동자를 포함한 만족조건의 문장을 추출하는 것이 효과적이다. 사전 조건은 본 유스케이스의 행동하기 전에 작동이 완료된 것을 의미한다. 마지막으로 만족조건은 유스케이스 추출에서 중요한 부분이며, 키워드가 가장 많이 작성되는 부분이다. 액터에서 언급된 작동자를 포함한 줄을 추출하여 해당하는 줄에서 키워드를 추출한다. 키워드를 추출한 다음에 해당하는 키워드 언급 횟수를 표로 만들어 정리한다. 정리한 키워드 중 언급 횟수가 가장 많은 키워드는 작동자 그리고 액터가 포함된 행동이 될 것이며, 언급 횟수가 적은 키워드들은

고유하거나 적은 행동을 했다는 것을 알 수 있다. 작성된 키워드 추출 방법 순서대로 진행 한 뒤 테스트 케이스를 제작하고, 제작한 테스트 케이스를 로봇 프레임워크에 맞게 바꾼다. 키워드 추출부터 테스트 케이스 제작까지 모든 순서가 수동으로 이루어진다.

위에서 작성한 키워드 추출 방법을 알고리즘으로 정리하면 다음과 같이 나타낸다.

알고리즘 1 키워드 추출 알고리즘

```
1.  유스케이스 불러오기;
2.  명사(키워드) 추출;
3   select 명사
4.    case 시나리오 이름 : 키워드 추출;
5.    case 액터: 키워드 추출;
6.        액터 키워드가 포함된 만족 조건 추출;
7.        추출한 만족 조건 키워드 추출;
8.  추출한 키워드를 표로 제작;
9.  표를 참고하여 키워드의 우선 순위를 지정;
10. for each 추출한 키워드 {
11.   키워드 카운트;
12.   If (사용할 키워드) 키워드 검증;
13.   If (키워드 중복) 적당한 키워드로 대체;
14.   테스트 케이스 제작;
15. }
```

그림 8 키워드 추출 알고리즘

유스케이스 명세는 “시나리오 이름”, “설명”, “액터”, “사전 조건”, “만족 조건”으로 이루어진다. 앞서 설명한 키워드 추출 방법과 알고리즘을 적용한 과정을 표 1의 예를 들어 설명한다.

표 1 유스케이스 명세 예시

시나리오 이름	회원가입
설명	사용자가 회원가입을 한다.
액터	사용자
사전 조건	사용자는 인터넷을 실행한 상태이다.
만족 조건	
1.	사용자가 인터넷 주소를 입력한다.
2.	시스템이 입력한 인터넷 주소를 출력한다.
3.	사용자가 회원가입 버튼을 클릭한다.
4.	시스템이 회원가입 창을 출력한다.
5.	사용자가 아이디 : ‘hgd20’, 비밀번호 : ‘test123!’로 회원가입을 진행한다.
6.	시스템이 회원가입 완료 창을 출력한다.
7.	시스템이 사이트메인으로 돌아간다.

표 1과 같이 시나리오 이름은 테스트 케이스 이름을 나타내는 것으로 “회원가입”은 테스트 케이스 이름이면서 동시에 조건에 자주 나올 키워드로 추측이 가능하다. 설명은 유스케이스 명세의 전체적인 흐름을 나타내는 부분으로 “사용자”가 “회원가입”을 한다는 내용이다. 액터는 “사용자”로 조건에 “사용자”라는 키워드가 들어가면 테스트 케이스로 작성이 가능하다.

사전 조건은 만족 조건을 시작하기 전에 완료되었거나 필요한 내용들이 있다. 마지막으로 만족 조건에는 표 1에 언급된 “사용자” 키워드만 아니라 “시스템”이라는 키워드도 있다. “시스템” 키워드는 “사용자” 키워드가 어떤 이벤트를 발생한 후 프로그램이나 사이트에 이벤트 결과를 출력한다. 추가로 만족 조건에 “사용자” 라는 키워드에 “입력”, “클릭”, “진행” 이라는 키워드와 관련이 있어 해당 키워드를 사용하거나, 중복이 있다면 다른 키워드로 대체하여 사용하면 된다. “시스템” 키워드에 언급되는 “출력”, “돌아간다”와 같은 키워드가 나올 때, 테스트 케이스를 작성 후 실행 할 때, 해당 이벤트가 발생 되는지 확인을 해야하는 키워드이다.

다음에는 위와 같은 유스케이스에서 키워드를 추출한 것을 이용하여 테스트 케이스 작성 방법에 대해 설명한다.

3.2 키워드 기반 테스트 케이스 작성 방법

키워드 기반 테스팅은 테스트 시나리오를 키워드에 맞게 저장하여 스크립트를 통해 구분하기가 용이한 방법으로 스크립트를 추가하거나 삭제, 복사 등도 간편하게 가능하다. 본 논문에서는 키워드 기반 테스트 프레임워크인 로봇 프레임워크를 사용할 예정이다. 로봇 프레임워크를 쉽게 사용하기 위해 키워드 기반 GUI 편집기인 RIDE를 사용하여 테스트 케이스를 작성하였다.

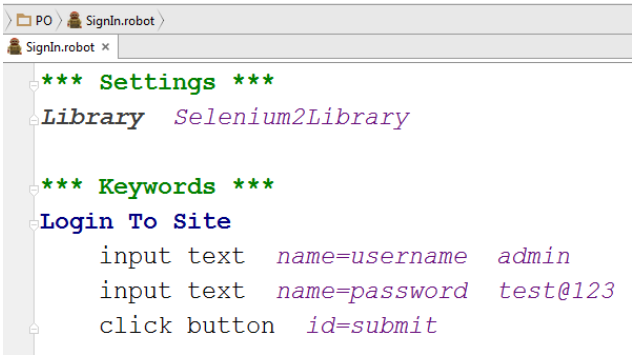


그림 9 로봇 프레임워크 테스트 케이스 예시[10]

그림 9와 같이 설정(Settings)에는 문서에 대한 설명과 기본적으로 설치되어 있거나 로봇 프레임워크를 지원할 수 있도록 작성된 외부 라이브러리를 호출하여 설정을 완료해준다.

키워드에는 각 사용자가 사용할 수 있는 키워드를 제작하여 사용할 수 있다. 그림 9 작성된 데이터는 외부 라이브러리인 셀레니움을 설정하였고, 키워드에는 “username”에 아이디인 {admin}을, “password”에 비밀번호인 {test@123}을 넣어주고, id가 “submit”설정되어있는 버튼을 오른쪽 클릭하는 것을 “Login To Site”라는 키워드로 제작하였다.

본 논문에서는 프로그램을 사용하여 예제를 제작하는

것이기 때문에 윈도우 GUI 자동화와 스크립팅을 위해 설계된 AutoIt 라이브러리를 사용하여 작성한다. AutoIt 라이브러리를 부르기 위해서는 그림 10과 같이 작성하여 선언하면 사용이 가능하다.

```
*** Settings ***
Library           AutoItLibrary    WITH NAME    AI
```

그림 10 AutoIt 라이브러리 선언

4. ATM을 사용한 키워드 기반 테스트 사례 연구

앞서 3장에서 설명한 유스케이스로 키워드를 추출하는 방법과 테스트 케이스를 작성하는 방법을 ATM 프로그램을 사용하여 본 논문에서 제안하는 방법으로 설명한다.

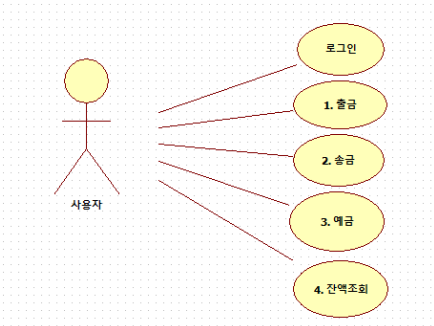


그림 11 ATM 유스케이스 다이어그램(사용자)

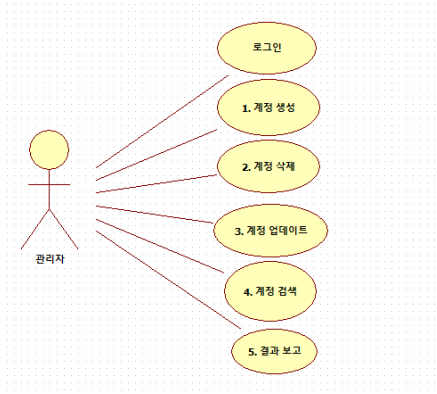


그림 12 ATM 유스케이스 다이어그램(관리자)

표 2 ATM 유스케이스 명세 예제 (출금)[9]

시나리오 이름	ATM 출금
설명	500 출금 예정
액터	사용자
사전 조건	사용자는 로그인을 완료했다. 사용자는 1000의 초기자본을 가진다
만족 조건	
1.	시스템에 1.출금, 2.송금, 3.예금, 4.잔액조회까지 선택 창이 뜬다.
2.	사용자가 메인화면에서 [1.출금] 선택한다.
3.	시스템에 a)Fast Cash, b)Normal Cash 선택 창이 뜬다.
4.	사용자가 출금옵션을 a)Fast Cash 선택
5.	시스템에 1) 500, 2)1,000, 3)2,000 ~ 7)20,000 까지 출금 금액 선택과 취소 모드 선택창이 뜬다
6.	사용자가 출금금액 1)500 선택한다
7.	시스템에 출금 500 재확인 알람이 나온다
8.	사용자가 재확인 알람 동의(y) 선택한다.
9.	시스템에 성공적으로 출금이 되었다는 문장 출력한다.
10.	시스템에 사용자 남은 잔액 표시한다.
11.	시스템에 출금내역 영수증 출력 선택 창이 뜬다.
12.	사용자가 출금내역 영수증 출력 동의(y) 선택한다.

위 그림 11은 ATM 사용자의 유스케이스 다이어그램이고 그림 12은 ATM 관리자의 유스케이스 다이어그램이다. 표 2는 ATM프로그램의 유스케이스 명세의 예제이다. 이 3개의 예제를 가지고 키워드를 추출하는것과 테스트케이스 작성까지의 과정을 설명한다.

가장 먼저 유스케이스에서 키워드를 추출하기 위해 해당하는 다이어그램과 명세 부분을 크로스 체크한다. 명세 예제가 ATM 출금이므로 사용자가 로그인 뒤에 출금을 선택할 것이라는 것을 고려한 뒤 시작한다.

ATM 유스케이스 명세 예제의 각 이벤트를 잘 확인한다. 일단 시나리오 이름부터 사전 조건까지 확인하면, 시나리오 이름은 “ATM 출금”으로 ATM 프로그램에서 출금을 할 예정이라는 것을 유스케이스 다이어그램에서 확인이 가능하다. 그 후 설명에 “500 출금 예정”이라고 적혀 잇는데 500의 가치를 출금할 예정이라는 것이다. 액터는 “사용자”로 사용자가 작업자라는 의미다. 사전 조건은 “로그인을 완료 했다”, “사용자는 1000의 초기 자본을 가진다” 두 가지를 보면 로그인 유스케이스를 완료한 상태이며, 초기자본 1000으로 설정되어있다. 위의 내용을 다시 입력하면 사용자는 1000의 초기자본에서 500이라는 출금을 할 예정이며 로그인을 완료했다는 것이다.

위에서 테스트 키워드를 추출하면 {ATM 출금, 500, 사용자, 로그인, 완료, 1000, 초기자본}이다. {ATM 출금}은 시나리오 이름에서 추출하였고 전체적인 흐름을 담당하기 때문에 테스트 케이스 이름으로 작성하였다. 설명에서 추출한 {500, 출금}은 500이라는 금액을 출금할 것이라는 내용을 나타낸다.

액터에서는 {사용자}라는 키워드를 추출하여 사용자와 관련된 내용을 키워드로 만들면 적합한 테스트 케이스가 작성된다는 것을 알 수 있다. 사전 조건에서는 {로그인 완료, 1000, 초기자본}을 추출하여 로그인은 현재 완료된 상태이고, 사용자는 1000의 자본을 가지고 있다고 알 수 있다. 이 뒤로 만족 조건의 각 키워드를 추출하여 로봇 프레임워크에 맞는 테스트

케이스로 작성한다.

만족 조건은 12개의 시나리오로 이루어져있다. 그리고 처음 명사에 누가 이벤트를 발동하는지 나와있기 때문에 액터에 뽑았던 키워드를 비교하며 테스트 케이스를 작성하면 빠른 테스트 케이스를 작성할 수 있다.

각 만족 조건의 키워드를 추려내면 다음과 같다.

1.{시스템, 1.출금, 2.송금, 3.예금, 4.잔액조회, 선택창}
2.{사용자, 1.출금, 선택}
3.{시스템, a)Fast Cash, b)Normal Cash, 선택}
4.{사용자, 출금옵션, a)Fast Cash, 선택}
5.{시스템, 1) 500, 2)1,000, 3)2,000 ~ 7)20,000, 출금 금액 선택, 취소 모드, 선택 창}
6.{사용자, 출금금액, 1)500, 선택},
7.{시스템, 출금 500, 재확인 알람}
8.{사용자, 재확인 알람, 동의(y), 선택}
9. {시스템, 성공적, 출금, 문장, 출력}
10.{시스템, 사용자, 남은 잔액, 표시}
11.{시스템, 출금내역, 영수증, 출력, 선택 창}
12.{사용자, 출금내역, 영수증, 출력, 동의(y), 선택}

위 만족 조건에서는 위에 언급한 것과 같이 액터는 “사용자”이기 때문에 입력해서 테스트 케이스를 작성할 예제는 {사용자}라는 키워드가 포함된 조건의 키워드를 사용하여 작성하면 수월하다.

시스템이 들어간 1, 3, 5, 7, 9, 10, 11은 시스템의 출력이나 단순한 정보의 표시이기 때문에 작성을 안해도 무방하다. 하지만 위에 제외한 줄의 키워드를 확인해 보면 {선택 창}이 자주 나오는 것을 알 수 있다. 시스템이라는 언급이 없더라도 {선택 창}이라는 키워드가 언급되면 그 줄은 사용자의 입력을 요청하는 이벤트라는 것을 알 수 있다. 이제 나머지 2, 4, 6, 8, 12같은 경우 테스트 케이스를 작성을 해야 할 줄이다. 이 5줄의 키워드를 조합해서 테스트 케이스를 작성한다. 5줄을 “사용자”가 포함된 키워드 줄이라고 표현하겠다.

“사용자”가 포함된 키워드를 보면 2.{사용자, 1.출금, 선택}, 4.{사용자, 출금옵션, a)Fast Cash, 선택}, 6.{사용자, 출금금액, 1)500, 선택}, 8.{사용자, 재확인 알람, 동의(y), 선택}, 12.{사용자, 출금내역, 영수증, 출력, 동의(y), 선택}로 추릴 수 있다. 코드로 변경하기 전에 “사용자”가 포함된 키워드 줄의 키워드를 중복을 포함하여 추출해 보면 {사용자, 1.출금, 선택, 출금옵션, a)Fast Cash, 출금금액, 1)500, 재확인 알람, 동의(y), 출금내역, 영수증, 출력}로 나타난다. 겹치는 키워드는 변수로 작성을 하고 겹치지 않는 키워드는 값을 입력하여 넣는 방식으로 작성하여야 한다. 위에 추출한 키워드를 표로 정리하면 다음과 같다.

표 3 키워드 횟수 정리표

키워드	횟수
사용자, 선택	5
동의(y)	2
1.출금, 출금옵션, a)Fast Cash, 출금금액, 1)500, 재확인 알람, 출금내역, 영수증, 출력	1

위와 표 3와 같이 [사용자, 선택] 두 키워드가 모든 줄에 언급된 것을 살펴보면, “사용자”라는 키워드가 아니라도 “선택”이라는 키워드가 들어가 있으면 ‘사용자가 무엇인가를 선택한다’라는 판단이 가능하며, 테스트 케이스로 작성이 가능하다. {동의} 키워드는 2번 언급이 되었는데 이와 같은 경우 동의라는 키워드는 사용자에게 무엇인가 선택하게 하였고 그 선택을 동의하였다는 의미로 받아드릴 수 있고 이 키워드가 있으면 사용자의 액션이 필요한 테스트 케이스로 작성할 수 있다라고 예측이 가능하다.

“사용자” 키워드가 포함된 만족 조건을 로봇 프레임워크를 이용한 테스트 케이스로 바꿔보면, 그림 13과 같다.

1	AI.send	1	
2	sleep	3	
3	AI.send	a	
4	sleep	3	
5	AI.send	1	
6	sleep	3	
7	AI.send	y	
8	sleep	3	
9	AI.send	y	

그림 13 ATM 출금 테스트 케이스

‘AI.send’는 AutoIt 라이브러리를 사용해 프로그램에 보내줄 입력 값을 이야기 한다. ‘AI.send 1’이라고 하면 1이라는 수를 프로그램에 전송한다. ‘sleep’은 대기하라는 명령어로 자동으로 빠르게 입력 값을 전송해 주기 때문에 누락되거나 버그가 걸릴 수 있어 3(sec)를 기다려 주어 입력 시간을 대기시킨다. 위의 키워드에서 추출한 입력 값인 {1.출금, a)Fast Cash, 1)500, 동의(y)}는 콘솔 창에 입력하기 편하게 숫자나 소문자로 선택하기 쉽도록 작성된 프로그램을 적용하기 때문에 출금을 위해 1번을, Fast Cash모드를 선택하기 위한 a, 출금금액 500을 위한 1, 동의를 위한 y를 각 넣어주어 테스트가 간결하고 잘 작동 될 수 있도록 작성을 해주어 그림 13와 같은 테스트 케이스를 작성한 것이다. 그림 13와 같이 작성하게 되면 입력 값이 동일하거나 실수할 수 있는 요소가 많아 각 값에 키워드로 만들어 주어 헛갈리지 않게 작성할 수 있다.

위에서 추출했던 키워드 중 겹치는 값들은 {사용자, 선택, 동의}등 이였는데 {사용자}는 명세의 액터에서도 나왔던 키워드로써 프로그램 작동자임을 알 수 있는 키워드로 겹쳐도 문제가 없었고 {선택}키워드도 마찬가지로 프로그램 작동자인 “사용자”가 선택을 하는 액션이기 때문에 문제가 없었다. {동의} 키워드는

{선택} 키워드랑 같은 키워드로 별 다른 문제가 없었다.

횟수가 한 번씩 언급된 키워드들은 변수로 지정해 사용할 수 있는 키워드 들이다. 위의 키워드를 변수화 하여 테스트 케이스를 작성하면 그림 14와 같이 나타낼 수 있다.

1	AI.send	${main}
2	sleep	3
3	AI.send	${mode}
4	sleep	3
5	AI.send	${money}
6	sleep	3
7	AI.send	${check_op}
8	sleep	3
9	AI.send	${Receipt}

그림 14 변수를 사용한 테스트 케이스

그림 14를 보면 변수 ${main, mode, money, check_op, Receipt}를 사용하여 순서대로 메인 선택 화면, 모드 선택, 출금액 선택, 출금 확인 여부와 마지막으로 출금 결과 영수증을 준다는 내용을 변수로 처리 하였고 위와 같은 코드를 만들면 다른 선택을 위해 새로운 코드 제작 필요 없이 변수에 해당하는 값을 넣으면 편리하게 사용할 수 있다. 또한 위 값들을 키워드로 설정해 둔다면 키워드 테스트 케이스를 간결하고 보기 편리하게 정리할 수 있을 것이다.

5. 결론 및 향후 연구내용

본 논문에서는 유스케이스를 이용한 키워드 기반 테스트 케이스 작성 방안에 대해 연구하였다. 유스케이스에서 키워드 추출을 사용하는 방법을 설명하고 추출한 키워드를 바탕으로 테스트 케이스를 제작하였다. 키워드의 추출 방법으로는 유스케이스 안의 이벤트를 확인하여 명사를 추출한다. 다음 추출한 키워드를 나열하여 후보를 만든 뒤, 어디에 무엇을 쓸지 결정하는 방법으로 진행을 하였다. 위와 같은 방법을 사용하면 키워드를 추출하는데 시간이 조금 걸리지만 한번에 정리를 하여 보기 쉽게 관리 할 뿐만 아니라 테스트 케이스를 위한 변수 선택에도 용이하다. 키워드 추출을 위한 알고리즘을 작성하고 최적화하면 더 빠르게 키워드를 정리하여 사용할 수 있을 것이다.

유스케이스를 보고 테스트 케이스를 작성하기위해 많은 노력을 들일 필요 없이 주요 키워드를 추출하여 테스트 케이스를 작성한다. 이 방안의 성능을 더 향상시키기 위해 유스케이스에서 주요 키워드 추출 후 테스트 케이스까지 작성하는 자동화 프로세스를 연구하면 효율적인 테스트 케이스를 작성할 수 있을 것이다. 또한 자연어 처리를 사용한 언어의 형태 분석과 같은 인공지능 기술을 결합한 연구를 진행하는 것이다. 이 기술을 사용하여 사용 빈도가 높은 키워드 추출을 자동화하는 것이 가능할 것이다.

6. 참고문헌

[1] 정상미,"더 괜찮은 QA가 되기 위한 프랙티컬 테스트 자동화", 프리렉, 2018.

[2] STRESNJAK, Stanislav; HOCENSKI, Zeljko. "Usage of robot framework in automation of functional test regression", *Proc. 6th Int. Conf. Softw. Eng. Adv.(ICSEA)*. pp. 30-34, 2011.

[3] "ISO/IEC/IEEE International Standard - Software and systems engineering -- Software testing -- Part 5: Keyword-Driven Testing," in ISO/IEC/IEEE 29119-5 First edition, 2016.

[4] 황준선. 탐색적 키워드 기반 자동화 테스트 모델 설계 및 검증. 석사학위논문, 동국대학교 대학원, 2020.

[5] Sinha, A., S.M. Sutton Jr, and A. Paradkar. "Text2Test: Automated inspection of natural language use cases", *Proc. of 2010 Third International Conference on Software Testing, Verification and Validation*, pp. 155-164, 2010.

[6] Sarmiento, E., J.C.S. do Prado Leite, and E. Almentero. "C&L: Generating model based test cases from natural language requirements descriptions", *Proc. of 2014 IEEE 1st International Workshop on Requirements Engineering and Testing (RET)*, pp. 32-38, 2014.

[7] M.J.Escalona, J.J.Gutierrez, M.Mejías, G.Aragón, I.Ramos, J.Torres, F.J.Domínguez. "An overview on test generation from functional requirements", *Journal of Systems and Software*, Vol. 84, No. 8, pp. 1379-1393, 2011.

[8] Masud, M., Iqbal, M., Khan, M. U., & Azam, F. "Automated user story driven approach for web-based functional testing", *International Journal of Computer and Information Engineering*, Vol.11, No.1, pp.91-98, 2017.

[9] 최은만, 소프트웨어 공학의 모든 것, 생능출판사, 2020.

[10] Robot Framework, http://robotframework.org

[11] 데이터 기반 테스트 원리, www.guru99.com/data-driven-testing.

[12] Rwemalika, Renaud, et al. "On the Evolution of Keyword-Driven Test Suites." *Proc. of 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*, IEEE, pp.335-345, 2019.

[13] Rashm, Neha Bajpai, "A Keyword Driven Framework for Testing Web Applications", *Proc. of (IJACSA) International Journal of Advanced Computer Science and Applications*, Vol. 3, No. 3, 2012.

이슈 관리를 위한 다중 레이블 분류 봇

박도제 [1], 양혜진 [1], 최광현 [1], 이선아 [1], 강성원 [2]

경상대학교 [1], 한국과학기술원 [2]

parkdo619511@gmail.com, yhj8636@naver.com, jpg04260@naver.com,
saleese@gnu.ac.kr, sungwon.kang@kaist.ac.kr

A Multi-label Classification Bot for Issue Management

Do-je Park[1], Hye-jin Yang[1], Kwang-hyun Choi[1], Seonah Lee[1], Sungwon Kang[2]
Gyeongsang National University[1], KAIST[2]

요 약

많은 개발자들이 오픈 소스 소프트웨어를 개발할 때 사용하는 GitHub 플랫폼은 이슈 관리 시스템을 제공한다. 이 시스템을 활용하여 관련자들은 소프트웨어 문제가 생기거나 기능적인 요청을 이슈로 보고할 수 있다. GitHub 이슈 관리 시스템은 이슈 보고서 양식 작성을 제공하고 개발자들이 레이블을 만들어 이슈를 분류하도록 허용한다. 그러나 레이블링 작업은 수작업으로 이루어지기 때문에 많은 노력이 들어가고 부정확한 레이블링이 쉽게 발생할 수 있다. 유명한 프로젝트의 경우 하루에 수십 개에서 수백 개의 이슈보고서가 작성되는데 개발자가 이슈 보고서를 직접 읽고 피드백을 주기 위해서는 많은 노력과 시간이 요구된다. 이러한 문제점을 해결하기 위한 기존 연구로 단일한 레이블을 붙이는 연구들이 존재한다. 그러나 실제 이슈 보고서에는 여러 개의 레이블이 붙어야 하는 이슈 보고서들도 적지 않게 존재한다. 따라서 본 연구에서는 GitHub 프로젝트 관리자들이 이슈 보고서를 읽고 피드백 주는데 들이는 노력을 줄일 수 있도록 복수개의 레이블을 자동으로 붙이는 다중 레이블링 봇을 제안한다.

키워드 : 머신 러닝, 소프트웨어 산출물, 소프트웨어 유지보수, 이슈 보고서, FastText, 다중 레이블

1. 서 론

많은 개발자들이 오픈 소스 소프트웨어 개발을 할 때 GitHub와 같은 오픈 소스 플랫폼을 사용한다. GitHub 등의 오픈 소스 플랫폼에서는 이슈 관리 시스템을 제공하여 관련자들이 프로젝트들을 사용하면서 발생하는 문제 사항이나 불편한 점들을 이슈 보고서를 통해 보고할 수 있도록 한다. 프로젝트 관리자들은 보고된 이슈보고서들을 읽고 피드백을 주거나 이슈를 이슈 관리 시스템에 반영하여 해당 프로젝트의 문제점들을 개선한다. 이와 같이 이슈 관리 시스템은 개발에 도움을 주지만, 하루에 수십 개에서 수백 개까지 보고되는 이슈보고서들을 관리자들이 일일이 읽고 분류하고 피드백을 주는 데에는 많은 시간과 노력이 들어간다[1]. 그리고 이슈보고서의 요구사항이나 특징들을 추출한 뒤 머신 러닝 기법을 사용하여 단일 레이블로 분류를 하는 연구들도 존재한다[2-5]. 그러나 실제 프로젝트의 이슈 보고서들은 단일 레이블로만 분류되지 않고, 다수의 레이블로 분류되어야 하는 이슈 보고서들도 많다..

본 연구는 다중 레이블링을 이용하여 이러한 이슈보고서들을 단일 레이블로 분류하는 기존 연구의 한계점을 극복하고자 한다. 이를 위하여 자동으로 분류하는 다중 레이블 분류 봇을 제안한다. 제안하는 분류 봇은 사용자가 제출하는 이슈보고서를 자동으로 분류하여 다수의 레이블을 붙여준다. 이러한 자동 분류를 하기 위해서 제안하는 분류 봇을 만들기 위해 머신 러닝 모델로는 FastText 분류기를 이용하여 학습 모델을 형성하였고, 이후 GitHub에서 지원하는 Git APP 제작 도구를 활용하였다.

2. FastText 분류기

문서 분류를 위해서 다양하고 많은 머신러닝 기법들이 존재한다. 하지만 이 기법들은 데이터의 양이 많아지면 많아질수록 학습에 많은 시간을 걸리고 많은 메모리를 요구한다. 이 문제를 해결하기 위해 성능은 동등하지만 시간적인 소모는 압도적으로 적은 FastText 분류기를 사용한다.

다중 레이블 분류에 대한 접근 방식은 다중 레이블 손실 함수를 최소화하는 것이다. 이러한 손실 함수를 구성하기 위해서는 다중 레이블 학습을 이진 또는 다중 클래스 문제로 축소하여서 보는 것이다. FastText 분류기는 기본적으로 다중 레이블 분류를 위해 손실 함수로 아래의 수식1과 같이 One-vs-all(ova)함수를 지원한다. FastText 분류기는 아래의 손실함수를 최소화 하는 것을 목표로 한다. 이에 대해서 시그모이드 또는 소프트 맥스 함수를 가정하고 다중 레이블에 대한 손실 함수를 구한다.

$$\sum_{i\in[L]}\{-y_i \cdot log\frac{e^{f_i}}{1+e^{f_i}} - (1-y_i)\cdot log\frac{1}{1+e^{f_i}}\}$$

수식 1 FastText 분류기의 손실함수(ova)

본 연구의 제안 방법은 위의 수식과 같이 L개의 독립적인 이진 분류 모델을 학습한 뒤 소프트맥스 함수를 활용하여 각 $\mathbf{y_i} \in \{\mathbf{0}, \mathbf{1}\}$을 예측한다. 이에 따라 제안 방법은 학습 데이터가 입력 되면 이를 학습한 뒤 각각의 레이블에 대해서 이진 분류를 실행한다. 기존 연구와 제안 방법의 가장 큰 차이는 FastText의 손실함수로 기존연구에서는 Hirecal softmax를 이용하여 단일 레이블에 초점을 맞추어서 진행을 하였으나 본 연구에서는 Fasttext의 One-vs-all을 적용하여 각 레이블에 대해서 이진 분류로 접근을 하였다..

3. 다중 레이블 봇

다중 레이블 분류 봇은 GitHub내에서 작동하는 앱이다. 다중 레이블 분류 봇은 자체적으로 작동하며 GitHub에서 제공하는 API를 통하여 유연하게 자동화된 작업 흐름을 수행한다. 다중 레이블 분류 봇은 사용자 계정 또는 조직의 프로젝트에 대한 개인 엑세스 토큰을 통해 관리 권한을 부여받아 해당 프로젝트의 데이터를 관리 하거나 전송 할 수 있다.

4. 평가 기준

평가 기준으로는 머신 러닝 평가에서 가장 많이 쓰이는 혼동 행렬을 사용하여 각각 Precision(정확도), Recall(재현율), F1-score(조화평균)을 이용하였다. 평가에 대해서는 데이터셋에 존재하는 레이블을 실제 정답으로 한다. 이 때 Precision(정확도)은 모델이 추천한 레이블 중 실제 레이블 수를 모델이 추천한 모든 레이블 수로 나눈 값을 각 이슈 보고서마다의 정확도로 보고 이를 평균한 값을 의미한다. Recall(재현율)은 모델이 추천한 레이블 중 실제 레이블 수를 실제 레이블 수로 나눈 값을 각 이슈 보고서마다의 재현율로 보고 이를 평균한 값을 의미한다. F1-score(조화평균)은 정확도와 재현율의 조화평균을 의미한다.

데이터 셋의 정밀한 평가를 위해 제안 방법의 평가는 10-fold cross validation 방법으로 진행하였다. 기본적인 비교 방법의 경우 전체 데이터 셋이 n:n으로 한번 비교를 하지만 k-fold의 경우 데이터 셋의 전체를 구간별로 나누어 기본적인 비교 방법보다 정밀하게 비교 할 수 있다.

실험에 있어 비교되는 파라미터는 아래 표와 같다.

표 1 비교 파라미터 값

기존 연구	제안 방법
'hs'	'ova'
'k'=1	'k'=2
나머지 = default	나머지 = default

5 실험 결과

기존의 Ticket-tagger[9]의 제안 방법과 본 연구의 제안방법과의 레이블 예측 성능의 비교 평가를 3개의 프로젝트를 대상으로 진행하였다. 평가 결과는 그림 1,2,3과 같다.

정밀도에서는 기존방법과 유사한 성능을 보였으나, 재현율에서 제안 방법이 뛰어난 성능을 보였다. 기존 방법[9]은 0.26, 021, 0.37의 재현율을 보였으나, 제안 방법의 재현율은 0.5, 0.41 그리고 0.76으로 0.20~0.39정도 더 높은 결과가 나왔다. 해당 결과에서 recall에서의 차이는 기존 연구[9]와 본 연구의 제안방법과의 차이를 잘 보여준다. 이는 2개 이상의 레이블을 추천 할 수 있으므로 다중의 레이블을 가진 이슈 보고서를 대상으로 더 높은 재현율을 보였음을 의미한다.

결과적으로 정확도와 재현율의 조화 평균인 F1-score에서는, 재현율의 상대적 우위에 따라 제안 방법이 기존 연구보다 다중 레이블을 고려한 학습에서 0.14~0.27정도 더 높은 성능을 보였다.

표 2 Vscode 평가 결과

	기존 연구[5]	제안 방법
Precision	0.62	0.59
Recall	0.26	**0.5**
F1-score	0.37	**0.54**

표 3 Dartlang 평가 결과

	기존 연구[5]	제안 방법
Precision	0.5	0.5
Recall	0.21	**0.41**
F1-score	0.29	**0.45**

표 4 TypeScript 평가 결과

	기존 연구[5]	제안 방법
Precision	0.80	0.81
Recall	0.37	**0.76**
F1-score	0.51	**0.78**

8. Reference

[1] Q. Fan, Y. Yu, G. Yin, T. Wang, and H. Wang, "Where is the road for issue reports classification based on text mining?" ESEM 2017, pp. 121-130. [Online].

[2] Sohrawardi, S. J., Azam, I., & Hosain, S. (2014). "A comparative study of text classification algorithms on user submitted bug reports" ICDIM 2014,pp. 242-247. IEEE.

[3] Pandey, N., Sanyal, D. K., Hudait, A., & Sen, A. (2017). "Automated classification of software issue reports using machine learning techniques: an empirical study" Innovations in Systems and Software Engineering, 13(4), 279-297.

[4] Fan, Q., Yu, Y., Yin, G., Wang, T., & Wang, H. (2017, November). "Where is the road for issue reports classification based on text mining?". ESEM 2017 (pp. 121-130). IEEE.

[5] Kallis, R., Di Sorbo, A., Canfora, G., & Panichella, S. (2019). Ticket Tagger: Machine Learning Driven Issue Classification. ICSME 2019, pp. 406-409. IEEE.

적합도와 메타-학습을 결합한 하이브리드 학습 기반 소프트웨어 신뢰도 예측 모델 선정 기법

이낙원[1], 류덕산[2], 조일훈[3], 송재근[4], 백종문[1]
[1]한국과학기술원, [2]전북대학교, [3]LIG 넥스원, [4]국방과학연구소

A Hybrid Learning-based Software Reliability Prediction Model Selection Technique combining Goodness-of-fit and Meta-learning

Nakwon Lee[1], Duksan Ryu[2], Ilhoon Cho[3], Jeakun Song[4], Jongmoon Baik[1]
[1]KAIST, [2]JBNU, [3]LIG Nex1, [4]ADD

요 약

다양한 종류의 소프트웨어 신뢰도 예측 모델이 존재하지만 이들 중에서 주어진 실패 데이터의 신뢰도를 예측하는데 성능이 좋은 모델을 선택하는 것은 어려운 일이다. 이러한 문제를 해결하기 위해서 적합도 기반의 학습을 통한 모델 선택 기법과 메타-지식 기반의 학습을 통한 모델 선택 기법이 제안되었다. 그러나 적합도 기반 학습 기법은 적합도를 계산할 수 없는 비 분석적 모델을 선택할 수 없다는 문제가 있다. 또한 메타-지식 기반 학습 기법은 비 분석적 모델을 선택할 수 있지만 사용한 메타-지식이 예측 성능과 관련성이 낮은 경우에 예측 성능이 낮아지는 경우가 있다.

본 연구에서는 비 분석적 모델을 선택할 수 있으면서도 높은 모델 선택 정확도를 확보하기 위하여 적합도와 메타-지식을 모두 사용하는 하이브리드 학습 기반 모델 선택 기법을 제안한다. 하이브리드 학습 기법은 메타-지식 기반의 학습을 통해 비 분석적 모델이 주어진 실패 데이터에서 높은 예측 성능을 보이는지 먼저 판단한다. 비 분석적 모델이 높은 예측 성능을 보일 것으로 판단되면 비 분석적 모델을 사용하고 그렇지 않으면 적합도 기반 학습 기법을 이용하여 모델을 선택한다. 기존의 학습 기반 모델 선택 기법들과 제안한 하이브리드 기법을 비교하기 위하여 222개의 신뢰도 예측 태스크를 이용한 광범위한 실험을 실시했다. 실험 결과 평균 예측 오차를 예측 성능 기준으로 사용한 경우에 하이브리드 기법이 기존 학습 기반 기법들을 능가하였다. 제안한 하이브리드 기법을 이용하여 소프트웨어 개발자들이 쉽고 자동화된 방법으로 보다 정확한 신뢰도 예측 결과를 얻을 수 있을 것이며 결과적으로 더 높은 소프트웨어 신뢰도를 확보할 수 있을 것이다.

1. 서 론

소프트웨어 신뢰도 예측은 소프트웨어 시험 과정에서 발생하는 실패 데이터를 분석하여 미래의 소프트웨어 신뢰도를 예측하는 기법이다. 소프트웨어의 특성에 따라 실패 데이터의 특성 또한 달라지기 때문에 정확한 신뢰도를 예측하기 위해 지난 50여년간 100개가 넘는 소프트웨어 신뢰도 예측 모델들이 개발되어왔다 [1]. 그런데 다양한 모델들 중에서 주어진 실패 데이터의 신뢰도를 예측하는데 좋은 성능을 보이는 모델이 무엇인지 선택하는 것은 어려운 일이다 [2].

이러한 문제를 해결하기 위해서 많이 사용되는 방법이 적합도를 이용하는 것이다. 분석적 예측 모델이 수집된 실패 데이터에 얼마나 잘 적합 되었는지 평균 제곱 오차와 같은 적합도 기준을 계산하고 적합도가 높은 모델을 선택한다 [3]. J Park과 J Baik의 연구에서는 단순히 하나의 적합도 기준을 사용하는 것에서 멈추지 않고 다양한 적합도 기준과 실제 예측 성능의 관계를 학습함으로써 정확도 높은 모델 선택 기법을 개발하였다 [4]. 그런데 최근에 개발되고있는 기계학습 또는 탐색 기반의 비 분석적 예측 모델들은 적합도를 계산할 수 없다 [5]. 따라서 비 분석적 모델들이 좋은 성능을 보임에도 불구하고 적합도 기반 선택 기법들은 비 분석적 모델들을 선택할 수 없는 문제가 발생한다 [6].

메타 학습 기반 모델 선택 기법은 비 분석적 모델들을 선택하기 위해서 실패 데이터 자체로부터 추출한 분산도, 변동 정도와 같은 메타-지식만을 이용하여 모델을 선택하는 기법이다 [6]. 이 기법은 실패 데이터의 메타-지식 특성과 예측 모델의 성능 간의 관계를 학습한 모델을 만든다. 적합도 정보를 요구하지 않기 때문에 분석적 모델과 비 분석적 모델을 모두 선택할 수 있다. 그러나 메타-지식과 신뢰도 예측 성능의 관계는 충분히 연구되지 않은 부분으로 적합도

기준에 비해 모델 선택 성능이 검증되지 않았다. 특히 기존 연구에서는 선택 대상 모델이 최대 4개밖에 되지 않아 성능 평가가 충분히 이루어졌다고 볼 수 없다.

본 연구에서는 비 분석적 모델과 분석적 모델을 모두 비교하여 모델을 선택할 수 있으면서도 높은 추정 성능을 확보하기 위하여 적합도 기준과 메타-지식 특성을 모두 사용하는 하이브리드 기법을 제안한다. 하이브리드 학습 기법은 메타-지식기반 학습을 이용하여 비 분석적 모델이 주어진 실패 데이터에 대하여 얼마나 높은 예측 성능을 보일지 판단한다. 비 분석적 모델이 높은 예측 성능을 보일 것으로 판단되면 비 분석적 모델을 선택하여 신뢰도 예측을 수행한다. 비 분석적 모델의 예측 성능이 낮을 것으로 판단되면 적합도 기반 학습을 통한 예측 기법을 이용하여 분석적 모델 중 하나를 선택한다.

기존 기법들을 비롯하여 제안하는 하이브리드 기법의 성능을 평가하기 위해 대규모 실험을 통한 경험적 평가를 수행한다. 본 연구에서는 37개의 실패 데이터로부터 222개의 신뢰도 예측 태스크를 추출하여 성능 평가 대상으로 사용하였다. 실험 결과 평균 예측 오차를 예측 성능 평가 기준으로 사용하는 경우에 제안하는 하이브리드 기법이 기존의 적합도 기반 학습을 통한 선택 기법과 메타-지식 기반 학습을 통한 선택 기법을 능가했다. 또한 추가적인 분석을 수행하여 전체적인 예측 성능의 차이가 없는 기법 간에도 어떤 실패 데이터에 적용하느냐에 따라 성능 차이가 크게 발생할 수도 있음을 확인하였다.

2. 배경지식 및 관련 연구

2.1. 분석적 소프트웨어 신뢰도 예측 모델

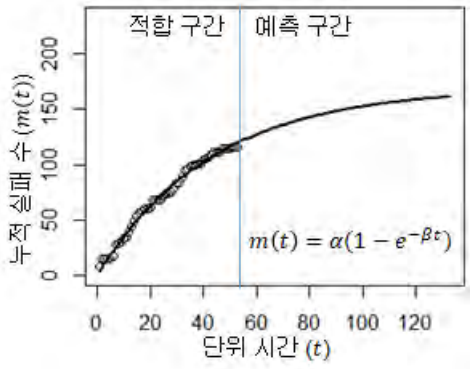


그림 1 분석적 모델을 사용 예시

분석적 소프트웨어 신뢰도 예측 모델은 소프트웨어의 실패가 발생하는 패턴이 특정한 통계적 모델을 따른다고 가정한다. 가정에 따른 수식을 수집된 실패 데이터에 대하여 인자 추정을 통해 적합 시킴으로써 신뢰도 예측을 수행한다. 이때 실제 데이터에 모델이 얼마나 잘 적합 되었는지를 나타내는 것이 적합도 기준이다. 대표적인 적합도 기준은 평균 제곱 오차 (Mean Squared Error), 평균 절대 오차 (Mean Average Error)등이 있다.

그림 1은 실패 데이터에 대하여 분석적 모델을 적용하여 신뢰도를 예측하는 예시를 보여준다. 단위 시간은 시험을 수행한 기간을 특정 기간에 따라 나눈 것이며 각 단위 시간에서의 누적 실패 수를 표시한다. 분석적 모델 식 $\mathrm{m(t)} = \alpha(\mathrm{a} - e^{-\beta t})$에서 α와 β는 적합 구간의 실제 수집된 실패 데이터를 통해 추정하여 설정된다. 수식이 완성되면 예측 구간에 그려진 누적 실패 수를 통해 미래의 신뢰도를 예측할 수 있다. 이때 적합 구간에 대해서도 수식의 값이 존재하기 때문에 적합을 통해 계산된 수식의 값과 실제 실패 데이터를 비교하여 적합도를 계산할 수 있다.

2.2. 비 분석적 소프트웨어 신뢰도 예측 모델

비 분석적 소프트웨어 신뢰도 예측 모델은 통계적 모델에 대한 가정 없이 순수하게 현재까지의 실패 데이터 패턴 만을 이용하여 미래의 실패 발생 패턴을 예측한다 [5]. 예를 들어 그림 2와 같이 53 단위시간의 실패 데이터가 수집되었고 3 단위 시간으로부터 다음 단위 시간을 예측하는 방식으로 54 단위 시간의 실패를 예측한다면 비 분석적 모델은 t단위 시간의 실제 실패 수를 의미하는 m_t에 대하여 $\mathrm{m(t)} = \mathrm{f}(m_{t-1}, m_{t-2}, m_{t-3})$와 같이 표현되며 f 를 학습을 통해 알아내기 위해 (m_3, m_2, m_1) 부터 (m_{52}, m_{51}, m_{50}) 까지 49개의 입력을 사용한다. 이렇게 학습된 f 를 이용하여 $\widehat{m(54)} = \mathrm{f}(m_{53}, m_{52}, m_{51})$ 을 계산하여 54 단위 시간의 실패를 예측할 수 있다. 55 단위 시간과 같이 더 이상 수집된 실패 데이터를 이용한 예측이 불가능한 경우에는 $\widehat{\mathrm{m(55)}} = \mathrm{f}(\widehat{m(54)}, m_{53}, m_{52})$ 와 같이 예측된 값을 입력으로 사용하여 다음 단위 시간의 신뢰도를 예측할 수 있다 [7].

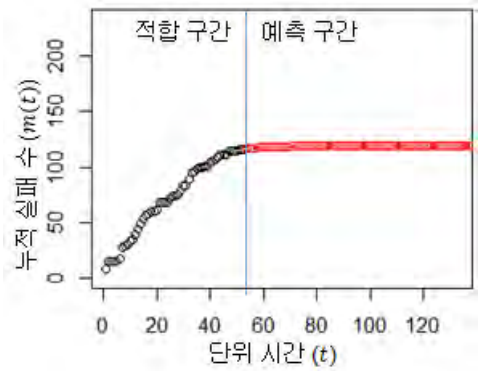


그림 2 비 분석적 모델 사용 예시

비 분석적 모델은 분석적 모델과는 다르게 적합 구간에 대해서 출력 값을 얻을 수 없기 때문에 적합도의 개념이 적용될 수 없다. 또한 수식 형태가 아니기 때문에 단위 시간 별 이산 값만을 얻을 수 있는

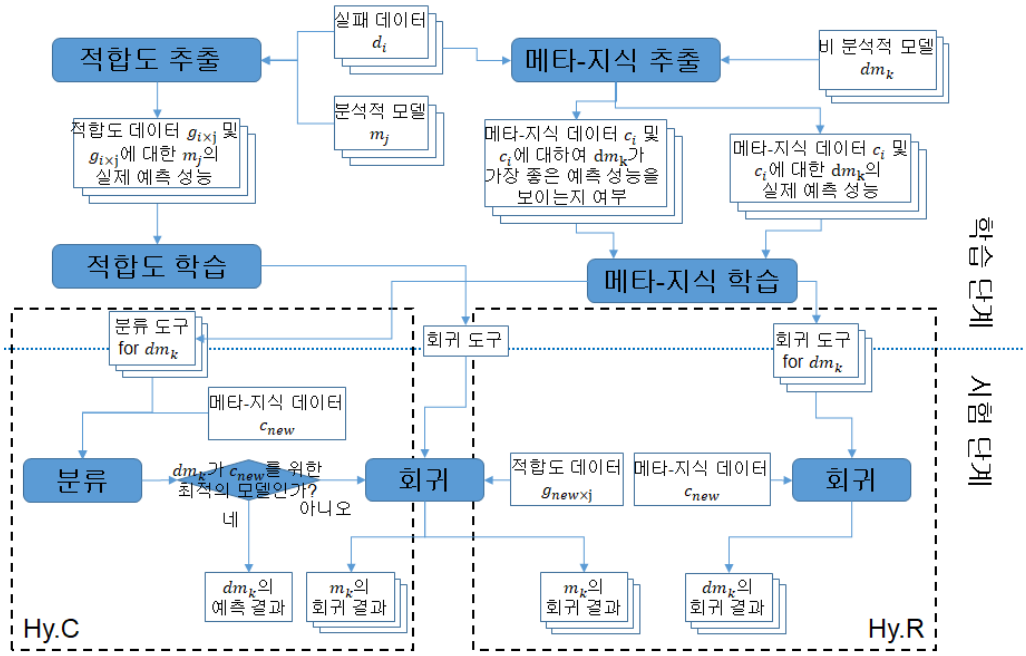


그림 3 두 가지 하이브리드 기법 Hy.C와 Hy.R의 개요

것 또한 분석적 모델과는 다르다.

2.3. 학습 기반 소프트웨어 신뢰도 예측 모델 선택

J Park과 J Baik의 연구에서는 10가지 적합도 기준을 이용하여 적합도 값과 실제 예측 성능과의 관계를 학습하였다 [4]. 학습에 사용된 10가지 적합도 기준은 평균 제곱 오차, 평균 절대 오차, R^2, Noise, Bias, Variation, Predictive Ratio Risk, 가중 최소 자승 오차, 적합 최종 오차, 적합 평균 오차이며 참조 논문에서 각 적합도 기준의 계산법을 확인할 수 있다. 학습 기법은 결정 나무를 사용하였으며 여러 개의 모델이 최적의 모델로 선택될 수 있기 때문에 최적의 모델로 선택된 여러 기법을 조합하는 방법 또한 제안하였다. 선택 대상인 분석적 모델은 13개의 모델 (Goel-Okumoto, Generalized Goel, Gompertz, Inflection S-shaped, Modified Duane, Musa-Okumoto, Yamada Imperfect debugging1/2, Delayed S-shaped, PNZ, Pham-Zhang, Pham-Zhang Imperfect debugging, Logistic growth curve)로 다양한 통계적 가정을 반영한 모델들을 선택 대상으로 사용했다.

Rafael Caiuta 등의 연구는 메타-지식을 이용하여 각각의 실패 데이터의 (메타-지식)특성에 따라 어떤 모델이 가장 좋은 성능을 보이는지 다중 분류 모델을 학습했다 [6]. 따라서 학습에 사용되는 데이터는 실패 데이터의 메타-지식 특성 값들과 그 실패 데이터에서 최적의 성능을 보이는 모델이다. 4가지 메타-지식 특성 (Variance, Inclination, Auto Correlation, Noise)을 사용하였으며 각각의 계산 방법은 참조 논문에 자세히 나와있다. 이 연구에서는 2가지 분석적 모델 (Jelinski-Moranda, Geometric)과 2가지 비 분석적 모델 (ANN, GP2)을 대상으로 선택하였으며 단위 시간당 실패 수를 수집한 실패 카운트 형태의 데이터가 아니라 실패 발생 간의 소요시간 형태의 데이터를 사용하였다.

3. 다양한 적합도 기준과 메타 지식을 이용한 하이브리드 학습 기반 소프트웨어 신뢰도 예측 모델 선정 기법

다양한 적합도 기준과 메타-지식을 모두 사용하기 위하여 본 논문에서는 (1) 분류 기반 하이브리드 학습 기법 (Hy.C)과 (2) 회귀 기반 하이브리드 학습 기법 (Hy.R)의 두 가지 하이브리드 학습 방법을 제시한다.

그림 3은 Hy.C와 Hy.R의 개요를 나타낸 것이다. Hy.C는 메타-지식 기반의 학습을 통해 실패 데이터에 대하여 비 분석적 모델의 예측 성능이 최적인지 분류할 수 있는 분류 도구를 만든다. 분류 도구를 이용해 분류를 진행했을 때 비 분석적 모델이 최적인 것으로 판단한 경우 비 분석적 모델을 선택한다. 비 분석적 모델이 최적의 모델이 아닌 것으로 판단한 경우 기존의 적합도 기반 학습 기법을 이용하여 분석적 모델 중 하나를 선택한다. 각각의 비 분석적 모델마다 하나씩 분류 도구를 학습함으로써 여러 개의 비 분석적 모델을 고려하여 모델 선택이 가능하다.

Hy.R은 메타-지식 기반의 학습을 통해 주어진 실패 데이터에 대하여 각각의 비 분석적 모델의 예상 예측 성능을 계산하는 회귀 도구를 만든다. 적합도 기반 학습을 통해 계산된 분석적 모델들의 예상 예측 성능과 메타-지식 학습을 통해 계산된 비 분석적 모델들의 예상 예측 성능을 비교하여 최적의 모델을 선정한다.

적합도 학습과 메타-지식 학습을 통해 분류 및 회귀 도구들을 생성하는 시점까지 학습 단계이며 이후

단계에서 새롭게 주어진 메타-지식 데이터와 적합도 데이터를 이용하여 시험을 수행한다.

4. 성능 평가 실험 설정

4.1. 연구 질문

적합도 학습 기반 모델 선택 기법, 메타 학습 기반 모델 선택 기법, 그리고 하이브리드 학습 기반 모델 선택 기법들의 성능을 평가하기 위해서 두개의 연구 질문을 설정하고 이에 답하기 위한 실험을 수행하였다.

● **RQ1: 예측 성능 측면에서 다른 모든 기법들을 능가하는 기법(들)이 있는가?**

예측 성능이 다른 모든 기법들을 능가하는 기법이 존재하는 경우 모든 실패 데이터에 대하여 해당 기법을 사용하면 높은 예측 성능을 얻을 것으로 기대할 수 있다.

● **RQ2: 예측 성능이 가장 좋은 그룹의 기법들 간에 차이가 존재하는가?**

예측 성능이 좋은 것으로 식별된 기법들이 여러 개 존재할 때 이들 기법 간에 차이가 없다면 이들 중 아무 기법을 사용하더라도 비슷한 결과를 얻을 수 있을 것이다. 그러나 이들 기법 간에 차이가 존재한다면 어떤 실패 데이터에 적용하느냐에 따라 차이가 발생할 수 있다. 그림 4는 이러한 현상이 나타날 수 있는 간단한 예시를 보여준다. 두 기법 T1과 T2는 여섯 개의 실패 데이터 (ds1~ds6)에 왼쪽과 오른쪽 두 경우 모두 서로를 능가하지 못하는 모습을 보여준다. 그러나 왼쪽 그림에서는 실패 데이터 별로 예측 성능의 차이가 거의 없기 때문에 T1과 T2가 다르다고 보기 어렵지만 오른쪽 그림에서는 실패 데이터 별로 예측 성능의 차이가 크기 때문에 어떤 실패 데이터에 대하여 신뢰도 예측을 수행하느냐에 따라 T1과 T2의 성능 편차가 클 수 있다.

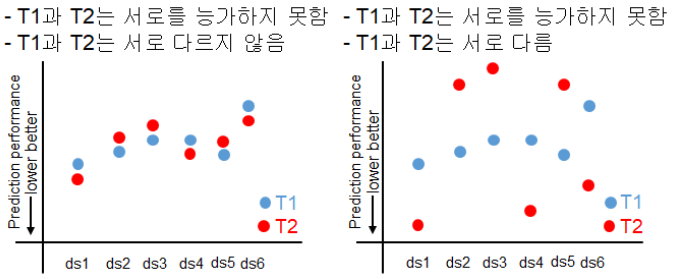


그림 4 두 기법이 서로를 능가하지 못하지만 실패 데이터 별로 다른 성능을 보일 수 있음을 보여주는 예시

4.2. 실험 대상 실패 데이터

본 실험에서는 37개의 실패 데이터를 대상으로 실험을 수행하였다. 각각의 실패 데이터는 시험 수행 단위 시간 별 누적 실패 수를 수집한 것으로 다양한 단위 시간과 결함 수를 가지고 있다. 표 1은 37개 실패 데이터의 특성을 간단하게 보여준다.

표 1 검증 대상 실패 데이터 개요

#	응용 유형	총 실패 수	출처
1	실시간 명령 및 제어	136	[9]
2		54	[9]
3		38	[9]
4		53	[9]
5	전자 시스템	461	[10]
6		211	[10]
7	항공우주 시스템	133	[10]
8		224	[10]
9		351	[10]
10		188	[10]
11		367	[10]
12	상용 소프트웨어	73	[9]
13		100	[11]
14		120	[11]
15		61	[11]
16		42	[11]
17	군용 소프트웨어	38	[9]
18		41	[9]
19		101	[9]
20		108	-
21	무선 네트워크 시스템	22	[12]
22		181	[12]
23	의료용 시스템	231	[13]
24		245	[13]
25		83	[13]
26	실시간 제어	535	[10]
27		198	[10]
28	데이터 베이스 응용	328	[15]
29	모니터링 및 실시간 제어	481	[14]
30		266	[16]
31		86	[14]
32	오픈 소스	74	[17]
33		50	[17]
34		58	[17]
35		85	[17]
36		54	[17]
37		54	[17]

본 실험의 대상 기법들은 학습 기반 기법들이기 때문에 37개 실패 데이터중에서 36개 데이터를 학습에 사용하고 나머지 1개 실패 데이터를 검증에 사용하는 리브-원-아웃 (Leave-One-Out) 검증 방식을 적용하였다.

각각의 실패 데이터에 대하여 X%의 파티션의 데이터 포인트를 모델 적합에 사용하고 나머지 100-X%의 데이터 포인트를 예측하는 것으로 예측 성능을 평가한다. X는 [40,50,60,70,80,90] 의 6개 파티션으로 설정하였다. 결과적으로 222 (37*6)회의 신뢰도 예측 태스크에 대하여 모델 선정을 수행한다.

4.3. 예측 성능 측도

본 실험에서는 정규화된 최종 오차 (Normalized End-Point error, NEP)와 정규화된 평균 예측 오차 (Normalized Mean Error Of Prediction, NMEOP)를 모델 선택 기법의 예측 성능을 의미하는 측도로 사용한다. 먼저 각각의 실패 데이터에 대하여 비교대상인 모델들의 최종 오차 (End-Point error, EP) 또는 평균 예측 오차 (Mean Error Of Prediction, MEOP) 값을 계산한다. EP와 MEOP는 예측한 마지막 단위 시간 q, 적합에 사용한 마지막 단위 시간 p, 단위 시간 q에서 실제 누적 실패 수 m_q, 그리고 단위 시간 q에서 예측된 누적 실패 수 $\widehat{m(q)}$ 를 이용하여 아래 수식에 의해 계산된다.

$$\mathrm{EP} = |m_q - \widehat{m(q)}|$$

$$\mathrm{MEOP} = \frac{\sum_{i=p}^{q} |m_i - \widehat{m(i)}|}{q-p+1}$$

계산된 모델 별 EP(MEOP)값을 비교하여 정규화된 EP(MEOP) 값을 계산한다. 모델 j의 실패 데이터 i에 대한 NEP(MEOP)값 $\mathrm{NEP(NMEOP)}_{i\times j}$는 모델 j 의 실패 데이터 i 에 대한 EP(MEOP)값 $\mathrm{EP(MEOP)}_{i\times j}$, 14개 모델의 실패 데이터 i에 대한 EP(MEOP)의 최대값 $\max(\mathrm{EP(MEOP)}_{i\times J})$, 최소값 $\min(EP(MEOP)_{i\times J})$ 를 이용하여 아래 수식과 같이 계산된다.

$$\mathrm{NEP(NMEOP)}_{i\times j} = \frac{\max(\mathrm{EP(MEOP)}_{i\times J}) - \mathrm{EP(MEOP)}_{i\times j}}{\max(\mathrm{EP(MEOP)}_{i\times J}) - \min(EP(MEOP)_{i\times J})}$$

따라서 NEP(NMEOP) 값이 1에 가까울수록 선택 대상 모델 중 가장 예측 성능이 좋은 모델에 가까운 성능을 보인 것이다. 모델 선택 기법의 예측 성능은 기법이 선택한 모델의 NEP(NMEOP) 값으로 표현한다. 실패 데이터 별로 예측 성능의 편차가 있을 수 있기 때문에 실패 데이터 별로 동일한 기준에서 비교하기 위하여 정규화된 값을 측정하였다.

4.4. 비교 대상 기법

● MSE

MSE는 적합도 기준 중 하나인 평균 제곱 오차가 가장 작은 모델을 선택하는 기법이다. 적합도 기준을 이용하므로 2장에서 설명한 13개의 분석적 모델 중 하나의 모델을 선택 할 수 있다.

● DDM

DDM은 비 분석적 모델의 일종인 재귀적 전략 기반의 서포트 벡터 회귀 기법을 의미한다. 선택 기법이 아닌 비 분석적 모델 자체의 결과 값을 의미한다 [5].

● GOFL.sp

GOFL.sp는 적합도를 이용한 학습 기법이다. J Park과 J Baik의 연구에서 사용한 10개의 적합도 기준을 이용하여 회귀 모델을 생성한다 [4]. 회귀 모델 학습에는 로지스틱 회귀 기법을 사용하였다. GOFL.sp는 학습을 할 때 학습 데이터 중에서 검증 데이터와 파티션이 일치하는 데이터들만 사용하여 학습한다. 예를 들어 검증 데이터가 40% 파티션에 해당한다면 216 (36개 실패 데이터*6 파티션)개의 학습 데이터 중에서 40% 파티션에 해당하는 36개의 실패 데이터 만을 이용하여 학습한 후 검증을 실시한다. GOFL.sp는 J Park과 J Baik의 연구에서 선택 대상으로 사용한 13개의 분석적 모델 중 하나의 모델을 선택한다.

● GOFL.add

GOFL.add는 적합도를 이용한 학습 기법이다. GOFL.add는 GOFL.sp와 거의 유사하지만 사용하는 학습 데이터를 증가시켰다는 것과 실패 데이터를 설명하는 메타-지식 하나를 추가하여 학습한다는 차이가 있다. GOFL.add는 학습 데이터의 양을 증가시키기 위해 다른 파티션의 데이터와 추가적으로 생성한 데이터들을 사용한다. 추가적으로 데이터들을 생성하기 위해 기존의 X%의 파티션으로 100-X%의 데이터 포인트를 예측하는 데이터 이외에 X%의 파티션으로 X+10%의 데이터 포인트를 예측하는 실패 데이터들을 추가한다. 따라서 하나의 검증 태스크에 대하여 GOFL.add가 학습에 사용하는 학습 데이터는 216 (36*6)개의 기존 데이터와 540 (36*15)개의 추가 데이터를 합하여 756개이다.

그런데 이렇게 파티션 구분없이 모든 데이터를 한꺼번에 사용할 경우 적합도 값은 동일하지만 예측 구간의 길이가 달라지면서 예측 성능이 달라지는 현상이 발생한다. 예측 구간의 길이에 따라 성능이 달라지는 현상 또한 학습하기위해서 PRatio라는 새로운 메타-지식 기준을 추가한다. PRatio는 모델 적합에 사용되는 데이터 포인트의 수 p와 예측하는 데이터 포인트의 수 k를 이용하여 아래 수식과 같이 정의된다.

$$\mathrm{PRatio} = \frac{p}{p+k}$$

GOFL.add는 10개의 적합도 기준과 PRatio를 포함한 11개의 기준을 이용하여 회귀 모델을 학습한다.

● Meta.sp

Meta.sp는 메타-지식 기반 학습을 통한 모델 선택 기법이다. Meta.sp는 Rafael Caiuta 등의 연구에서

사용한 4개 메타-지식 특징을 사용한다. 학습은 결정 나무를 이용했다.

Meta.sp는 검증 데이터의 파티션과 일치하는 36개의 학습 데이터만을 이용하여 분류 모델을 학습한다. Meta.sp는 13개의 분석적 모델과 DDM을 포한한 14개 모델 중 하나를 선택한다.

● Meta.add

Meta.add는 학습에 사용하는 데이터를 증가시킨 것과 하나의 메타-지식 기준을 추가하여 학습한다는 것을 제외하고 나머지 부분은 Meta.sp와 동일하다. Meta.sp는 GOFL.add에서 수행한 것과 마찬가지로 756개의 학습 데이터로부터 메타-지식 특징을 추출하여 학습에 사용한다. 파티션에 상관없이 학습데이터를 이용하기 때문에 PRatio를 추가한 총 5개의 메타-지식 기준을 사용하여 예측 구간의 길이에 따라 성능이 달라지는 현상을 학습한다.

● Hy.C.sp

Hy.C.sp는 2장에서 설명한 분류 모델 기반 하이브리드 학습을 이용한 모델 선택 기법이다. 메타-지식 학습을 위해 결정 나무 기법을 이용하였다. J Park과 J Baik의 연구에서 사용한 10개의 적합도 기준과 Rafael Caiuta 등의 연구에서 사용한 4개의 메타-지식 특성을 학습에 사용한다. 13개의 분석적 모델과 DDM을 포함한 14개의 모델 중 하나를 선택한다. Hy.C.sp는 학습할 때 검증 데이터의 파티션과 동일한 파티션에 해당하는 학습 데이터들만을 사용한다.

● Hy.C.add

Hy.C.add는 학습 데이터를 증가시켰다는 점과 적합도 기반 학습과 메타-지식 기반 학습을 할 때 PRatio를 추가하여 학습한다는 점을 제외하고는 Hy.C.sp와 동일하다.

● Hy.R.sp

Hy.R.sp는 2장에서 설명한 회귀 모델 기반 하이브리드 학습을 이용한 모델 선택 기법이다. 메타-지식 학습을 위해 로지스틱 회귀 기법을 사용하였다. 10개의 적합도 기준과 4개의 메타-지식 기준을 사용하여 학습한다. 13개의 분석적 모델과 DDM을 포함한 14개의 모델 중 하나를 선택한다. 검증 데이터와 동일한 파티션에 해당하는 학습 데이터들만을 학습에 사용한다.

● Hy.R.add

Hy.R.add는 학습 데이터를 증가시켰다는 점과 적합도 기반 학습과 메타-지식 기반 학습을 할 때 PRatio를 추가하여 학습한다는 점을 제외하고는 Hy.R.sp와 동일하다.

● BEST

BEST는 모든 검증 데이터에 대하여 최적의 선택을 하는 경우를 가정한 것이다. 항상 예측 성능이 가장 높은 모델을 선택한다. 다른 기법들이 실제 정답에 얼마나 근사할 수 있는지 확인하기 위해 BEST를 포함하였다.

4. 성능 평가 실험 결과

4.1. RQ1. 예측 성능 측면에서 다른 모든 기법들을 능가하는 기법(들)이 있는가?

그림 5와 그림 6은 222개 검증 데이터에 대한 각 비교대상 기법의 NEP값과 NMEOP값을 상자 그림으로 나타낸 것이다. NEP를 예측 성능 기준으로 사용한 경우 BEST를 제외한 10개의 비교대상 기법에서 큰 차이를 보이지 않는 것을 확인할 수 있다. 각 검증 데이터에 대한 최적의 기법은 NEP값이 항상 1이기 때문에 BEST는 모든 검증 데이터에 대하여 1의 NEP값을 갖는다.

NMEOP를 예측 성능 기준 값으로 사용하는 경우에는 다른 기법들에 비하여 메타-지식 기반 학습을 통한 모델 선택 기법들 (Meta.sp와 Meta.add)이 상대적으로 낮은 성능을 보이는 것으로 나타났다.

기법들 간의 성능을 보다 정확하게 비교하기 위하여 통계적 검정을 수행하였다. NEP와 NMEOP 각각에 대하여 한 기법이 다른 기법에 비하여 유의미하게 큰 중위 값을 보이는지 통계적 검정을 수행하였다. 윌콕슨 순위합 검정 (Wilcoxon rank sum test)을 쌍을 이뤄 수행 하였으며 95% 신뢰 수준에서 검정을 수행하였다. 귀무가설 (H0)과 대립가설 (H1)은 아래와 같다.

- H0: 왼쪽 기법의 중위 값이 오른쪽 기법의 중위 값보다 크지 않다.
- H1: 왼쪽 기법의 중위 값이 오른쪽 기법의 중위 값보다 크다.

검정의 p-value가 0.05보다 작은 경우에 대립가설을 채택하며 대립가설이 채택된 경우에 대하여 Rosenthal의 공식을 이용하여 효과 크기를 계산하였다 [8].

그림 7은 NEP 값에 대하여, 그림 8은 NMEOP 값에 대하여 각각의 기법을 다른 기법에 대하여 통계적 검정을 수행한 결과의 p-value를 나타낸 것이다. 음영 처리된 구역은 p-value가 0.05보다 작아 왼쪽 기법의 중위 값이 오른쪽 기법의 중위 값보다 유의미하게 작은 경우를 의미한다. 연 노란색의 음영은 0.1 이상 0.3 이하의 작은 효과 크기를 의미하며 짙은 주황색의 음영은 0.5 이상의 큰 효과 크기를 의미한다.

NEP를 기준으로 이용한 검정 결과 BEST를 제외하고 다른 모든 기법을 능가하는 기법은 없는 것으로 확인되었다. BEST는 모든 기법을 큰 효과 크기를 가지고 능가하는 것으로 확인되어 BEST와 비슷한 성능을 보이는 기법이 없는 것으로 나타났다. BEST와

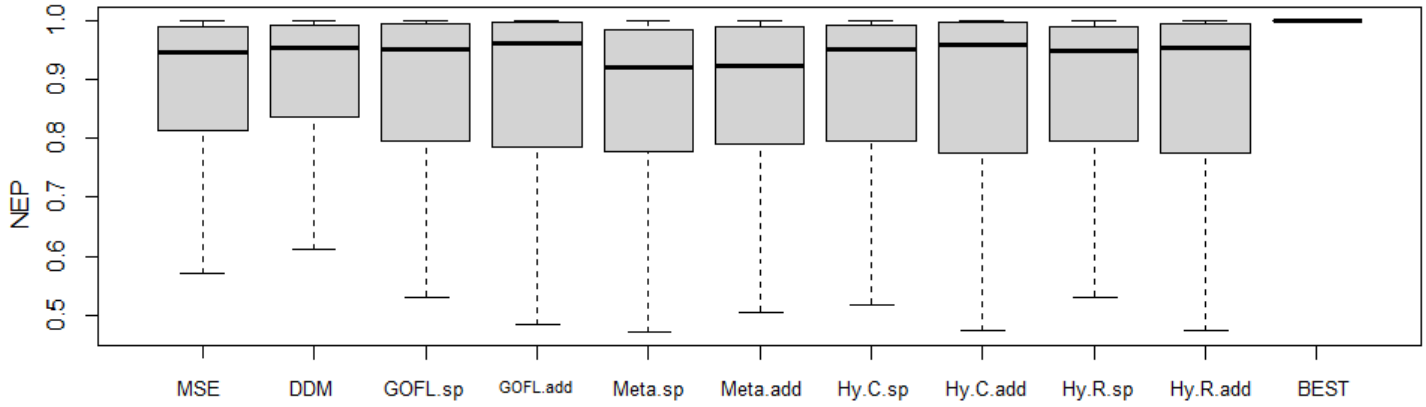


그림 6 비교 대상 기법 별 222개 검증 태스크에 대한 NEP 값의 상자 그림

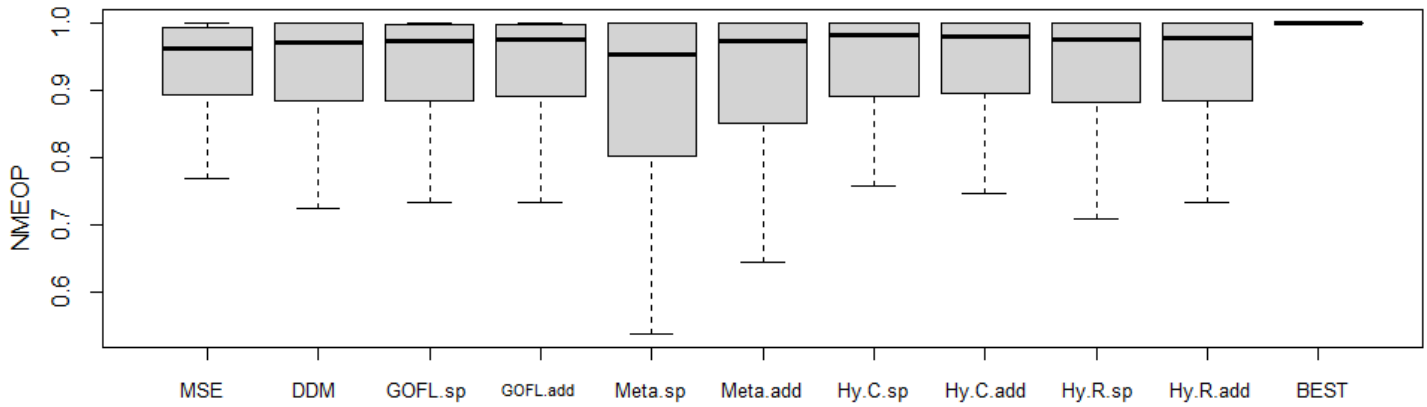


그림 5 비교 대상 기법 별 222개 검증 태스크에 대한 NMEOP 값의 상자 그림

비슷한 성능 보이는 기법이 없다는 것은 신뢰도 모델 선택 기법의 성능을 향상 시킬만한 여지가 충분히 있다는 것을 의미한다.

세개의 기법 (DDM, GOFL.sp, 그리고 Hy.R.sp)이 Meta.sp를 통계적으로 유의미하게 능가하는 것으로 확인되어 가장 많은 수의 기법들을 능가하는 것으로 식별되었으나 BEST를 제외한 10개 기법 중 9개의 기법이 BEST를 제외한 다른 어떤 기법 에게도 지지 않았기 때문에 이들 9개 기법 간에 예측 성능의 차이가 유의미하지 않다고 볼 수 있다. 따라서 9개 기법 모두 예측 성능이 가장 좋은 기법 그룹으로 분류하여 RQ2에서 차이가 있는지 비교할 것이다.

NEP를 예측 성능 기준으로 사용할 때 학습 데이터의 양을 증가시키는 것은 메타-지식 기반 학습을 통한 모델 선택 기법의 성능을 향상시키는 효과가 있는 것으로 나타났다. 세 개의 기법이 검증 데이터와 동일한 파티션의 학습 데이터만을 학습에 사용한 Meta.sp를 능가하는 반면에 어떤 기법도 학습 데이터의 양을 증가시킨 Meta.add를 능가하지 못했다. 다만 Meta.add는 Meta.sp를 능가하지 못했다.

NEP를 예측 성능 평가 기준으로 사용할 때 단일 기준을 사용한 MSE와 모델 선택을 적용하지 않은 단일 모델인 DDM이 다른 기법에 비해 나쁘지 않은 성능을 보여주는 것을 확인할 수 있으므로 학습 기반 모델 선택 기법의 효과가 낮은 것을 확인할 수 있다.

NMEOP를 예측 성능 평가 기준으로 사용한 결과 BEST를 제외하고 다른 모든 기법을 능가하는 기법은 없는 것으로 확인되었다. BEST는 모든 기법을 큰 효과 크기를 가지고 능가하는 것으로 확인되어 BEST와 비슷한 성능을 보이는 기법이 없는 것으로 나타났다.

Hy.C.sp와 Hy.C.add 두 기법이 각각 다섯 개의 다른 기법을 낮은 효과 크기를 가지고 능가하는 것으로 나타나 가장 많은 기법을 능가하는 기법들로 확인되었다. 특히 기존의 학습을 통한 네 가지 기법 (GOFL.sp, GOFL.add, Meta.sp, 그리고 Meta.add)중에서 Hy.C.sp는 GOFL.add를 제외한 세 가지 기법을, Hy.C.add는 네 가지 기법 모두를 능가하는 것으로 확인되어 분류 기반 하이브리드 학습을 통한 모델 선택 기법이 예측 성능을 향상시키는 효과가 있는 것으로 확인되었다. 그러나 DDM과 Hy.R.add 기법은 Hy.C.sp와 Hy.C.add와 마찬가지로 다른 어떤 기법 에게도 지지 않았으므로 DDM, Hy.C.sp, Hy.C.add, 그리고 Hy.R.add의 네 기법을 예측 성능이 가장 좋은 그룹으로 분류하여 RQ2에서 비교한다.

NMEOP를 예측 성능 평가 기준으로 사용할 때 학습 데이터의 양을 증가시키는 것은 예측 성능을 향상시키는데 도움이 되는 것으로 나타났다. 먼저 기존의 메타-지식 기반 학습을 통한 모델 선택

NEP p-val	MSE	DDM	GOFL.sp	GOFL.add	Meta.sp	Meta.add	Hy.C.sp	Hy.C.add	Hy.R.sp	Hy.R.add	BEST
MSE		0.75	0.88	0.74	0.11	0.23	0.84	0.76	0.91	0.64	1.00
DDM	0.25		0.21	0.31	0.02	0.06	0.19	0.28	0.15	0.16	1.00
GOFL.sp	0.12	0.79		0.52	0.03	0.08	0.49	0.43	0.33	0.24	1.00
GOFL.add	0.26	0.69	0.49		0.07	0.08	0.37	0.41	0.46	0.17	1.00
Meta.sp	0.89	0.98	0.97	0.93		0.51	0.94	0.93	0.98	0.82	1.00
Meta.add	0.77	0.94	0.92	0.92	0.49		0.88	0.90	0.89	0.79	1.00
Hy.C.sp	0.16	0.81	0.52	0.63	0.06	0.12		0.52	0.36	0.30	1.00
Hy.C.add	0.24	0.72	0.57	0.59	0.07	0.10	0.48		0.54	0.22	1.00
Hy.R.sp	0.09	0.86	0.67	0.54	0.02	0.11	0.64	0.46		0.26	1.00
Hy.R.add	0.36	0.84	0.76	0.83	0.19	0.21	0.70	0.78	0.74		1.00
BEST	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

그림 8 모든 기법 쌍에 대하여 왼쪽 (행 방향) 기법들이 오른쪽 (열 방향) 기법을 NEP 측면에서 통계적으로 능가하는지 확인한 검정의 p-value

NMEOP p-val	MSE	DDM	GOFL.sp	GOFL.add	Meta.sp	Meta.add	Hy.C.sp	Hy.C.add	Hy.R.sp	Hy.R.add	BEST
MSE		0.95	0.99	1.00	0.10	0.81	1.00	1.00	0.99	1.00	1.00
DDM	0.05		0.27	0.40	0.00	0.08	0.67	0.62	0.29	0.52	1.00
GOFL.sp	0.01	0.73		0.77	0.00	0.14	1.00	0.96	0.50	0.87	1.00
GOFL.add	0.00	0.61	0.23		0.00	0.04	0.87	0.97	0.26	0.91	1.00
Meta.sp	0.90	1.00	1.00	1.00		0.99	1.00	1.00	1.00	1.00	1.00
Meta.add	0.19	0.92	0.86	0.96	0.01		0.97	0.99	0.83	0.97	1.00
Hy.C.sp	0.00	0.33	0.00	0.13	0.00	0.03		0.46	0.04	0.26	1.00
Hy.C.add	0.00	0.38	0.04	0.04	0.00	0.01	0.54		0.07	0.35	1.00
Hy.R.sp	0.01	0.71	0.50	0.74	0.00	0.17	0.96	0.93		0.85	1.00
Hy.R.add	0.00	0.48	0.13	0.09	0.00	0.03	0.74	0.65	0.15		1.00
BEST	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

그림 7 모든 기법 쌍에 대하여 왼쪽 (행 방향) 기법들이 오른쪽 (열 방향) 기법을 NMEOP 측면에서 통계적으로 능가하는지 확인한 검정의 p-value

기법에서 학습 데이터의 양을 증가시킨 Meta.add는 동일한 파티션의 학습 데이터만을 사용한 Meta.sp를 능가하는 것으로 나타났다. 선택 대상인 모델이 13개나 되기 때문에 Meta.sp가 사용한 36개의 학습 데이터만으로는 13개 모델을 분류하기에는 분류 도구의 성능이 낮은 것으로 판단된다. 이외에도 GOFL.add는 GOFL.sp보다 하나 더 많은 기법을 능가하였고 Hy.R.add또한 Hy.R.sp보다 하나 더 많은 기법을 능가하는 것을 확인하였다.

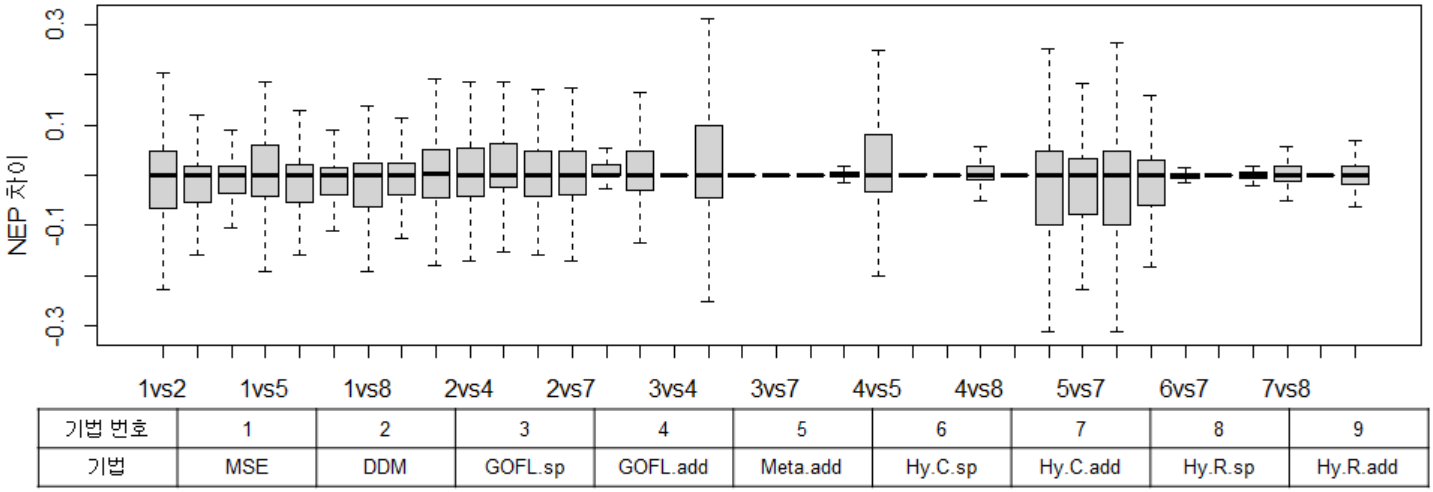
NMEOP를 예측 성능 평가 기준으로 사용할 때 모델 선택을 적용하지 않은 단일 모델인 DDM은 다른 모든 기법에 대하여 지지 않는 것을 확인할 수 있었다. 그러나 BEST가 DDM를 높은 효과 크기로 능가하는 것으로 보아 분석적 모델 중에서 예측 성능이 좋은 모델을 선택하는 정확도를 향상시킬 필요가 있다.

4.2. RQ2. 예측 성능이 가장 좋은 그룹의 기법들 간에 차이가 존재하는가?

RQ1의 실험 결과 NEP를 예측 성능 평가 기준으로 사용할 때는 9개의 기법이 예측 성능이 가장 좋은 그룹에 속한 것으로, NMEOP를 예측 성능 평가 기준으로 사용할 때는 4개의 기법이 예측 성능이 가장 좋은 그룹에 속한 것으로 확인되었다.

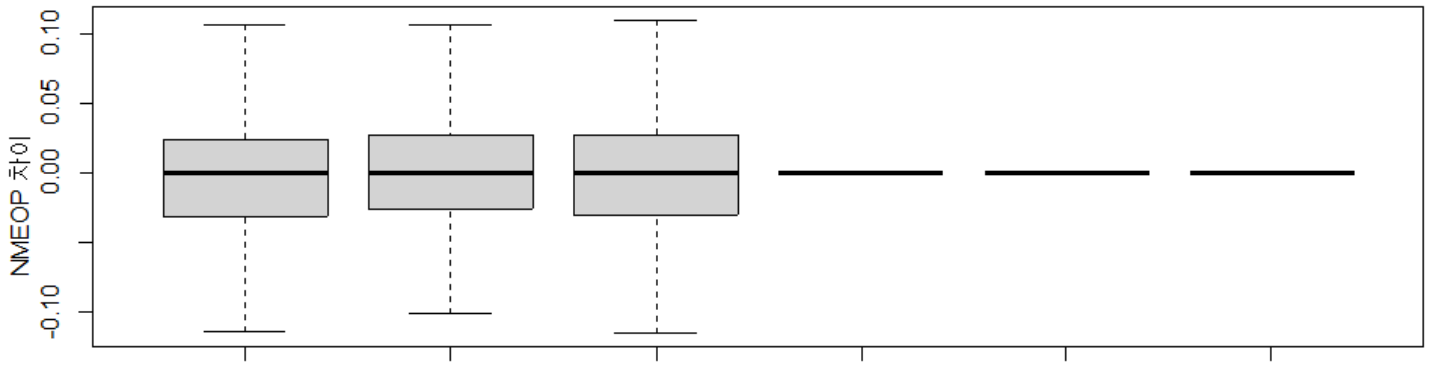
그러나 중위 값 기준의 성능에서 차이가 없다고 하더라도 개별 실패 데이터에 대한 성능에서는 차이가 있을 수 있다. 본 실험에서는 예측 성능이 가장 좋은 그룹의 기법들 간에 실패 데이터 별 예측 성능의 차이를 분산을 이용하여 확인하고자 한다.

그림 10와 그림 9은 두 기법의 실패 데이터별 예측 성능의 차이를 상자그림으로 나타낸 것이다. NEP와 NMEOP 각각에 대하여 모든 최고 성능 그룹 기법 쌍의 예측 성능의 차이를 계산하였다. 예를 들어 그림 10의 1vs2 항목은 NEP를 예측 성능 평가 기준으로 사용했을 때 최고 성능 그룹에 속한 MSE와 DDM의 222개 실패 데이터 별 NEP 값의 차이를 상자 그림으로 나타낸 것이다. 최고 성능 그룹에 속한 기법들 사이에는 중위 값의 차이가 없는 것으로 나타났기 때문에 모든



기법 번호	1	2	3	4	5	6	7	8	9
기법	MSE	DDM	GOFL.sp	GOFL.add	Meta.add	Hy.C.sp	Hy.C.add	Hy.R.sp	Hy.R.add

그림 10 NEP 기준의 최고 성능 그룹에 속한 아홉 개 기법들의 쌍에 대하여 222개 검증 태스크에 대한 기법 간 NEP 차이를 나타낸 상자 그림



기법 번호	1	2	3	4
기법	DDM	Hy.C.sp	Hy.C.add	Hy.R.add

그림 9 NMEOP 기준의 최고 성능 그룹에 속한 네 개 기법들의 쌍에 대하여 222개 검증 태스크에 대한 기법 간 NMEOP 차이를 나타낸 상자 그림

비교에서 중위 값이 0에 근접하여 나타나는 것을 확인할 수 있다.

상자 그림 분석 결과에 대해서는 다음과 같은 방식으로 해석 가능하다. 먼저 NEP 차이의 분산이 가장 큰 경우는 5vs6으로 Meta.add와 Hy.C.sp를 비교한 경우이다. 두 기법의 NEP 차이의 25% 분위 값은 마이너스 0.10으로 4분의 1개의 실패 데이터에서 Meta.add를 이용하여 모델을 선택했을 때 Hy.C.sp를 이용하여 모델을 선택했을 때 보다 NEP값이 0.10이상 작은 예측 성능을 보인다는 것을 의미한다. 두 기법의 NEP 차이의 75% 분위값은 0.05로 4분의 1개의 실패 데이터에서 Meta.add를 이용하여 모델을 선택했을 때 Hy.C.sp를 이용하여 모델을 선택했을 때 보다 NEP값이 0.05이상 큰 예측 성능을 보인다는 것을 의미한다.

NMEOP 차이의 분산이 가장 큰 경우는 1vs4로 DDM과 Hy.R.add를 비교한 경우이다. 이 경우의 NEP 차이의 25% 분위 값은 마이너스 0.03으로 4분의 1개의 실패 데이터에서 DDM을 이용하여 모델을 선택했을 때 Hy.R.add를 이용하여 모델을 선택했을 때 보다 NMEOP값이 0.03이상 작은 예측 성능을 보인다는 것을 의미한다. 두 기법의 NMEOP 차이의 75% 분위값은 0.03으로 4분의 1개의 실패 데이터에서 DDM을 이용하여 모델을 선택했을 때 Hy.R.add를 이용하여 모델을 선택했을 때 보다 NMEOP값이 0.03이상 큰 예측 성능을 보인다는 것을 의미한다.

5. 결론 및 향후 연구

다양한 종류의 소프트웨어 신뢰도 예측 모델들이 개발되어 사용되고 있으나 주어진 실패 데이터에 대하여 좋은 예측 성능을 보이는 모델을 선택하는 것은 어려운 일이다. 이러한 문제를 해결하기 위해 적합도 기반의 모델 선택 기법과 메타-지식 기반의 모델 선택 기법이 제안되었다. 그러나 적합도 기반 학습을 통한

모델 선택 기법은 적합도를 계산할 수 없는 비 분석적 모델의 선택이 불가능한 문제를 가지고 있다. 메타-지식 기반의 학습은 비 분석적 모델을 포함하여 선택이 가능하지만 메타-지식은 적합도에 비해서 모델의 예측 성능에 대한 관련성이 작기 때문에 상대적으로 모델 선택의 정확도가 낮은 문제가 있다.

본 연구에서는 학습을 통한 신뢰도 예측 모델 선택 기법들의 성능을 향상시키기 위한 방안으로 적합도와 메타-지식을 모두 사용한 하이브리드 학습을 통한 모델 선택 기법을 제안한다. 하이브리드 학습은 메타-지식을 이용하여 비 분석적 모델이 대상 실패 데이터에서 좋은 성능을 보이는지 그렇지 않은지 먼저 학습한 후에 비 분석적 모델의 성능이 좋은 경우에만 비 분석적 모델을 선택하고 그렇지 않은 경우에는 적합도 기반의 학습을 통해 모델을 선택한다.

실험 결과 NMEOP를 예측 성능 기준으로 사용하는 경우에 하이브리드 학습을 통한 모델 선택 기법들이 기존의 학습을 통한 모델 선택 기법들을 능가하는 것을 확인하였다. 향후 연구에서는 학습을 위한 데이터를 더 많이 확보하기 위해서 기존의 실패 데이터들을 다양한 파티션으로 나누어 예측 성능을 높이고자한다.

감사의 말

본 연구는 LIG 넥스원의 지원, 국방과학연구소의 지원, 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 지원 (IITP-2020-2020-0-01795), 정부 (과학기술정보통신부) 의 재원으로 한국연구재단의 지원 (NRF-2019R1G1A1005047)을 받아 수행된 연구임

참고문헌

[1] Febrero, Felipe, Coral Calero, and Mª Ángeles Moraga. "A systematic mapping study of software reliability modeling." Information and Software Technology 56.8 (2014): 839-849.

[2] Kiran, N. Raj, and Vadlamani Ravi. "Software reliability prediction by soft computing techniques." Journal of Systems and Software 81.4 (2008): 576-583.

[3] Xiao, Xiao, and Tadashi Dohi. "Wavelet shrinkage estimation for non-homogeneous Poisson process based software reliability models." IEEE Transactions on Reliability 62.1 (2013): 211-225.

[4] Park, Jinhee, and Jongmoon Baik. "Improving software reliability prediction through multi-criteria based dynamic model selection and combination." Journal of Systems and Software 101 (2015): 236-244.

[5] Wu, Yumei, and Risheng Yang. "Software reliability modeling based on SVM and virtual sample." 2013 Proceedings Annual Reliability and Maintainability Symposium (RAMS). IEEE, 2013.

[6] Caiuta, Rafael, Aurora Pozo, and Silvia Regina Vergilio. "Meta-learning based selection of software reliability models." Automated Software Engineering 24.3 (2017): 575-602.

[7] Park, Jinhee, Nakwon Lee, and Jongmoon Baik. "On the long-term predictive capability of data-driven software reliability model: an empirical evaluation." 2014 IEEE 25th International Symposium on Software Reliability Engineering. IEEE, 2014.

[8] Parametric measures of effect size. In H. Cooper & L. V. Hedges (Eds.), The handbook of research synthesis. (pp. 231-244). New York: Russell Sage Foundation.

[9] J. Musa, A. Iannino, and K. Okumoto, Software Reliability: Measurement, Prediction, Application. 1987. New York, 1987.

[10] M. R. Lyu, Ed., Handbook of software reliability engineering. Los Alamitos, CA: IEEE Computer Society Press, 1996.

[11] A. Wood, “Predicting software reliability,” Computer (Long. Beach. Calif)., vol. 29, no. 11, pp. 69– 77, 1996.

[12] D. Jeske, X. Zhang, and L. Pham, “Adjusting software failure rates that are estimated from test data,” Reliab. IEEE Trans., vol. 54, no. 1, pp. 107–114, 2005.

[13] C. Stringfellow and A. Andrews, “An empirical method for selecting software reliability growth models,” Empir. Softw. Eng., vol. 7, no. 4, pp. 319–343, 2002.

[14] Y. Tohma and R. Jacoby, “Hyper-geometric distribution model to estimate the number of residual software faults,” in Proceedings of the 13th Annual International Computer Software and Applications Conference, 1989, pp. 610–617.

[15] J.-H. Lo and C.-Y. Huang, “An integration of fault detection and correction processes in software reliability analysis,” J. Syst. Softw., vol. 79, no. 9, pp. 1312–1323, 2006.

[16] K. Sharma, R. Garg, C. K. Nagpal, and R. K. Garg, “Selection of Optimal Software Reliability Growth Models Using a Distance Based Approach,” IEEE Trans. Reliab., vol. 59, no. 2, pp. 266–276, 2010.

[17] X. Li, Y. F. Li, M. Xie, and S. H. Ng, “Reliability analysis and optimal version–updating for open source software,” Inf. Softw. Technol., vol. 53, no. 9, pp. 929–936, Sep. 2011.

코드 블록과 장단기 메모리를 활용한 프로그램 버그 정정 기법

호혜민○, 양근석, 이병정
서울시립대학교 컴퓨터과학과
{hyemin, ypats87, bjlee}@uos.ac.kr

A Novel Technique of Program Bug Repair by using Code Block and LSTM Algorithm

Hu Huimin○, Geunseok Yang, Byungjeong Lee
Department of Computer Science, University of Seoul

요 약

최근 지능형 소프트웨어 분야에서 머신러닝을 활용한 프로그램 버그 정정 연구가 활발하게 진행되고 있다. 프로그램 규모와 복잡성 증가로 버그가 많이 발생하였으며, 이를 해결하기 위해 다양한 방법이 제안되고 있다. 본 논문에서는 코드 블록과 장단기 메모리를 활용하여 프로그램 버그 정정 기법을 제안한다. 먼저 프로그램 소스코드를 전처리를 진행하고 제안한 코드 블록 기법으로 처리한다. 처리된 코드 블록 소스코드를 장단기 메모리에 학습을 진행한다. 새로운 버기 라인을 학습된 모델에 입력하면 새로운 소스코드 라인을 생성한다. 생성한 후보 패치의 적합성을 평가하기 위해 적합도 함수를 적용하며, 주어진 테스트 케이스를 모두 통과하면 적합한 패치라고 본다. 제안한 모델을 평가 하기 위해 베이스라인과 성능 비교를 하였으며, 제안한 방법이 버그를 더 정정한 것을 보인다.

1. 서 론

프로그램의 규모와 복잡성 증가에 따라 프로그램 버그가 발생하게 되었다. 최근 지능형 소프트웨어 분야에서 머신러닝을 활용하여 프로그램 버그 정정 연구를 활발하게 진행하고 있다. 만약 자동화된 프로그램 버그 정정이 진행된다면, 개발자들의 개발 생산성을 향상할 수 있고, 소프트웨어 품질을 향상시킬 수 있다.

오픈 소스 프로젝트에서는 버그가 발생하면, 버그 리포트가 제출이 된다. 개발자는 버그 리포트를 읽으며, 프로그램 디버깅 과정을 진행하고 프로그램 버그 정정을 진행한다. 품질 보증자가 정정된 프로그램을 테스트 케이스 등의 방법을 통해 검증을 진행하고, 검증이 완료되면 최종 배포를 진행한다.

프로그램 버그 정정과 관련된 연구는 다음과 같다. DeepFix[1]는 Attention 알고리즘 기반 컴파일 에러를 위한 버그 정정 기법이다. 하지만 컴파일 에러뿐만 아니라 다양한 프로그램 실행 에러를 정정할 수 있는 방법이 필요하다. GenProg[2]는 유전 프로그래밍 (Genetic Programming, GP)을 사용한 C언어 기반 프로그램 정정 기술이다. 이 기법은 휴리스틱 방법으로 개체 프로그램 AST (Abstract Syntax Tree)에 대해 유전 연산을 진행한다. 생성된 프로그램의 적합성을 평가하기 위해 적합도 함수를 이용하였다. 이후 Java 언어를 처리할 수 있도록 jGenProg[3]라는 기술도 연구 되었다. 하지만 아직까지 다양한 프로젝트에 대한 검증과 프로그램 버그 정정 성능이 개선 될 필요가 있다.

이러한 문제를 해결하기 위해 본 논문에서는 코드 블록과 장단기 메모리(LSTM)를 활용하여 프로그램 버그 정정 기법을 제안한다. 먼저 프로그램 소스코드에 대해 전처리를 진행한다. 그리고 제안한 코드 블록 기법을 적용하여 코드 블록 기반 소스코드를 생성한다. 장단기 메모리의 입력으로 코드 블록 기반 소스코드를 넣고 모델을 학습한다. 모델을 학습할 때 입력은 코드 블록으로 처리된 소스코드 이며 전처리 과정에서 소스코드를 토큰으로 표현한다. 새로운 버기 라인을 해당 모델에 입력하면 새로운 프로그램 후보 패치를 생성한다. 생성한 후보 패치의 적합성을 평가하기 위해 적합도 함수를 적용한다. 주어진 테스트 케이스를 모두 통과하면 적합한 패치라고 본다.

본 논문이 기여하는 부분은 다음과 같다.

- 소스코드에 코드 블록 기법을 적용하여 새로운 코드 블록 기반 소스코드를 생성한다. 생성한 소스코드를 장단기 메모리에 학습하여 프로그램 버그 정정을 진행하였다.
- 모델의 성능을 향상하기 위해 프로그램 소스코드

전처리를 진행하였다.

- 제안한 방법의 효율성을 평가하기 위해 베이스라인과 성능 비교를 하였으며, 제안한 방법이 프로그램 버그를 더 정정한 것을 보였다.

본 논문의 구성은 다음과 같다. 2장에서 배경지식을 소개한다. 3장에서 프로그램 버그 정정 방법을 제안하고, 4장에서 실험을 진행한다. 5장에서 실험 결과에 대한 토의를 진행하고, 6장에서 관련 연구를 소개한다. 7장에서 결론과 향후 연구를 소개한다.

2. 배경 지식

2.1 프로그램 버그 자동 정정

프로그램 버그 자동 정정의 설명을 위해 다음 그림 1과 같이 기술한다. 그림 1은 IntroClass[4]의 Checksum 버기 데이터 샘플이며, 라인 11번 "character % 64"에서 적절하지 않은 변수를 사용하여 프로그램 버그가 발생하였다.

```
1   …
2     int sum = 0;
3     while (character!= '₩n') {
4       try {
5         character = scanner.findInLine (".").charAt (0);
6       } catch (java.lang.NullPointerException e) {
7           character = '₩n';
8         };
9       sum = sum + (int) character;
10    }
11    remainder = (char) (( character   % 64) + 22);
12    output += (String.format ("Check sum is %c₩n",
13  remainder));
14  …
```

그림 1 프로그램 버그 정정 예시

주어진 테스트 케이스가 다음과 같이 존재하며, 테스트 케이스 1번을 입력("O Brother Where Art Thou?") 하면 "Check sum is F"가 출력되어야 한다. 하지만 그림 1의 프로그램은 적절하지 않은 결과("Check sum is ")가 출력된다.

```
#테스트 케이스 1
입력: "O Brother Where Art Thou?"
출력: "Check sum is F"

#테스트 케이스 2
입력: "100 Degrees and sunny"
출력: "Check sum is @"
```

제안한 모델에 의해 다음과 같이 프로그램 패치를 생성하여, 프로그램 버그 자동 정정을 시도한다. 프로그램 후보 패치 1과 2를 적용하면 "Check sum is ㅁ"의 결과를 보이고, 주어진 테스트 케이스를 통과할 수 없다. 프로그램 후보 패치 3을 적용하면 주어진 테스트 케이스를 다 통과하여 적절한 패치라고 본다. 따라서 이 프로그램 버그 정정은 프로그램 후보 패치 3으로 생성한다.

```
#프로그램 패치1
라인 9 → sum = sum + (int) character; (제거)

#프로그램 패치2
라인 9 → character = (char) (sum + (int) character);
(변형)

#프로그램 패치3
라인 11 → remainder = (char) ((sum % 64) + 22); (변형)
```

2.2 코드 블록 (Code Block)

코드 블록은 소스코드로부터 처리된 코드 표현 과정이다. 먼저 소스코드에 대해 멤버 변수와 메서드를 식별한다. 그리고 각 메서드와 멤버 변수를 결합하여 코드 블록을 구성한다. 전체 소스코드로부터 코드 블록으로 처리한 예시는 그림 2와 같다. 그림 2에서 원본 코드는 목표 디렉터리 중에 있는 모든 Java 파일을 설정된 경로로 복사하는 기능을 갖고 있으며 예시로 써 메서드 내용과 일부의 코드를 생략하였다. 멤버 변수는 라인 5, 6이고 메서드는 총 3가지 있으며 라인 7-9번 main함수, 라인 10-12번 Java 파일인지를 판단 함수와 라인 13-15 파일 복사 함수이다.

```
1   package My_project;
2   import java.io.File;
3   …
4   public class Copy {
5     private static final String fromPath = "D: /Dataset";
6     private static final String toPath = "D:/Dataset_C";
7     public static void main(String[] args) {
8       …
9     }
10    private static void filter(File fromFile) {
11        …
12    }
13    private static void copyFile(File fromFile, File toFile){
14      …
15    }
16  }
```

그림 2 전체 소스코드로부터 코드 블록으로 처리

본 논문에서의 모델 학습 과정에서는 Package와 Import 멤버 변수로 써 코드 블록에 결합된다. 그림 2에서 각 메서드가 멤버 변수와 결합하여 3가지의 코드 블록이 구성할 수 있으며 하나의 예시는 그림 3과 같다. 라인 수는 코드 블록에서 새로운 번호로 적용되며 원래 소스코드와 다르다.

```
1  package My_project;
2  import java.io.File;
3  …
4  private static final String fromPath = "D: /Dataset";
5  private static final String toPath = "D:/Dataset_C";
6  private static void copyFile(File fromFile, File toFile) {
7     …
8  }
```

그림 3 코드 블록 예시

2.3 장단기 메모리

인코더 (Encoder)는 순환 신경망으로 입력을 처리하기 위해 LSTM를 이용한다. 이는 입력 데이터 전후 토큰의 정보를 통합할 수 있도록 하는 양방향 인코더이다. 디코더 (Decoder)도 장단기 메모리를 사용하는 순환 신경망이다. 인코더에 의해 초기화 되면 시작 토큰을 입력으로 받아 후보 패치를 생성하기 시작한다. 장단기 메모리는 주로 긴 시퀀스의 학습 과정에서 기울기 소멸 문제를 해결하기 위한 특수 RNN (Recurrent Neural Network)이다.

코드 블록과 전제 소스코드를 LSTM 모델에 적용하는 과정을 그림 4와 같다. W, X, Y, Z는 모델의 출력이고 입력은 소스 코드이다. 출력에는 새로운 소스코드 라인이 포함된다.

본 논문에서는 전체 소스코드에 대해 코드 블록 기법을 적용하여 새로운 코드 블록 기반 소스코드를 생성한다. 생성한 코드 블록 소스코드를 장단기 메모리에 학습하여, 프로그램 버그 정정을 진행한다. 버기 라인을 학습된 장단기 메모리 모델에 입력하면, 새로운 프로그램 패치를 생성한다.

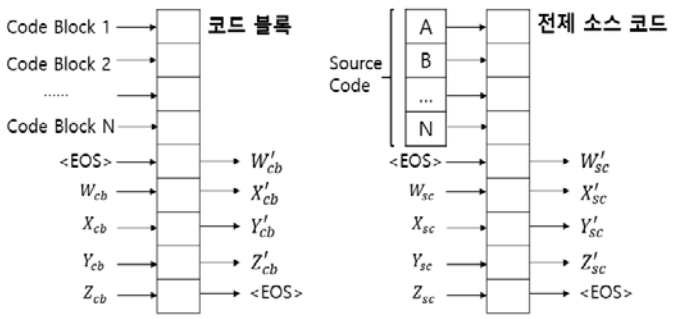


그림 4 LSTM 모델 예시

3. 프로그램 버그 자동 정정

3.1 전체 도식도

제안한 프로그램 버그 정정에 대한 전체 도식도는 그림 5와 같다.

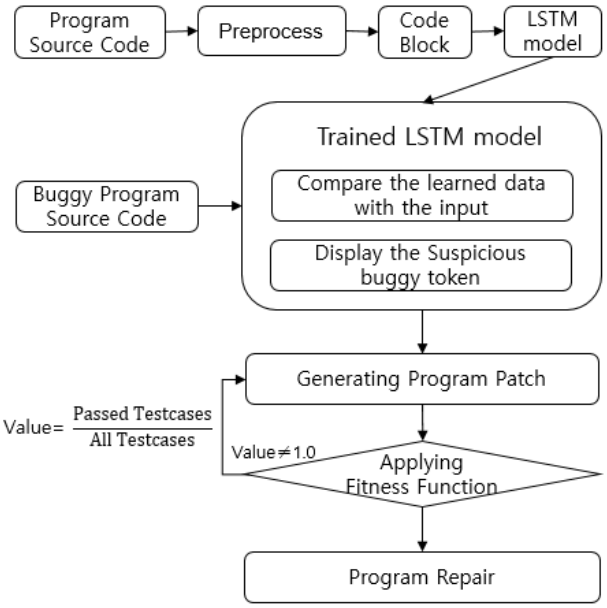


그림 5 프로그램 버그 정정 과정 도식도

먼저 소스코드에 대해 전처리를 진행한다. 전처리된 소스코드에 대해 제안한 코드 블록 기법을 적용하여 새로운 코드 블록 기반 소스코드를 생성한다. 코드 블록 기반 소스코드를 장단기 메모리 모델을 이용하여 학습한다. 버기 소스코드를 학습된 모델에 입력하면 새로운 프로그램 후보 패치 생성을 진행한다. 생성된 프로그램 후보 패치가 적합한지를 판단하기 위해 적합도 함수를 적용하고 패치의 적합성을 평가한다. 적합도 함수에서 value 값으로 적합도를 표시한다. 값이 1에 가까울 수록 해당 후보 패치의 적합성이 높다고 볼 수 있다. 주어진 테스트 케이스를 모두 통과하면 Value의 값은 1이 된다. 모든 테스트 케이스를 통과 못하는 경우 다시 패치 생성을 진행한다. 모든 테스트 케이스를 통과하면 해당 후보 패치가 적합하다고 본다. 이 과정에서 프로그램 버그 정정은 적합한 패치로 진행한다.

3.2 소스코드 전처리

장단기 메모리 모델의 성능을 향상하기 위해 소스코드에 적절한 규칙 정보를 추가하여 모델을 학습시킨다. 이를 진행하기 위해 소스코드 전처리 과정을 진행한다. 예를 들어, “*private static final double DEFAULT_EPSILON = 10e−9;*” 인 경우 “*<START> private static final double DEFAULT_EPSILON = 10e−9;*

<END>"로 표현된다. 버기 라인을 강조하기 위해 버기 라인 앞, 뒤에 표시 토큰을 추가한다. 이런 방식을 적용하면 버그를 찾기가 편해지고 시간 비용이 절약된다.

3.3 코드 블록 기반 모델 학습

전체 소스코드로부터 제안한 코드 블록 기법을 적용하여 새로운 코드 블록 단위의 소스코드를 생성한다. 본 논문에서는 그림 6과 같이 코드 블록 변환과정을 진행한다.

그림 6에서 Frequency.java를 예시로, 먼저 파일에 포함하는 멤버 변수와 메서드를 식별한다. 식별한 각 메서드와 멤버 변수를 결합해서 코드 블록을 구성하면서, 하나의 메서드가 하나의 코드 블록으로 구성한다. 즉, 코드 블록 개수는 메서드 개수와 동일하다. 멤버 변수의 경우 모든 코드 블록에 포함되어야 한다. 생성된 코드 블록 기반 소스코드를 장단기 메모리 모델에 학습을 진행한다.

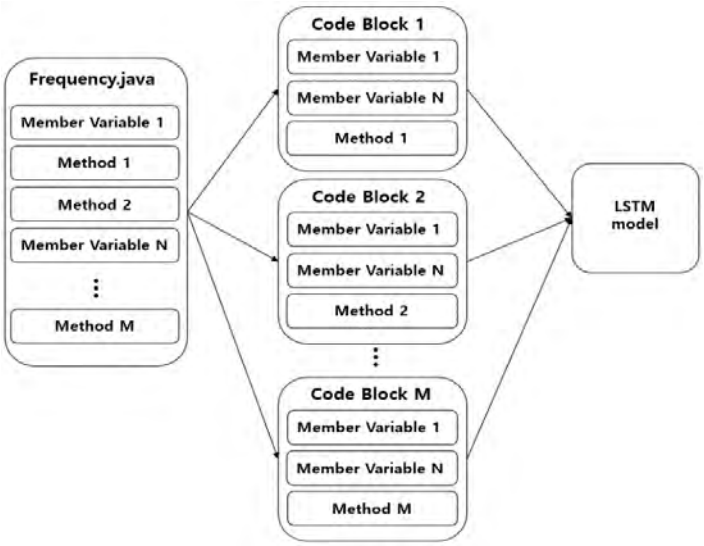


그림 6 코드 블록 처리 과정

3.4 검증

생성한 후보 패치가 적합한지를 평가하기 위해 본 논문에서는 적합도 함수를 이용한다. 적합도를 계산하는 수식은 다음과 같다.

$$Value = \frac{|Passed\ \ Testcases|}{|All\ \ Testcases|}$$

- Passed Testcases는 해당 패치가 통과한 테스트 케이스 개수를 의미한다.
- All Testcases는 테스트 케이스 총 개수를 의미한다.

따라서 통과한 테스트 케이스 개수가 많아질수록 Value의 값은 1에 근접한다. 테스트 케이스를 모두 통과한 경우 Value의 값은 1으로 계산된다.

4. 실험

4.1 데이터 집합

본 논문에서 사용한 데이터 집합은 CodRep[5] 와 Defects4J[6]이다. CodRep은 5 개 부분으로 구분된다. 사용된 데이터 집합은 다음 표 1과 같다.

표 1. 사용된 데이터 집합

Dataset	Project	Bugs
CodeRep	-	4,711
Defects4J	Chart	9
	Closure	21
	Lang	10
	Math	23
	Mockito	7
	Time	3

Defects4J에서 6개 프로젝트 (Chart, Closure, Lang, Math, Mockito, Time)를 사용하고, 사용된 모든 프로젝트에서 단일 라인 버그 만을 사용한다. 단일 라인 버그의 총 개수는 73개 이다. CodRep 데이터 집합에는 총 4,711개의 버그가 있다.

4.2 베이스라인

본 논문에서의 베이스라인은 다음과 같다.

- SequenceR[7]은 딥러닝 학습 기반으로 오픈 소스 커밋 정보를 활용하여 프로그램 버그 정정을 진행한다.

베이스라인의 선정 기준으로는 온라인에 소스코드가 공개가 되고, 재현이 가능한 연구 기법을 선정한다.

4.3 실험 환경

실험에서 사용한 CPU는 Intel® Xeon® processor E5-1650 v3 3.50GHz 12Cores 이다. 모델 학습 횟수는 20,000번으로 정하였으며, 베이스라인과 동일하게 적용한다.

4.4 연구 질문

- 제안한 프로그램 버그 정정 모델의 성능은?
- 제안한 모델이 프로그램 버그 정정에서 베이스라인 보다 더 좋은 성능을 보일 수 있는가?

4.5 결과

4.5.1 코드 블록과 모델의 성능

제안한 코드 블록 기반 프로그램 버그 정정 모델의 성능은 그림 7과 같다. 그림 7에서 X축은 모델 학습 횟수를 의미하고, Y축은 정확도를 의미한다. 학습 모델에는 두 가지 (CodeRepo, Defects4J) 데이터 셋을 학습하고, 베이스라인도 동일하게 진행한다. 제안한 모델이 약 85%의 정확성을 보이며, 적절하게 프로그램 버그 정정을 하는 것을 보인다.

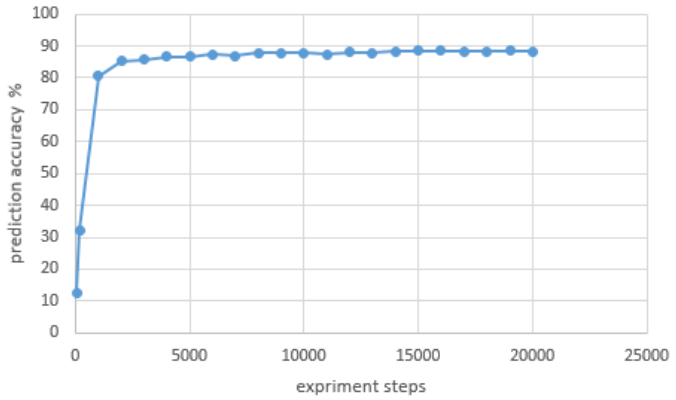


그림 7 예측 정확성

4.5.2 베이스라인과의 성능 비교

제안한 모델의 성능을 평가 하기 위해 베이스라인과 동일한 학습환경에서 프로그램 버그 정정 성능 비교를 진행하였고, 그림 8과 같다.

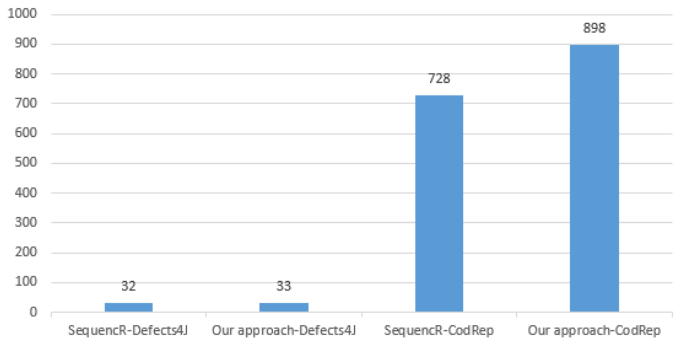


그림 8 베이스라인간 성능 비교

그림 8에서 X축은 종류를 의미하고, Y축은 정정된 개수를 의미한다. 제안한 방법은 CodRep에서 898개의 버그를 정정하였고, Defects4J에서 33개의 버그를 정정하였다. 반면, 베이스라인은 CodRep에서 728개의 버그를 정정하였고, Defects4J에서 32개의 버그를 정정하였다. 생성된 패치 중 주어진 테스트 케이스를 모두 통과한 개수를 의미하며, 적합한 패치의 개수를 의미한다.

정정된 버그 개수는 [7]의 논문 결과와 다르며, 이유는 다음과 같다.

- 본 논문에서는 제안한 방법과 베이스라인 실험에서 20,000번 학습을 진행하였고, SequenceR[7] 논문에서는 다른 학습 횟수에서 다른 결과를 보였다.
- 학습 횟수는 모델 성능에 영향을 미친다. SequenceR[7]에서는 학습 횟수가 성능에 어떤 영향을 미치는지 파악하기 위해 추가 실험을 진행하였는데, 학습 횟수가 증가하면서 성능이 떨어질 수도 있다는 결과를 보였다.

향후 본 논문에서 제안한 방법을 작용하여 적절한 학습 횟수를 파악하며 모델의 성능을 향상할 예정이다.

5. 토의

5.1 실험 결과

제안한 코드 블록 기반 프로그램 버그 정정 방법은 CodRep에서 총 4,711개의 버그 중 898개의 버그를 정정하였다. Defects4중에 총 73개의 단일 라인 버그에서 33개의 버그를 정정하였다. 전체 소스코드 기반으로 학습한 베이스라인 SequenceR 보다 코드 블록 기법을 적용한 학습 기법이 더 잘 예측 한 것을 보인다. 이는 장단기 메모리의 구조상 입력 시퀀스의 전후 순서를 고려하여, 멤버 변수와 메서드를 결합하여 더 많은 프로그램 버그 정정 정보를 주어 더 잘 정정하였다. 이에 대해 향후 자세히 연구할 예정이다.

본 논문에서 벤치마크 데이터 셋 기준으로 코드 블록을 적용하고, 향후 새로운 소스코드가 적용이 되어도 코드 블록을 적용시켜 프로그램 정정을 진행할 예정이다.

5.2 프로그램 버그 정정 성능의 정성적 비교

부록 1은 Defect4J에 대하여 제안한 모델과 베이스라인간 프로그램 버그 정정 성능의 비교를 나타낸다. 회색으로 표시된 버그 Closure-18, Lang-59, Math-70은 베이스라인으로 버그를 정정하지 못하지만, 제안한 방법으로는 버그를 정정한 경우이다. Time-16는 제안한 방법으로는 버그를 정정할 수 없고, 베이스라인은 정정한 경우를 의미한다. 생성된 패치는 다음과 같다. 제시된 모든 패치에서 '-'는 제거된 소스코드 라인을 의미하고, '+'는 추가된 소스코드 라인을 의미한다.

#Closure-18

```
-    if(outStream!=0){
+    if(!(outStream())){
```

Closure-18 버그는 조건문 내용을 변경하여 버그를 정정하였다. 하지만 베이스라인은 해당 조건문을

적절하게 변경하지 못하여 버그 정정을 실패한다.

#Lang-59

```
-    str.length(strLen,strLen=strLen=width);
+    str.length(strLen,strLen);
```

Lang-59 버그는 인접한 멤버 변수로부터 파라미터를 strLen,strLen로 변형하여 버그를 정정하였다.

#Math-70

```
-            min=true;
+            min=fmin;
```

Math-70 버그는 변수에 특정 변수의 값을 저장해야 하는데, 참/거짓으로 값을 저장하여 버그가 발생하였다. 제안한 방법에서는 인접한 멤버 변수로부터 정확하게 변수명 치환을 하였다.

#Time-16

```
-        assert String(text);
+        assert text;
```

Time-16 버그는 assert시 형변환 없이 변수명을 진행한 경우이다. 하지만 제안한 방법에서는 형변환을 제거하여 프로그램 버그를 정정하지 못하였다. 베이스라인은 적절하게 버그를 정정하였다. 이에 대해 추가적인 분석을 할 예정이다.

SequenceR에서는 전체 소스코드 기반으로 학습을 진행하고, 제안한 방법은 코드 블록 기반으로 학습을 진행한다. 향후 코드 블록 기반 학습으로 버그 정정 성능이 향상된 이슈에 대해서 추가 연구할 예정이다.

5.3 위협 요소

본 논문의 위협 요소는 다음과 같다.

- 프로그램 버그 정정에서 잘 사용되는 CodeRep, Defects4J 벤치 마크 데이터를 사용하여 프로그램 버그 정정을 진행하였다. 하지만, 제안한 모델이 다른 오픈 소스 프로젝트에 항상 잘 적용될 수 있다고 일반화 할 수 없고, 상업 프로젝트 적용과 함께 추가 연구가 필요하다.
- 생성된 후보 패치의 적합성을 판단하기 위해 적합도 함수를 사용하였다. 하지만, 모든 테스트 케이스를 통과하였다고 정확한 프로그램 버그 정정이라고 할 수는 없다.

6. 관련 연구

Prophet[8]은 개발자가 정정한 패치들 사이에서 공통 특징이 있다는 연구를 기반으로 개발자의 패치를 활용해 패치의 정확도를 예측하고 우선 순위화하는 기술을 제안하였다.

DeepRepair[9]은 머신러닝 기반으로 오토 인코더(AutoEncoder)를 활용하여 프로그램 버그 정정을 진행한 연구이다. 버기 프로그램 내 모든 메서드를 학습하여 단어 벡터를 만든다. 이를 통해 학습된 모델은 메서드간의 유사도를 계산할 수 있다. 메서드와 유사한 메서드를 순위화 시키고, 유사도가 높은 메서드 내의 코드들을 재료코드로써 활용할 것을 제안하였다. 하지만 소스코드에서 메서드가 아닌 부분도 있으며 메서드만 집중 하게 되면 다른 중요한 정보를 잃을 수 있다.

DLFix[10]는 이전 버그 정정 및 정정 정보의 주변 코드 컨텍스트를 학습한 2-계층 딥러닝 모델이다. 첫 번째 계층은 버그 정정의 컨텍스트를 학습하는 트리 기반 RNN 모델이고, 이 결과는 두 번째 계층에 대한 추가 가중치 입력으로 사용된다.

HDRepair[11]는 실제 프로그램에서 특정한 패치가 반복 사용된 경우, 관련 정보가 자주 사용된다는 것을 발견하여 이전 정정 패턴의 유효한 정보를 제공할 수 있는 인사이트를 도출하였다. 먼저 다양한 프로젝트로부터 버그 정정 패턴을 추출한다. 그리고, 기존 돌연변이 연산자를 사용하여 주어진 버그 프로그램에 대해 후보 패치를 생성한다. 이어서 생성한 후보 중에 자주 발생하는 버그 정정 이용 빈도수가 높은 것을 우선 순위화 시킨다. 그리고 모든 테스트 케이스를 통과한 후보도 가능한 패치로 권장한다.

SequenceR[7]은 Attention and Copy mechanism 기반 Sequence-to-Sequence Network를 이용하여 프로그램 버그 정정을 진행한다. 주어진 전체 소스코드에 대해 프로그램 소스코드 추상화를 진행하여 모델을 학습한다.

Hu et al.[12]은 GAN 알고리즘을 활용하여 프로그램 버그 정정 기법을 제안하였다. 먼저 정상적인 프로그램 소스코드를 활용한 코드 블록 기반 모델을 제안하였다. 하지만 본 논문은 코드 블록 기법을 적용하기 전 프로그램 소스코드 전처리를 진행한다. 그리고 복잡한 프로젝트에 적용되어 모델이 개선되어야 하고, 프로그램 버그 정정 성능 또한 개선이 되어야 한다.

CoCoNut[13]은 CNN (Convolution Neural Network)과 새로운 컨텍스트 주의를 이용한 기계 번역 (NMT) 조합하여 앙상블 학습(ensemble learning)을 진행하고 자동으로 프로그램 버그 정정하는 기법을 제안하였다. 이 기법은 버기 라인 소스코드와 주변에 나타내는 버기 컨텍스트를 별도로 표시하여 더 다양한 버그를 정정하기 위해 임의 하이퍼파라미터 튜닝으로 구성한 여러 개의 모델에 대해 앙상블 학습을 진행한다.

표 2. 관련 연구와의 정성적 비교

Approach	Technology	Feature
DeepFix[1]	딥러닝	버그 식별과 패치 생성은 동시 진행
GenProg[2]	유전 프로그래밍	개체를 프로그램의 AST(Abstract Syntax Tree)로 표현
jGenProg[3]	유전 프로그래밍	C언어부터 Java언어로 확장
SequenceR[7]	딥러닝	LSTM model with attention and copy mechanism
Prophet[8]	머신러닝	다양한 응용에서 심층적인 의미론적 코드 정확성 속성 캡처
DeepRepair[9]	딥러닝	재료코드라는 리소스 준비
DLFix[10]	딥러닝	2 계층 DL 모델
HDRepair[11]	검색 알고리즘	Stochastic Search
Hu et al.[12]	딥러닝	코드 블록
CoCoNut[13]	딥러닝	하이퍼 파라미터 튜닝의 임의성
Our approach	딥러닝	코드 블록, 소스코드 전처리

표 2에서 대부분의 연구는 딥러닝/머신러닝 알고리즘을 사용하여 프로그램 버그 정정을 진행한다. 그리고 유전 프로그래밍을 활용하여 C언어와 Java언어 기반 프로그램 버그도 정정하였다.

코드 블록 기반으로 진행된 연구는 Hu et al.[12]만 있고, 본 논문에서는 전체 소스코드에 대해 소스코드 전처리를 진행하고, 코드 블록 기법을 적용하는 차이점을 지닌다.

7. 결론

본 논문에서는 코드 블록과 장단기 메모리를 활용하여 프로그램 버그 정정 기법을 제안하였다. 자세히는 전체 소스코드에 대해 소스코드 전처리를 적용하고, 제안한 코드 블록 기법을 이용하여 새로운 코드 블록 기반 소스코드를 추출한다. 추출된 소스코드를 장단기 메모리에 학습한다. 새로운 버기 라인이 학습된 모델에 입력되면, 새로운 프로그램 후보 패치를 생성한다. 생성한 패치의 적합성을 평가 하기 위해 적합도 함수를 이용하여, 모든 테스트 케이스를 통과한 프로그램 후보 패치를 프로그램 버그 정정 패치로 사용한다. 제안한 모델의 성능을 평가 하기 위해 학습 기반 프로그램 버그 정정 관련 연구를 베이스라인으로 설정하고, 동일한 실험 환경에서 비교를 하였으며 제안한 모델이 더 잘 프로그램 버그를 정정하는 것을 보였다. 향후 다양한 검색 기반 방법을 이용하여 모델을 더욱 구체화하여 프로그램 버그 정정의 성능을 더욱 개선할 예정이다.

8. Acknowledgement

이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 한국연구재단 중견연구자지원사업의 지원을 받아 수행된 연구임(NRF-2020R1A2B5B01002467).

참고문헌

[1] R. Gupta, S. Pal, A. Kanade, and S. Shevade, “DeepFix: Fixing common C language errors by deep learning”, In Proc. of the 31st AAAI Conference on Artificial Intelligence, pp. 1345-1351, 2017.

[2] C. Le Goues, T. Nguyen, S. Forrest, and W. Weimer, “GenProg: A generic method for automatic software repair”, IEEE Transactions on Software Engineering, vol.38, no. 1, pp. 54-72, 2012.

[3] M. Martinez and M. Monperrus, "ASTOR: A Program Repair Library for Java", In Proc. of the 25 International Symposium on Software Testing and Analysis, pp. 441-444, 2016.

[4] T. Durieux, M. Monperrus, “IntroClassJava: A benchmark of 297 small and buggy Java programs”, pp.1-6, 2016.

[5] Z. Chen and M. Monperrus, “The CodRep Machine Learning on Source Code Competition,” ArXiv eprints, Jul. pp.1-6, 2018. arXiv: 1807.03200 [cs.SE].

[6] M. Martinez and M. Monperrus, "Astor: A Program Repair Library for JAVA", In Proc. of International Symposium on Software Testing and Analysis, pp. 441-444, 2016.

[7] Z. Chen, S. J. Kommrusch, M. Tufano, L. N. Pouchet, D. Poshyvanyk, & M. Monperrus, “Sequencer: Sequence-to-sequence learning for endto-end program repair”, IEEE Transactions on Software Engineering, pp.1-19, 2019.

[8] F. Long, and M. Rinard, “Automatic patch generation by learning correct code”, In Proc. of the 43rd Symposium on Principles of Programming Languages, pp. 298-312, 2016.
[9] M. White, M. Tufano, M. Martínez, M. Monperrus, and D. Poshyvanyk, “Sorting and transforming program repair ingredients via deep learning code similarities”, In Proc. of the 26th IEEE International Conference on Software Analysis, Evolution and Reengineering, pp.479-490, 2019.
[10] Yi. LI, Shaohua. WANG, NGUYEN, Tien N. “DLfix: Context-based code transformation learning for automated program repair”, In Proc. of the 42nd ACM/IEEE International Conference on Software Engineering, pp.602-614, 2020.
[11] XBD Le, D. Lo and C. Le Goues. "History driven program repair", In Proc. of the 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), vol. 1, pp. 213-224. IEEE, 2016.
[12] 호혜민, 양근석, 이병정, "코드 블록과 GAN 알고리즘을 활용한 자동 프로그램 버그 정정 기법", In Proc. of Korea Software Congress 2020 (KSC 2020), pp. 156-158, Dec. 2020.
[13] T. Lutellier, H. V. Pham, L. Li, Y. Pang, M. Wei, & L. Tan, “CoCoNuT: Combining context-aware neural translation models using ensemble for program repair”, In Proc. of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 101-114, 2020.

부록 1. Defect4J에 대한 프로그램 버그 정정 비교

Buggy Version	SequenceR	Our Approach	Buggy Version	SequenceR	Our Approach	Buggy Version	SequenceR	Our Approach
Chart_1	O	O	Closure_10	O	O	Lang_6	O	O
Chart_8	X	X	Closure_14	O	O	Lang_16	O	O
Chart_9	X	X	Closure_18	X	O	Lang_21	X	X
Chart_10	O	O	Closure_38	X	X	Lang_24	O	X
Chart_11	O	O	Closure_51	X	X	Lang_26	X	X
Chart_12	X	X	Closure_52	O	O	Lang_29	X	X
Chart_13	X	X	Closure_57	X	X	Lang_33	X	X
Chart_20	X	X	Closure_62	X	X	Lang_57	O	O
Chart_24	X	X	Closure_65	X	X	Lang_59	X	O
			Closure_67	O	O	Lang_61	O	O
			Closure_70	X	X			
			Closure_71	X	X			
			Closure_73	X	X			
			Closure_86	O	O			
			Closure_92	X	X			
			Closure_104	O	O			
			Closure_113	X	X			
			Closure_114	O	O			
			Closure_123	O	O			
			Closure_125	X	X			
			Closure_130	O	O			

Buggy Version	SequenceR	Our Approach	Buggy Version	SequenceR	Our Approach	Buggy Version	SequenceR	Our Approach
Math_2	O	O	Mockito_5	O	O	Time_4	O	O
Math_5	O	O	Mockito_8	X	X	Time_16	X	O
Math_11	O	O	Mockito_24	O	O	Time_19	X	X
Math_27	O	O	Mockito_26	X	X			
Math_30	X	X	Mockito_29	O	O			
Math_32	X	X	Mockito_34	O	O			
Math_33	O	O	Mockito_38	O	O			
Math_34	X	X						
Math_41	O	X						
Math_57	X	X						
Math_58	O	O						
Math_59	O	O						
Math_63	X	X						
Math_69	X	X						
Math_70	X	O						
Math_75	X	X						
Math_80	X	X						
Math_82	X	X						
Math_85	O	X						
Math_94	X	X						
Math_96	X	X						
Math_104	X	X						
Math_105	X	X						

테스트 커버리지 기반 장단기 메모리 모델 구조 결정

김민하, 이민수, 이찬근

중앙대학교 소프트웨어학부

minas.rtse@gmail.com, als950901@naver.com, cglee@cau.ac.kr

Test Coverage based LSTM Structure Decision

Min-Ha Kim, Min-Soo Lee, Chan-Gun Lee

Dept of Computer Science and Engineering, Chung-Ang University

요 약

최근 심층 신경망에 대한 연구가 활발히 진행되고 있으며, 이와 함께 심층 신경망의 검증을 위한 테스트 커버리지에 대한 관심이 높아지고 있다. 본 논문은 장-단기 메모리 모델에 적합한 테스트 커버리지 방법들을 사용하여 데이터의 규모와 특성에 따른 적절한 장-단기 메모리 모델 구조를 결정하는 데에 이용할 수 있다는 아이디어를 제안한다.

1. 서론

최근 수 년간 심층 신경망(Deep Neural Network)은 널리 연구됐으며, 자율주행 자동차[1], 음성 인식[2], 안면 인식[3], 의학 진단[4] 그리고 코드 분석[5] 등과 같은 다양한 도메인에서 비약적인 성장을 이루었다. 또한 이러한 성장을 바탕으로 심층 신경망에 대한 검증의 필요성이 부각되며, 이에 대한 많은 연구가 활발히 진행되고 있다[6]. 이러한 많은 연구들은 기존 소프트웨어 시스템을 검증하기 위해 사용하는 방법들을 심층 신경망에 사용할 수 있도록 변화를 거쳐 다양한 검증 방법을 만들어 적용하였으며, 이 중 널리 알려진 방식은 테스트 커버리지(test coverage) 방법이다.

일반적인 소프트웨어 테스트 커버리지 방법은 주로 코드 수준(level)에서 진행되는 검증 방식으로, 사용되지 않는 코드들을 탐색하지만, 심층 신경망에 적용할 수 있는 커버리지 측정 방식은 기존 소프트웨어 시스템에 적용할 수 있는 커버리지 측정 방식과는 다르게 코드 수준이 아닌 신경망을 검증 대상으로 하는 방식으로, 각 계층은 뉴런(neuron) 혹은 셀(cell)과 같은 유닛(unit)으로 구성하고 있기 때문에, 각 모델에 적합한 학습 알고리즘에 따라 커버리지 측정 방법을 달리한다. 즉, 이는 다음과 같이 2가지 방식으로 나누어 측정한다.

1) 뉴런 기반 커버리지 측정
2) 셀 기반 커버리지 측정

뉴런 기반 커버리지 측정 방식은 완전 연결 신경망(Fully Connected Neural Network, FCN)과 합성곱 신경망(Convolutional Neural Network, CNN) 등에 적용될 수 있으며, 셀 기반 커버리지 측정 방식은 순환 신경망(Recurrent Neural Network, RNN) 등에 적용할 수 있다. 즉, 순환 신경망의 계열인 장-단기 메모리(Long-Short Term Memory, LSTM)도 이러한 셀 기반 커버리지 측정 방식으로 커버리지를 측정할 수 있으며, 커버리지 측정 방식에 대한 자세한 설명은 2.2절에서 소개한다.

본 논문은 셀 기반 커버리지 측정 방식을 바탕으로, 본 논문에서는 신경망을 구성하고 있는 장-단기 메모리 셀을 심층적으로 분석하여, 데이터의 규모에 따라 적절한 신경망 구조를 결정하는 것을 목표로 한다.

이러한 커버리지 기반 신경망 테스트는 소프트웨어 결함(fault) 발견에 엄청난 성공을 보여주었으며, 이를 통해 피드 포워드 신경망(Feed-forward Neural Networks, FNN)을 이용한 작업이 광범위해졌다[7].

본 논문의 구성은 2절에서 실험에 관한 다양한 배경지식을 소개하며, 3절에서는 실험 진행을 위한 몇 가지 모델과 데이터 및 방법을 소개한다. 그리고 4절에서 실험 결과와 이에 따른 분석 결과를 소개한다. 마지막으로, 본 논문의 결론, 한계점 그리고 향후 연구에 대해 소개하며, 논문을 마무리한다.

2. 관련 연구

본 절에서는 본 논문에서 사용한 장-단기 메모리 모델에 관한 내용을 소개하며, 다양한 커버리지 방법을 소개한다.

2.1 장-단기 메모리 신경망

장-단기 메모리 셀은 이미 많이 알려져 있으며, 순환신경망에서 사용하는 은닉 계층(hidden layer)의 내부의 셀을 정교한 구조로 변형한 형태이며, 구조는 그림 1과 같다.

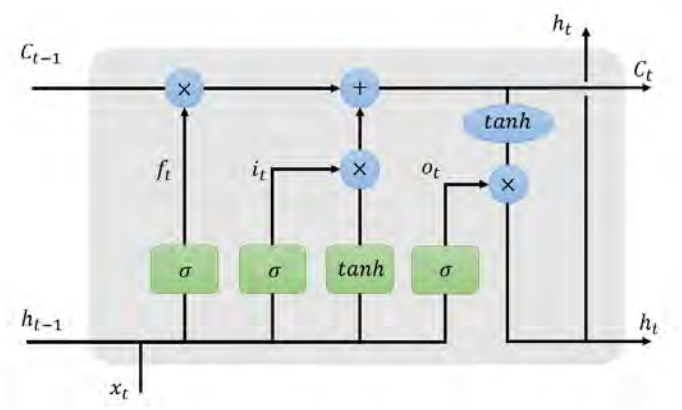


그림 1 장-단기 메모리 셀의 구조

가장 널리 알려진 장-단기 메모리 계층은 시간 t에 따라 다음과 같은 수식으로 표현된다[8].

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$c_t = f_t * c_{t-1} + i_t * tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(c_t) \quad (1)$$

위 수식에서 σ는 시그모이드(sigmoid) 함수를 뜻하며, 이에 대한 값은 0과 1사이의 값이다. 그리고 Wt, Wi, Wc, Wo는 가중치 행렬이며, bf, bi, bc, bo는 편향(bias) 벡터, ft, it, ot는 내부 게이트(gate) 변수들을 말한다. 마지막으로, ht는 은닉 상태(state) 변수이고 Ct는 셀 상태 변수를 의미하며, 이를 전부 합쳐서 장-단기 메모리의 구조적 구성요소(Structural Component)라고 한다.

위 구조적 구성요소를 바탕으로 만들어진 테스트 평가 지표는 통합 지식(Aggregate Knowledge)과 기억 비율(remember rate)에 의존한다. 그리고 장-단기 메모리의 단기 메모리로 표현되는 출력 값 h는 정보들이 어떻게 최신화가 이루어지는지 이해를 돕기 위해 사용되며, 이에 대해 다음과 같은 수식을 얻을 수 있다.

$$\xi_t^{h,+} = \sum\{h_t(i) \mid i \in \{1, \ldots, \mid h_t \mid\}, h_t(i) > 0\} \quad (2)$$

$$\xi_t^{h,-} = \sum\{h_t(i) \mid i \in \{1, \ldots, \mid h_t \mid\}, h_t(i) < 0\} \quad (3)$$

또한, (2), (3)의 수식을 이용하여 인접한 셀들의 은닉 상태를 비교하기 위해 다음과 같은 수식을 사용하여, 셀의 상태에 대해 관측할 수 있다.

$$\xi_t = \mid \xi_t^+ - \xi_{t-1}^+ \mid + \mid \xi_t^- - \xi_{t-1}^- \mid \quad (4)$$

망각 게이트(forget gate)인 f는 장-단기 메모리에서의 장기 메모리에 대한 핵심 요인이며, 통합 정보를 다음 셀로 통과시킬 지에 대해 제어한다. 이렇게 통과된 정보의 일부분은 다음과 같은 수식으로 계산될 수 있다.

$$\xi_t^{h,avg} = \frac{1}{|f_i|} \sum_{i=1}^{|f_t|} f_t(i) \quad (5)$$

이러한 장-단기 메모리를 사용하는 여러가지 유형이 있는데, 이를 구분하는 요소는 층의 개수와 입력 및 출력 값의 개수이다. 이에 따라 다양한 유형의 모델을 구축할 수 있다. 층의 개수에 따라 N-Stacked LSTM 모델이라 표현되며, 또한 입력 및 출력 값의 개수에 따라 One-to-Many, Many-to-One 등으로 표현된다.

본 논문에서는 이를 조합하여 4가지 모델을 구축하고, 이에 대해 검증을 진행하였다.

2.2 신경망 대상 커버리지 방법

Kexin Pei 등[9]은 심층 신경망을 검증하기 위해 뉴런 커버리지(Neuron Coverage, NC)를 제안하였으며, 심층 신경망 검증에 대한 발전 가능성을 이끌어냈다. 뉴런 커버리지는 모든 테스트 입력에 대해 활성화된 뉴런의 수와 심층 신경망의 총 뉴런수의 비율로 정의되며, 출력 값이 임계 값(threshold) 이상이면 뉴런이 활성화된 것으로 간주한다. 뉴런 커버리지를 계산하는 수식은 다음과 같다.

$$NC(T, x) = \frac{|\{n \mid \forall x \in T, out(n, x) > t\}|}{|N|} \quad (6)$$

또한, 뉴런 커버리지를 기반으로 다양한 커버리지 측정 방식이 등장했는데, Lei Ma 등[10]은 뉴런 수준과 계층

수준에서의 커버리지 측정 방식으로 구분하여, 다양한 커버리지 측정 방법을 제안하였다. 이에 대한 커버리지 측정 방법은 다음과 같다.

1) K-다중영역 뉴런 커버리지
(K-Multisection Neuron Coverage, KMNC)

$$KMNC(T,k) = \frac{\sum_n \in N \mid \{S_i^n \mid \exists x \in T : \phi(x,n) \in S_i^n\}}{k\times \mid N\mid} \quad (7)$$

KMNC는 주어진 테스트 입력에 대한 집합이 [low_n, high_n]의 범위를 얼마나 잘 커버하는지 측정하는 방식이다.

2) 뉴런 영역 커버리지
(Neuron Boundary Coverage, NBC)

$$NBC(T) = \frac{|UpperCornerNeuron \mid + \mid LowerCornerNeuron|}{2\times \mid N|} \quad (8)$$

NBC는 테스트 입력의 집합이 얼마나 많은 코너 케이스(Corner Case) 영역을 커버하는지 측정하는 방식이다.

3) 강한 뉴런 활성 커버리지
(Strong Neuron Activation Coverage, SNAC)

$$SNAC(T) = \frac{|UpperCornerNeruon|}{|N|} \quad (9)$$

SNAC는 테스트 입력에 의해 얼마나 많은 상한 경계 값을 커버하는 지에 대해 측정한다.

4) Top-K 뉴런 커버리지
(Top-K Neuron Coverage, TKNC)

$$TKNC(T,k) = \frac{|\bigcup_{x\in T}(\bigcup_{1\le i\le l} top_k(x,i))|}{|N|} \quad (10)$$

TKNC는 각 계층에서 한 시점에 얼마나 많은 뉴런이 활성화 되었는지를 측정한다.

5) Top-K 뉴런 패턴
(Top-K Neuron Pattern, TKNP)

$$TKNP(T,k) =\mid \{(top_k(x,1),\ldots,top_k(x,l)) \mid x \in T\}\mid \quad (11)$$

TKNP는 각 계층에 있는 상위 k개 뉴런들의 시퀀스를 형성한다.

위 방법들 중 KMNC, NBC, SBAC는 뉴런 수준 커버리지 측정 방법이며, TKNC와 TKNP는 계층 수준 커버리지 측정 방법이다. 이러한 다양한 커버리지 측정 방법을 사용하여 심층 신경망의 테스트 타당성을 평가하는 데에 유용한 지표가 될 수 있음을 보여주었다.

그리고 Junjie Chen 등[11]은 앞서 설명한 NC, KMNC, BNC, SNAC, TKNC를 구조적 커버리지(Structural Coverage)로 분류하였고, Jin-han Kim 등[12]이 제안한 가능성 기반 기습적 적절성(Likelihood-based Surprise Adequacy, LSA)과 거리 기반 기습적 적절성(Distance-based Adequacy, DSA)에 대응하는 커버리지 측정 방법들을 비 구조적 커버리지(Non-Structural Coverage)로 구분하였다. 비 구조적 커버리지는 심층 신경망의 구조적 요소가 테스트 중 포함되는지에 대한 여부를 고려하지 않으며, 테스트 입력과 학습 데이터 사이의 심층 신경망의 행동(Behavior)의 차이를 테스트 입력의 기습적 적절성(Surprise Adequacy, SA)로 간주한다.

또한, Wei Huang 등[8]은 순환 신경망에 대해 검증을 하고자 testRNN 도구를 개발하였으며, 이에 적용된 테스트 평가 지표는 다음과 같고, 수식은 [12]을 참고하였다.

1) 셀 커버리지 (Cell Coverage, CC)

$$CC = \frac{|t \mid \exists t \in T, \triangle\xi_t > \alpha_c|}{|T|} \quad (12)$$

셀 커버리지는 각 시간 단계(time step)마다 중요한 히든 상태에 대한 변화를 다루는 것을 목표로 한다. 셀의 값이 설정한 임계 값보다 더 크면, 테스트 케이스(test case)에 의해 활성화 및 커버되어졌다고 판단한다.

2) 게이트 커버리지 (Gate Coverage, GC)

$$GC = \frac{|(t \mid f_t > \alpha_c)|}{|T|} \quad (13)$$

게이트 커버리지는 셀 커버리지와 비슷하지만, 장-단기 메모리의 게이트로부터 정보가 추출되며, 게이트 커버리지를 사용하기 위해서 임계 값이 요구되어진다. 이러한 임계 값을 사용하여 망각 게이트의 셀 활성화 여부를 관측하고, 이를 바탕으로 커버리지를 계산한다. 즉, 해당 시간에 대한 망각 게이트의 출력 값이 임계 값 α를 초과하면, 활성화되었다고 판단한다.

3) 시퀀스 커버리지 (Sequence Coverage, SC)

$$SC = \frac{|(\gamma_1, \ldots, \gamma_{|T|} \mid \gamma \in \eta)|}{(|\eta|)^{|T|}} \quad (14)$$

시퀀스 커버리지는 은닉 상태를 거쳐온 순차적 정보를 캡쳐하며, 크게 양의 시퀀스 영역과 음의 시퀀스 영역으로 구성되며, 정규 분포를 따르는 경향이 있으며, 이러한 경향을 바탕으로 은닉 상태 h의 출현 영역을 같은 구간으로 분할이 될 수 있다. η는 각 구간에 할당되는 상징적 표현(symbolic representation)의 집합이다. 그리고 은닉 상태에 대한 정보를 바탕으로 양의 범위를 가지는 상징적 표현을 할당하는 방법을 양성 시퀀스 커버리지(Positive Sequence Coverage, PSC), 음의 범위를 가지는 상징적 표현을 할당하는 방법을 음성 시퀀스 커버리지(Negative Sequence Coverage)라고 표현한다.

또한 장-단기 메모리 모델들을 테스트 하기 위해 Wei Huang 등[7]은 영역 커버리지(Boundary Coverage, BC), 단계적 커버리지(Step-wise Coverage, SC) 그리고 시간적 커버리지(Temporal Coverage, TC)를 형식화하여 광범위한 실험을 진행하였다.

이에 대해, 앞서 설명한 커버리지 방법들을 다수 이용하여 Min-soo Lee 등[13]은 학습 데이터 대비 최적의 심층 신경망 구조를 찾는 분석 방법을 제시하였다. 즉, 같은 데이터를 학습한 여러 개의 심층 신경망 구조를 구축한 뒤, 커버리지 값을 이용하여 비교 분석하였고, 장-단기 메모리 층의 개수 및 완전 결합 계층의 뉴런 개수를 최적화하는 실험을 진행하였다. 이를 통해 서울 관악구 기상 데이터에 대한 최적의 모델 구조를 찾을 수 있었다.

본 논문에서는 장-단기 메모리의 유형 별로 모델을 구축하여 실험을 진행하기 때문에, 셀 커버리지, 게이트 커버리지 그리고 시퀀스 커버리지 측정 방법을 사용하여 실험을 진행하였으며, 자세한 실험 내용은 3절에서 소개한다.

3 실험 방법

본 절에서는 사용한 데이터와 모델에 관해 소개한다.

3.1 데이터 셋

순환 신경망을 이용하여 학습을 시킬 때, 일반적으로 순차적 데이터를 사용한다. 본 연구는 중국 베이징의 2010년 1월에서 2014년 12월 까지의 기상 데이터 관측 자료인 PRSA[12] 데이터를 사용한다. 해당 데이터의 특징(feature)에는 이슬점, 기압, 누적 풍속, 누적 강우 시간, 누적 강설 시간, 기온, 미세먼지 지수(PM2.5)등이 있다. 데이터의 설명에서 알 수 있듯이, 데이터의 특징으로 미세먼지 지수(PM2.5)를 포함하고 있으나, 기록되지 않은 시간대가 상당히 많다. 따라서, PM2.5를 제외하고 위에 언급한 총 6개의 특징을 사용하였다. 그리고 5년치를 1시간 마다 측정한 결과이기 때문에 총 데이터의 개수는 43,824개이며, 이를 분할하여 훈련 데이터(training data)는 35,024개를 사용하였고, 테스트 데이터(test data)는 8,760개를 사용하였다.

3.2 실험 프로세스

본 실험에 적용된 프로세스는 그림 2와 같다. 데이터를 통해 학습된 장-단기 메모리 모델에 테스트 케이스를 바탕으로 커버리지를 측정한다.

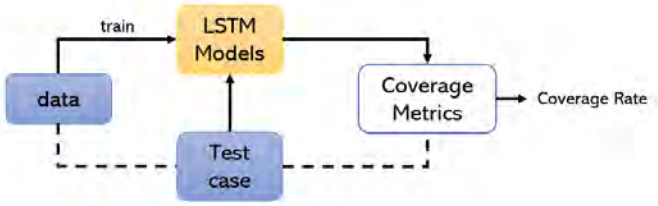


그림 2 본 논문에서 사용된 실험 프로세스

그림 2를 보면 실선과 점선으로 선이 구분되어 있는데, 실선은 작업에 대한 진행 흐름을 나타내며, 점선은 필요로 되어지는 정보를 표시하였다.

3.3 모델 평가

본 논문은 데이터 규모에 따라 적절한 장-단기 메모리 모델을 결정하는 데에 도움을 주기 위해, 비교를 위한 장-단기 메모리 모델의 유형들을 구현하였으며, 이에 PRSA 데이터를 적용하였다. 구현한 장-단기 메모리 모델의 유형은 다음과 같다.

표 1 실험에 사용된 모델들의 RMSE 결과

모델	RMSE
Many-to-One / 2 Stacked LSTM	0.125
Many-to-Many / 2 Stacked LSTM	0.131
Many-to-One / Multi-Layered LSTM	0.128
Many-to-Many / Multi-Layered LSTM	0.126

표 2 각 모델에 대한 커버리지 측정 결과

모델	CC	GC	PSC	NSC
Many-to-One / 2 Stacked LSTM	1.00	0.83	0.63	0.63
Many-to-Many / 2 Stacked LSTM	1.00	0.92	0.63	0.64
Many-to-One / Multi-Layered LSTM	1.00	0.75	0.62	0.63
Many-to-Many / Multi-Layered LSTM	1.00	0.91	0.63	0.62

1) Many-to-One / 2 Stacked LSTM
2) Many-to-Many / 2 Stacked LSTM
3) Many-to-One / Multi-Layered LSTM
4) Many-to-Many / Multi-Layered LSTM

본 논문에서는 순차적 데이터를 이용하기 때문에, Many-to-One, Many-to-Many 유형을 사용하였다. 또한, 실험을 진행하면서, 커버리지를 계산하기 때문에, 양방향(Bi-Directional) 모델은 사용하지 않았다. 즉, 단방향(Uni-Directional) 모델만을 구축하여 실험을 진행하였다.

평가 지표로는 루트 평균 제곱근 에러(Root Mean Squared Error, RMSE)를 사용하여 모델을 평가하였으며, 그 결과는 표 1과 같다.

표 1을 보면, Many-to-One / 2 Stacked LSTM 모델은 0.125%p, Many-to-Many / 2 Stacked LSTM 모델은 0.131%p, Many-to-One / Multi-Layered LSTM 모델은 0.128%p, Many-to-Many / Multi-Layerd LSTM 모델은 0.126%p를 기록했다. 이러한 결과로 보았을 때, 4가지 유형의 모델 모두 큰 차이가 없는 것을 확인할 수 있다.

즉, 다시 말해서 오차에 관한 비용 함수의 값만으로는 4가지 모델 중 해당 데이터에 적합한 모델을 선택하기가 어렵다는 결론을 내릴 수 있다.

4 실험 결과

본 절에서는 커버리지 측정 방식을 이용하여 4개의 모델 중 적합한 모델을 관찰하고, 이에 대한 결과를 소개한다.

4.1 커버리지 측정 결과

실험에 사용되는 4가지 모델에 대한 커버리지 측정 결과는 표 2에 표현하였다.

표 2에서 알 수 있듯, 셀 커버리지는 모두 100%p이고, 게이트 커버리지는 각각 83%p, 92%p, 75%p, 91%p이다. 또한 양성 시퀀스 커버리지는 Many-to- One / Multi-Layered LSTM 모델만 62%p이고, 나머지는 63%p이다. 마지막으로, 음성 시퀀스 커버리지는 각각 63%p, 64%p. 63%p, 62%p이다. 결과를 보았을 때, 다른 커버리지 방식들은 같거나, 값의 차이가 매우 적음을 알 수 있다.

하지만, 게이트 커버리지의 경우 다른 커버리지 측정 방법에 비해 상대적으로 더 큰 차이를 보여준다. 결과적으로, Many-to-One / Multi-Layered LSTM 모델은 다른 모델에 비해 실험에 사용한 순차적 데이터에 적합하지 않음을 알 수 있다. 표 2의 결과를 직관적으로 관찰하기 위해 이를 그림 3에 표현하였다.

그림 3의 가로 축은 실험에서 사용한 커버리지 측정 방식이며, 세로 축은 커버리지 비율(Coverage Rate)를 뜻한다.

해당 데이터에 더욱 적합한 모델을 탐색하기 위해, 게이트 커버리지에서 성능이 뒤쳐지는 Many-to-One / Multi-Layered LSTM 모델을 제외한 나머지 3개의 모델에 대해 셀 커버리지와 게이트 커버리지의 활성(activation) 횟수를 계산하였으며, 결과는 표 3과 같다.

표 3을 보면 셀 커버리지의 커버 횟수는 각각 1458, 1438, 1356회를 기록하였으며, 게이트 커버리지의 커버 횟수는 Many-to-One / 2 Stacked LSTM이 1512회, Many-to-Many / 2 Stacked LSTM 1693회, Many-to-Many / Multi-Layered LSTM이 1402회를 기록했다.

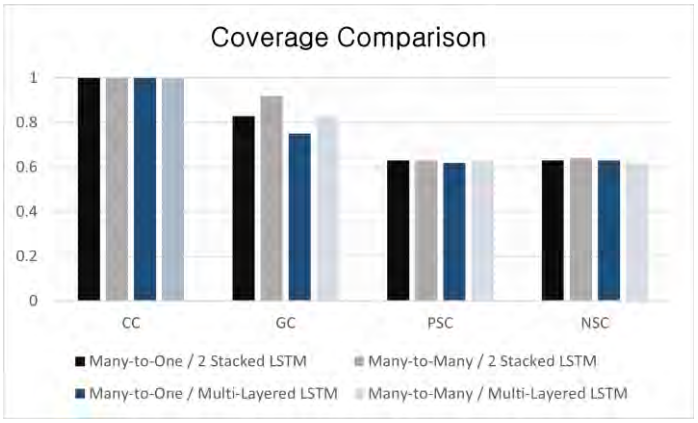
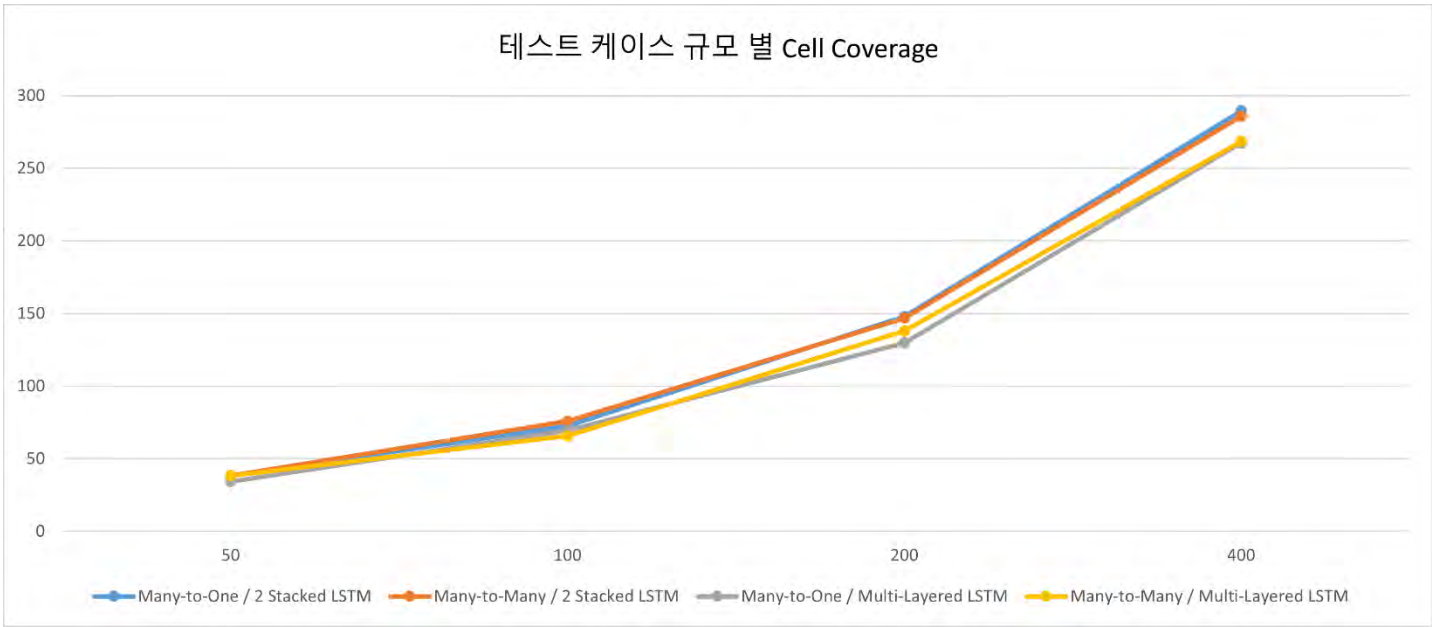


그림 3 모델 별 커버리지 측정 값 비교

표 3 모델 별 커버 횟수

모델	CC	GC
Many-to-One / 2 Stacked LSTM	1459	1512
Many-to-Many / 2 Stacked LSTM	1438	1693
Many-to-Many / Multi-Layered LSTM	1356	1402



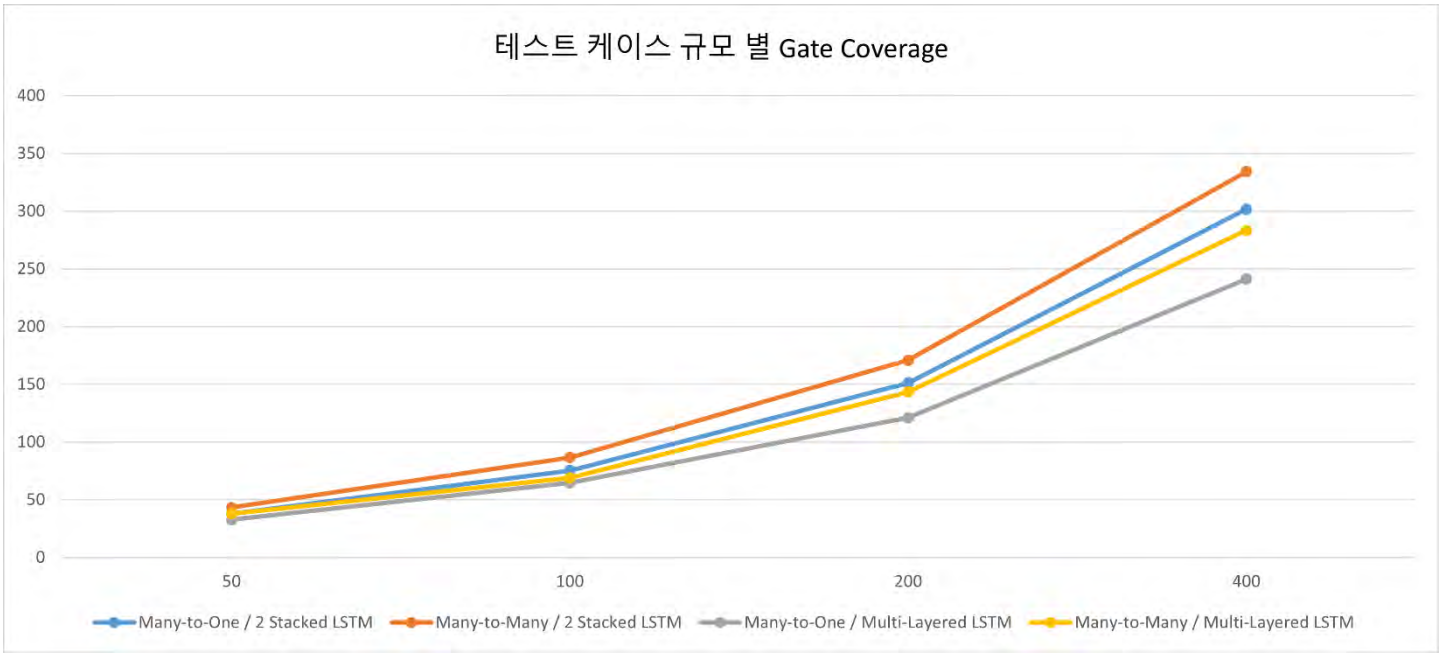


그림 4 테스트 케이스 규모 별 Cell Coverage(CC) 커버 횟수 변화량
테스트 케이스 규모 별 Gate Coverage(GC) 커버 횟수 변화량

이를 바탕으로 Many-to-Many / Multi-Layered LSTM 모델이 다른 두 모델에 비해 커버 횟수가 상대적으로 덜 된 것을 알 수 있다. 결과적으로, PRSA 데이터는 2 Stacked LSTM 모델이 좀 더 적합함을 알 수 있다.

추가적으로, 결과에 대한 신뢰성을 더 높이기 위해, 데이터 변화량에 따른 커버 횟수를 측정하는 실험을 진행하였다.

4.2 데이터 변화량에 따른 커버리지 측정

앞서 진행한 커버리지 측정 실험은 기본적으로 테스트 케이스(Test Case) 수를 2000으로 설정하였다.

이에 대해 추가적으로 테스트 케이스의 수에 변화를 주어 관찰함으로써 데이터 규모에 따른 영향력을 관찰한다.

변화는 50개부터 시작해서 400개 까지의 결과를 관측하며, 그 결과는 그림 4, 표 4, 5와 같다.

그림 4를 보면, 위 그래프는 셀 커버리지에 대한 커버 횟수 변화량이고, 아래 그래프는 게이트 커버리지에 대한 커버 횟수 변화량이다. 그리고 가로축은 테스트 케이스의 개수이며, 세로 축은 커버되어지는 횟수를 뜻한다.

표 4 테스트 케이스 규모별 Cell Coverage(CC) 커버 횟수

모델	50개	100개	200개	400개
Many-to-One / 2 Stacked LSTM	38	73	148	290
Many-to-Many / 2 Stacked LSTM	38	76	147	286
Many-to-One / Multi-Layered LSTM	34	70	130	268
Many-to-Many / Multi-Layered LSTM	38	66	138	269

표 5 테스트 케이스 규모별 Gate Coverage(GC) 커버 횟수

모델	50개	100개	200개	400개
Many-to-One / 2 Stacked LSTM	38	75	151	302
Many-to-Many / 2 Stacked LSTM	43	87	171	334
Many-to-One / Multi-Layered LSTM	33	65	121	241
Many-to-Many / Multi-Layered LSTM	38	69	144	284

셀 커버리지의 변화량의 경우, 비슷한 수의 셀을 커버하는 것으로 관찰할 수 있으며, Many-to-One / 2 Stacked LSTM 모델이 아주 근소하게 변화량이 더 높은 것을 확인할 수 있다.

게이트 커버리지 커버 횟수 변화량의 경우, 비록 그 수는 크지는 않지만, 눈에 띄게 Many-to-Many / 2 Stacked LSTM 모델이 높은 변화량을 가지는 것을 확인할 수 있다.

해당 실험을 통해 실험에 사용한 데이터에 대해서는 2 Stacked LSTM 모델이 더 적합함을 알 수 있으며, 해당 Many-to-Many / 2 Stacked LSTM 모델이 앞서 말한 4가지 모델 중 가장 최적화된 모델임을 관찰할 수 있다.

5 결론 및 향후 연구

우리는 오픈소스 기상 관측 데이터를 활용하여 다수의 장-단기 메모리 모델들을 학습시킨 뒤, 어떤 유형의 장-단기 메모리 모델에 더 최적인지 결정할 수 있는 분석 방법을 제시하였으며, 결과적으로 Many-to-One / 2 Stacked LSTM 모델이 가장 적합한 모델이라 판단하였다. 그리고 검증을 통해 모델을 선별할 때, 구조가 굉장히 비슷한 모델들을 비교할 경우, 시퀀스 커버리지 비율이 차이가 상당히 작음을 알 수 있었다. 즉, 구조가 비슷하다면, 최적의 모델을 선별하기에는 좋지 못한 지표가 될 수 있음을 확인하였다.

대규모의 데이터가 아닌 비교적 소규모 데이터로 실험을 진행했기 때문에, 일반화를 하기에는 한계점을 갖는다. 이를 바탕으로 후속 연구에는 대규모의 데이터들을 학습시킨 뒤, 분석하는 것을 목표로 한다. 또한 장-단기 메모리 모델 뿐만이 아닌, 합성곱 신경망과도 비교 및 분석하여, 더욱 다양한 카테고리에서 최적의 신경망을 찾는 것을 목표로 한다.

참고문헌

[1] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao “Deepdriving: Learning affordance for direct perception in autonomous driving,” Proc. of IEEE International Conference on Computer Vision(ICCV), pages 2722-2730, 2015

[2] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al, “Deep speech 2: End-to-end speech recognition in english and mandarin,” Proc. of International Conference on Machine Learning(ICML), pages 173-182, 2016

[3] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. “Deep learning face representation by joint identification-verification,” Proc. of the 27th International Conference on Neural Information Processing Systems(NIPS), Vol 2. pages 1988-1996, 2014

[4] Ziad Obermeyer, Ezekiel J Emanuel. “Predicting the future-big data, machine learning, and clinical medicine,” New England Journal of Medicine(NEJM), 375(13), pp. 1216-2019, 2016

[5] Xiaodong Gu, Hongyu Zhang, Sung-hun Kim, "Deep Code Search," Proc. of IEEE/ACM 40th International Conference on Software Engineering (ICSE), pp. 933-944, 2018

[6] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu, “Machine Learning Testing: Survey, Landscapes and Horizons,” IEEE Transactions on Software Engineering, pp. 1-37, 2019

[7] Wei Huang, Youcheng Sun, James Sharp, Wenjie Ruan, Jie Meng, Xiaowei Huang (2020, Mar 25), “Coverage Guided Testing for Recurrent Networks,”

(2nd ed.) [Online], Available: https://arxiv.org/abs/1911.01952
[8] Wei Huang, Youcheng Sun, James Sharp, Xiawei Huang (2019, Jun 20), “testRNN: Coverage-guided Testing on Recurrent Neural Network,” (1st ed.) [Online], Available: https://arxiv.org/abs/1906.08557
[9] Kexin Pei, Yinzhi Cao, Junfeng Yang, Suman Jana, "DeepXplore: auto mated whitebox testing of deep learning systems,” Proc. of 26th Symposium on Operating Systems Principles (SOSP), pp. 1-18, 2017
[10] 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, “DeepGauge: Comprehensive and multi-granularity testing criteria for gauging the robustness of deep learning systems,” Proc. of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE), pp. 120-131, 2018
[11] Junjie Chen, Ming Yan, Zan Wang, Yuning Kang, Zhuo Wu (2018, Aug 25), “Deep Neural Network Test Coverage: How Far Are We?,” (2nd ed.)[Online], Available: https://arxiv.org/abs/2010.04946
[12] Jin-han Kim, Robert Feldt, Shin Yoo, “Guiding Deep Learning System Testing using Surprise Adequacy,” Proc. of International Conference on Software Engineering(ICSE), pp. 1039-1049, May. 2019
[13] Min-soo Lee, Chan-gun Lee, “Deep Neural Network Structure Selection Using Coverage Methods,” Proc. of Korea Conference Software Engineering(KCSE), 2020
[14] ”Beijing PM2.5 Data Set”, UCI Machine Learning Repository, [Online], Available: https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data

협동 로봇 프로그래밍의 효율성 증대를 위한 마이크로서비스기반 플랫폼 설계

한성일 [1,2]O, 고인영 [2]

에이랍스 [1], 한국과학기술원 [2]

seongil.han@kaist.ac.kr, iko@kaist.ac.kr

Microservice-based Platform Design to Increase the Efficiency of Cobot Programming

Seong-il Han[1,2]O, In-Young Ko[2]

A-LABS[1], Korea Advanced Institute of Science and Technology[2]

요 약

협동 로봇은 사람과 같은 공간에서 함께 일할 수 있도록 개발된 로봇으로 산업현장에서 사용하기 위해서는 도메인에 맞는 프로그래밍 과정을 거쳐야 한다. 그러나 협동 로봇의 프로그래밍은 로봇 개발사, 주변기기 개발사, 최종사용자의 지속적인 개발과 통합이 이뤄져야 하는 구조를 갖고 있기 때문에 전통적인 소프트웨어 개발 방식인 모놀리틱(monolithic) 구조를 취하고 있는 대부분의 협동 로봇 소프트웨어 플랫폼은 기능(capability) 증가에 따른 개발과 배포의 비효율성 문제를 갖게 되었다. 이를 해결하기 위해 본 연구에서는 체계적인 컨텍스추얼 디자인 방법을 적용하여 협동 로봇 도메인의 다양한 이해관계자들의 사용 행태에 대해 조사하여 개선된 비전을 도출하고, 이를 반영한 마이크로서비스 기반의 협동 로봇 소프트웨어 플랫폼 아키텍처를 제안한다.

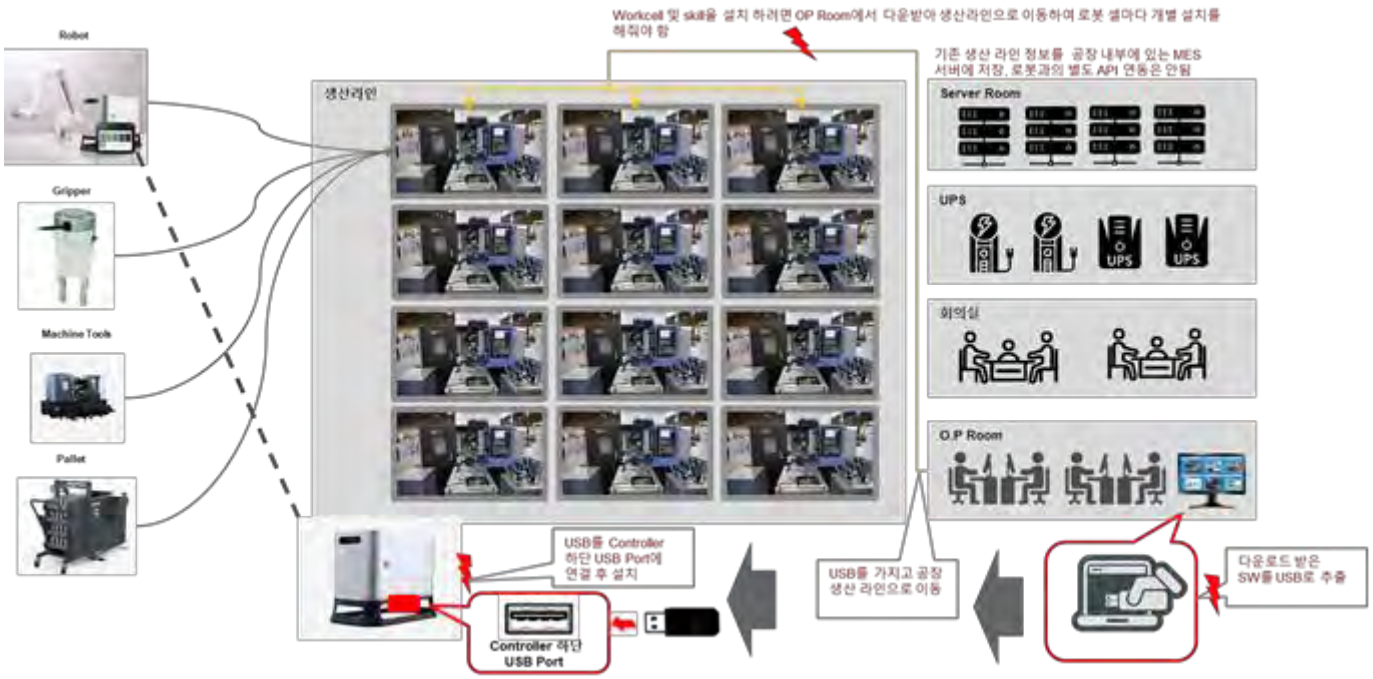
1. 서론

최근 산업 현장에서는 숙련기술자의 부족과 임금의 급격한 상승으로 인해 다양한 현장에서 로봇 등을 적용한 자동화시스템에 대한 요구가 지속적으로 증가하고 있고, 얘기치 못한 바이러스의 확산 및 근로자의 감염으로 인해 산업현장의 폐쇄가 이뤄지고 있는 상황에서 이런 자동화에 대한요구는 더욱더 증가되고 있다. 이런 상황에서 최근 부상하고 있는 로봇의 분야인 협동 로봇은 산업 현장에서 뿐만 아니라 바리스타 로봇, 치킨로봇의 등장으로 요식업이나 다양한 환경에서 로봇이 적용되는 사례가 늘어 감에 따라 많은 분야에서 협동 로봇을 활용한 자동화에 대해 관심이 높아지고 있는 상황이다.

최근 협동 로봇이 주목받게 된 이유는 산업 현장의 산업용 로봇 적용은 복잡하고 까다로운 안전규약으로 인해 로봇이 사람과 분리된 공간안에서 사람의 개입없이 로봇이 완전한 작업을 할 하도록 개발되어야 하기 때문에 기존 작업공정의 복잡도에 따라 수많은 보조장치들을 맞춤 제작을 하고, 맞춤 제작까지 걸리는 시간사이에 사전 프로그래밍을 위해 제작될 기기와 현장에 대한 모델링을 기반으로 가상환경에서 프로그래밍을 한 후 제작이 완료되면 현장에 설치하며 미세한 조정을 한다. 보조장치 제작까지 많은 시간이 걸리고 중간에 작은 오차가 있으면 다시 제작을 하는 상황도 자주 발생하여 수정하고 대기하는 상황이 자주 반복된다.

반면 협동 로봇은 안전 기능이 강화되어 사람과 같은 공간에서 협업할 수 있게 개발된 로봇으로 비교적 적은 안전제약과 간단한 설정으로 펜스 등의 설치 없이 사람과 같은 공간에서 일을 할 수 있다. 이로 인해 사람의 현장 개입이 자유로워지게 되며 맞춤형 장비가 없더라도 사람이 개입하여 작업공정이 진행 할 수 있기 때문에 로봇과 필수기기만으로도 현장에서 바로 프로그래밍 할 수 있어 로봇 현장 적용시간을 극적으로 줄일 수 있다.

[그림 1]협동 로봇과 GUI



[그림 2] 공장환경의 피지컬 모델

이처럼 현장 적용이 빨라진 협동 로봇은 GUI 기반의 직관적인 프로그래밍 환경을 제공하고 [그림 1]과 같이 주변기기와 통합된 기능을 제공하며 최종사용자가 프로그래밍 하기 쉽고 사용자 저변이 넓어지게 만들어 이는 또다시 주변기기가 모듈화 되고 전문화된 새로운 로봇 주변기기 회사가 등장할 수 있는 길을 열어 주는 선순환 구조를 만들게 되었다.

그러나 이런 다양한 로봇 주변기기와의 통합은 로봇 개발사와 주변기기 회사에 새로운 부담을 증가시켰다. 로봇프로그래밍 소프트웨어는 로봇 제어기에 통합된 모놀리틱(Monolithic)구조로 개발되어 왔고, 모놀리틱 구조는 증가하는 로봇 주변기기에 맞춰 통합된 새로운 기능이 추가될 때마다 새로운 로봇 소프트웨어 개발과 배포가 필요하게 되었다.

로봇 자체 기능 추가가 아닌 외부요인에 의한 기능(capability) 증가에 따른 전체 소프트웨어 개발과 배포 라는 부담을 로봇 개발 회사가 갖게 되었다.

주변기기 회사 역시 출시 초기에는 특정 로봇 회사와의 협업을 통해 출시하기도 하지만 모놀리틱 구조에서의 통합은 특정 회사에 맞춰진 개발이었고 이로 인해 각 로봇회사마다 별도의 소프트웨어를 개발하는데 부담을 갖게 되었다.

로봇회사의 배포 문제와 주변기기 회사의 중복 개발 문제는 협동 로봇 생태계 확장을 더디게 만들었으며 이는 최종사용자가 원하는 쉬운 프로그래밍 환경을 사용하지 못해 기존 산업용 로봇과 동일하게 외부 프로그래머를 고용하여 로봇을 프로그래밍 하게 되어 비용부담을 가중시키는 문제를 갖게 되었다.

이에 본 연구에서는 협동 로봇 프로그래밍의 효율성 증대를 위해 EUSE 관점에서 컨텍스추얼 디자인(Contextual Design)을 사용하여 다양한 이해관계자들의 불편함을 조사하여 개선된 비전을 제시하고, 아키텍처 개선을 위해 마이크로서비스(Microservice) 기반의 협동 로봇 환경소프트웨어 플랫폼 아키텍처를 제안하고자 한다.

2장은 연구와 관련된 문서에 대한 간략한 개요를 제공한다. 3장은 전체 과정을 이해하기 위한 배경지식에 대해 설명하고 4장에서는 효율성 증대를 위해 컨텍스추얼 디자인을 통해 이해관계자들의 불편함을 도출하여 개선된 비전을 제시한다. 5,6장은 4장에서 제시된 비전에 적합한 마이크로서비스 아키텍처 기반의 플랫폼 아키텍처를 제시하고 간략한 프로토타입을 구현하였고, 섹션 7에서 결론 마무리된다.

2. 관련연구

많은 협동 로봇 최종사용자는 프로그래밍 전문가가 아닌 로봇이 도입되기 전 현장에서 실제 업무를 담당하던 직원이 유지보수와 관리를 시작하며 로봇 프로그래밍을 배운다. 이처럼 협동 로봇 프로그래밍 소프트웨어는 프로그래밍 비전문가가 사용하는 소프트웨어로서 End User Software Engineering (EUSE)의 한 분야다. EUSE 관점에서 최종사용자는 일반 소프트웨어 엔지니어와 동일한 소프트웨어 엔지니어링 문제에 직면해 있다.

예를 들면, API 라이브러리 및 함수를 선택해야 하고 프로그래밍을 완료한 후에는 프로그램을 위한 테스트 및 디버깅을 하고 역시 프로그램의 오류 대한 큰 위험을 갖고 있다. EUSE의 일반 소프트웨어

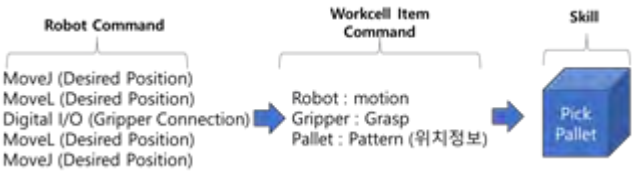
엔지니어링과는 다른 특징으로 도메인 키워드 기반의 프로그래밍이 가능하며[1] 서비스 지향 컴퓨팅의 패러다임으로 전문 개발자가 아닌 최종 사용자가 자신의 웹을 디자인하거나 사용자 정의한다는 것이다[2].

마이크로 서비스는 다양한 분야에 적용하기 위한 연구가 활발히 진행되어 왔고, 이에 연장선상으로 [3],[4] 스마트 팩토리(Smart Factory) 시스템의 일부인 Cyber-physical system(CPS) 로서 연구가 진행되었다. 그러나 이런 연구들은 전통적인 자동화 공정과 센서 기반의 시스템의 연구로서 사람과 함께 같은 공간에서 동작하는 협동 로봇 사용에서의 특징을 고려하지 않은 연구이다. [5]기존의 모놀리틱 시스템을 마이크로서비스 기반 클라우드 시스템으로 전환하여 유연한 배포 시스템을 통해 로봇 애플리케이션 개발이 편해 졌음을 증명하였지만 이는 폐쇄적인 네트워크 구조를 갖고 있는 일반적인 공장에서는 사용할 수 없는 방법이었다.

이에 본 연구에서는 최종 사용자의 사용성 개선을 위해 도메인지식 기반의 서비스 컴포지션(Service Composition)과 이를 가능하게 해주는 클라우드 기반 마이크로서비스를 사용하였고, 공장 상황에서의 폐쇄 망 사용의 한계를 극복하기 위해 엣지 클라우드의 개념을 추가하는 연구를 진행하였다. 또한 다양한 분야의 로봇 개발자들의 프로그래밍 효율성 개선을 위해 컨텍스추얼 디자인(Contextual Design)[6]을 통해 나온 실 사용에서의 불편함 들을 도출해 개선했다.

3. 배경지식

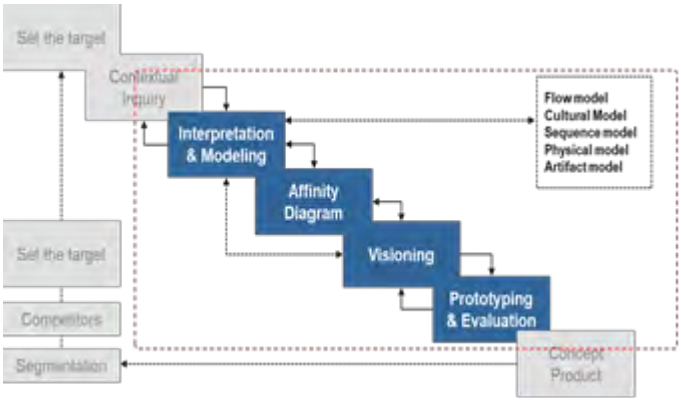
본 연구는 두산로보틱스의 개발환경인 스킬 기반 프로그래밍(Skill based Robot programming)을 사용하여 진행하였다[7]. 두산로보틱스의 스킬 기반 로봇 프로그래밍은 사용자가 로봇이 수행하여야 할 작업을 손쉽게 프로그래밍할 수 있도록 사용자 입력 정보를 기반으로 필요한 스킬(Skill) 및 태스크 템플릿을 제공하는 스킬 기반 로봇 프로그래밍 장치이다. 구성은 로봇과 로봇주변장치를 포함하는 워크셀 아이템(Workcell Item) 워크셀 아이템 중 사용할 워크셀 아이템을 선택하고, 선택된 워크셀 아이템의 파라미터를 입력하는 워크셀 매니저(Workcell Manager), 스킬을 바탕으로 로봇이 작업해야 할 태스크를 생성하는 태스크빌더(Task Builder)로 구성 되어있다. 실제 사용하는 환경은 [그림2]의 공장 환경의 피지컬 모델을 참조할 수 있다.



[그림 3]두산로보틱스 스킬의 구성

[그림 3]두산 로봇의 스킬 기반 프로그래밍은 로봇과 함께 구성되는 작업공간에 배치되는 주변 기기들에 대한 정의부터 시작된다. 작업 공간에 배치되는 하나의 기기를 워크셀아이템(Workcell Item)이라고 하며 워크셀 아이템에서 동작하는 워크셀아이템커맨드(Workcell Item Command)로 동작하도록 되어 있다. 최종사용자가 사용하게 되는 스킬은 워크셀아이템커맨드의 집합으로 구성되어 있다. 즉 스킬 하나는 로봇이 동작하는 공간에서 행해지는 하나의 도메인 지식으로 구성되어 있다.

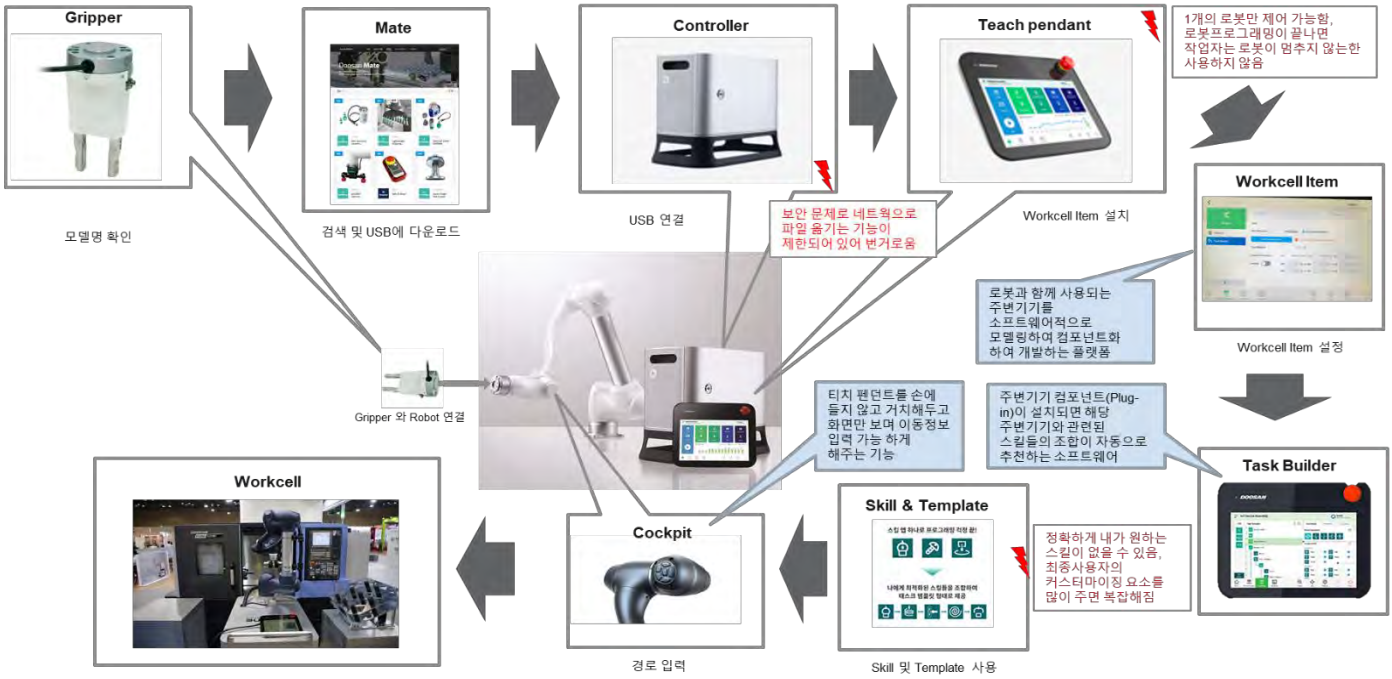
4.컨텍스추얼 디자인



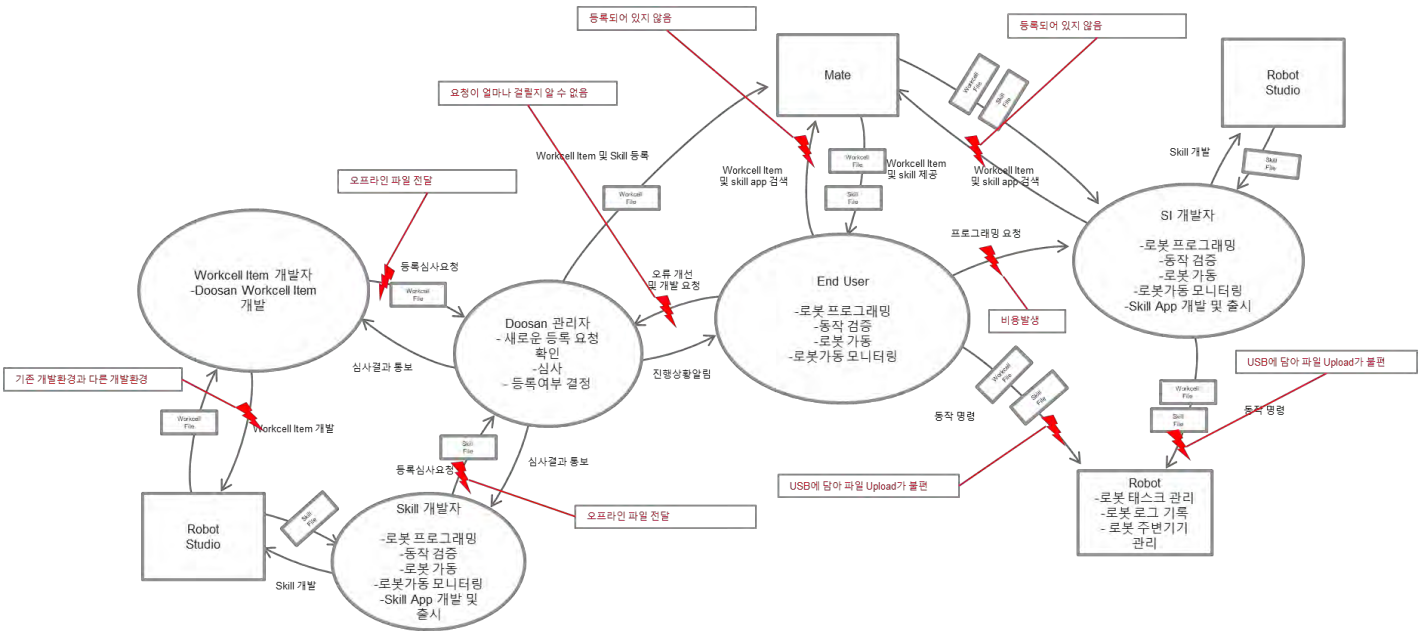
[그림4]컨텍스추얼 디자인

[그림 4]컨텍스추얼 디자인 이란 사용자 중심의 디자인 프로세스로 현장 연구를 통해 제품과 관련된 데이터를 수집하고 인간 사회와 문화의 다양한 현상을 정성적, 정량적 조사기법을 통해 워크 플로우를 합리화하고 디자인하는데 사용하는 기법이다. 협동 로봇소프트웨어 개발은 일반적인 소프트웨어 개발과는 다르게 공간과 사람 로봇으로 이루어진 공간에서 사람의 행태 변화를 이끌어 내어 주변환경의 물리적 환경에 많은 영향을 받아. 상황에 따른 물리적 제약과 관련된 문제점을 찾아내는 방법이 효과적이라 판단되어 적용하게 되었다. 문제 파악을 위해 실제 사용환경에 대한 모델정리를 중심으로 진행하였고, 본 연구에서는 협동 로봇의 대표적인 적용분야인 머신탠딩에서 공작기계에 작업물을 로딩 언로딩 하는 분야를 선택하여 진행하였다. 피지컬 모델(physical model), 아티팩트 모델(artifact mode)l, 플로우 모델(Flow model), 시퀀스모델(Sequence model) 사용해여 친화도 다이어그램(Affinity Diagram)을 수집하고 아이디어를 도출하는 방식으로 진행하였다.

4.1 피지컬 모델은 업무에 영향을 미치는 사용자의 실제 물리적 환경을 나타낸다. 기본적으로 협동 로봇은 사람과 함께 일하게 되는 소규모 셀로 구성되며 셀의 개수에 따라 작은 공장부터 여러 개의 셀로 구성된 중규모, 대규모까지 다양한 환경을 타겟으로 하고 있다.



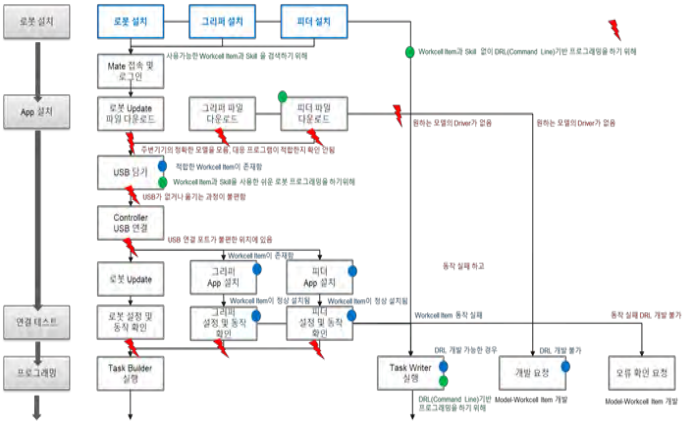
[그림 5]아티팩트 모델



[그림 6]플로우 모델

[그림1]을 보면 하나의 셀은 보통 로봇과 그리퍼 공작기계 팔레트로 기초적은 셀이 구성되고 생산 현장은 이런 셀들이 여러 개가 구성되는 형태로 되어있다. 보통의 공장은 공장을 운영하는 인력이 위치하는 사무실과 회의실, MES등을 운영하는 서버 실, 등으로 구성되며, 소규모 영세사업장이 아닌 이상 보안상의 이유로 외부망과 내부망을 분리하여 사용하는 형태를 갖고 있다. 두산로보틱스의 스킬과 템플릿을 사용하기 위해서는 온라인에서 Mate 사이트의 검색을 통해 적합한 App을 다운받아 각 로봇에 설치해야 하는데 이런 환경에서는 보안상이유로 네트웍 연결이 쉽지 않아 개별 로봇마다 USB를 연결해서 설치해줘야 하는 상황인다 [그림1]에서처럼 로봇의 USB포트는 컨트롤러의 하단에 있어 설치가 불편한 브레이크 다운요소들이 도출되었다.

4.2 아티팩트 모델은 사용자가 직무를 수행하면서 만들거나, 전달하거나, 또는 참고하는 실제 '사물'또는 전자 장비들의 사본이나 샘플을 말한다. 두산 로봇의 그리퍼를 쉽게 사용하기 위해서는 [그림 5]에 나와있는 설명처럼 Mate 사이트 접속을 통해 다운로드 받아 USB에 연결하고, 로봇을 조작하는 티치펜던트를 통해 설치 및 설정 값들을 입력하고 나면 추천된 스킬과 템플릿으로 프로그래밍하여 Cockpit을 통해 이동

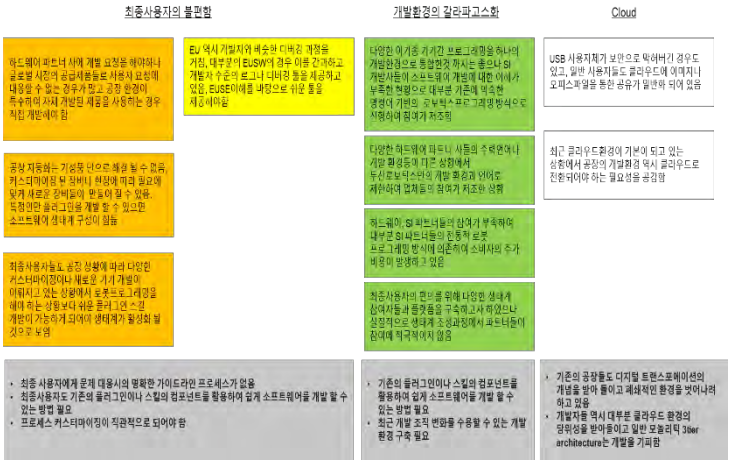


[그림 7] 시퀀스 모델

점들을 지정한 후 운영하게 된다. 이 과정에서 나온 불편함 들 역시 파일의 이동과 설치에 대한 문제가 확인되었으며, 스킬의 부재나 커스터마이징 요소가 많아짐에 나타나는 불편함 들이 확인되었다.

4.3 플로우 모델은 이해관계자들의 관계를 중심으로 어떤 흐름으로 일이 진행되는지 표현하는 모델로 이 모델을 통해 실제 최종사용자와, 잘 드러나지 않지만 중요한 역할을 하는 다양한 이해관계자들을 모델링 할 수 있었다. [그림 6]에서처럼 최종사용자가 스킬을 사용하기 까지는 하드웨어 개발사의 소프트웨어인 워크셀 아이템 개발자가 있고, 스킬 개발자, SI 개발자와 최종사용자 역시 개발자로 다양한 개발자들이 참여하고 있었음을 확인하였고, 현재 최종사용자가 원하는 어플리케이션이 없거나 오류가 발생했을 때 오프라인으로 이동되는 정보들이나 불편함 들로 인해 문제해결이 될 수 있는지에 대해 알 수가 없는 문제점이 도출되었다. 또 다른 특이사항으로는 최종사용자에게 적합한 스킬이 없고 프로그래밍을 하지 못하는 경우 SI 개발자에게 요청하여 추가적인 비용이 발생하는 상황이 확인되었다

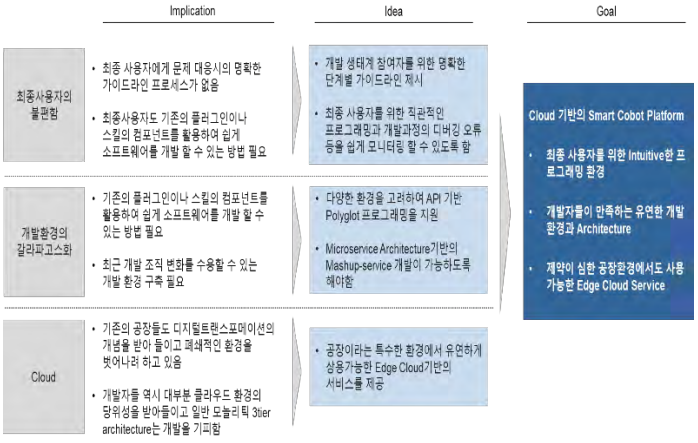
4.4 시퀀스 모델은 관찰된 직무를 단계적으로 기록하는 것으로 완성된 시퀀스는 사용자가 업무를 완수하고자 관여하는 핵심 전략과 활동을 보여주는 모델이다. 최종사용자가 두산로봇을 사용하는 것으로 시작해 핵심 활동 모델링하였다. 대표적으로 [그림7]최종 사용자의 시퀀스 모델을 보면 이전 모델에서도 나왔던 USB를 사용한 설치가 공통적으로 문제점으로 도출되었고, 주변기기에 대한 주변기기에 대한 정보를 입력하여 검색했으나 원하는 모델이 없는 경우가 발생하는 경우, 적합한 모델을 설치하였더라도 동작 실패 시 프로그래밍을 하지 못하는 경우 오류확인을 외부에 요청해야 하는 문제가 발생할 수 있는 문제가 발견되었다.



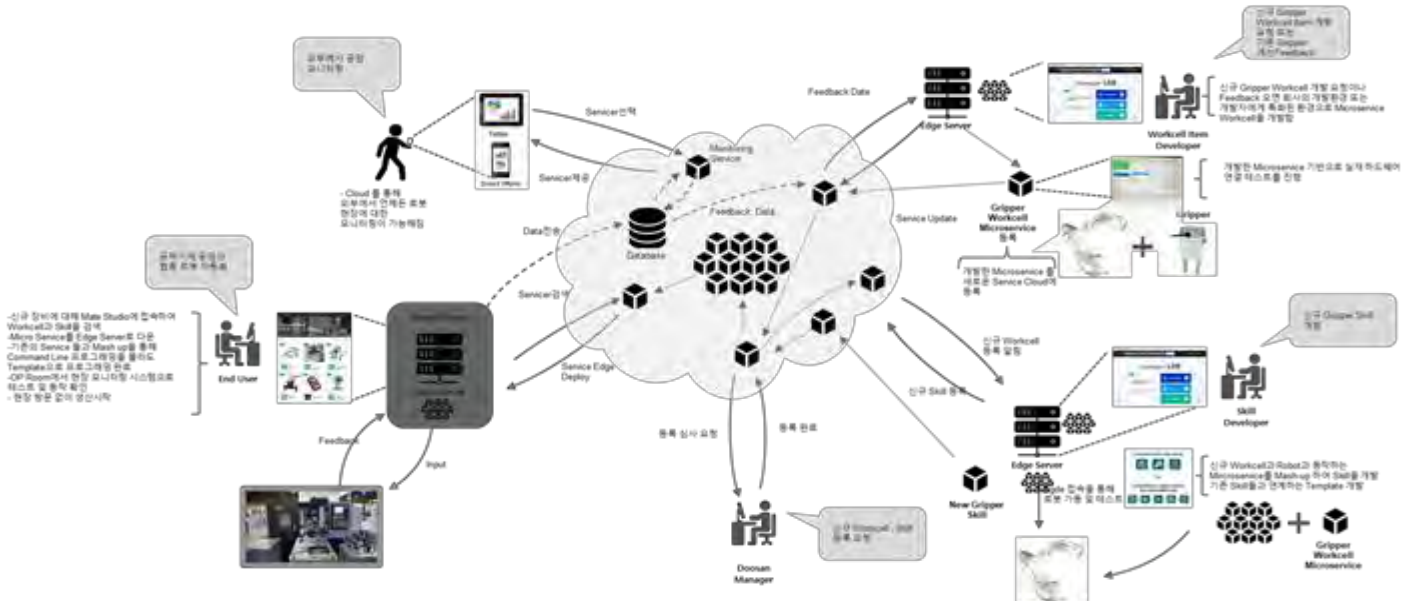
[그림 8]어피니티 다이어그램

4.5 어피니티 다이어그램은 사용자 집단에 관한 이슈를 계층적으로 표현한 것으로 각 모델 도출 과정에서 나온 것들을 해석하여 구축된 어피니티 노트로부터 구축된다. [그림 8]과 같이 최종사용자의 불편함, 개발환경의 갈라파고스와, 클라우드의 3가지 주제가 도출되었다. 명확한 워크 플로우 흐름이 나타나지 않는 문제가 발견되었고, 그리고 플랫폼 활성화를 위해서는 다양한 개발자들의 참여가 가장 중요하다는 것이 확인되었다. 개발자들의 진입 장벽을 낮출 수 있는 방법이 필요하고, 클라우드 환경의 니즈에 대해서도 확인할 수 있게 되었다

4.6 해결방안 도출 [그림 9]와 같이 크게 3가지 그룹으로 해결 방안이 정리되었다. 최종목표는 클라우드 기반의 스마트 협동 로봇 플랫폼으로 최종 사용자를 위한 직관적인 프로그래밍환경 개발자들이 만족하는 유연한 개발환경과 아키텍처 제약이 심한 공장환경에서도 사용가능한 엣지 클라우드 서비스를 목표가 수립되었다



[그림 9]문제 해결방안 도출



[그림 10] 비전

4.7 비전 수립

업무의 모델을 정리하고 어피니티를 만든 다음 이슈와 시사점을 찾아내 새로운 목표를 세웠다. 이를 기반으로 재 디자인된 시스템이 어떻게 효율성을 개선하는지를 보여주는 비전 작업을 진행했다.

[그림 10]을 보면 제일 먼저 하드웨어 관련된 개발자들은 클라우드 또는 엣지 클라우드 기반에서 개발한 서비스를 엣지클라우드를 통해 오프라인 연결 없이 온라인으로 로봇에서 테스트할 수 있게 되었으며 이 개발된 서비스는 바로 클라우드로 전송되어 두산 관리자에게 승인 요청이 진행되며 승인된 서비스는 스킬과 템플릿 개발자들에게 새로운 서비스로 알림이 가도록 되었다.

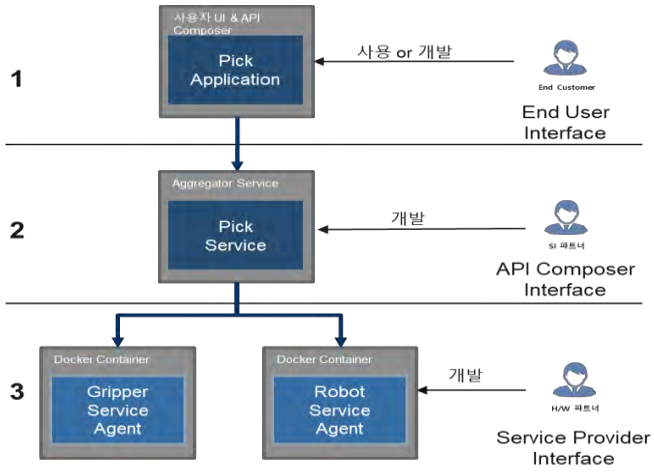
스킬과 템플릿 개발자들 역시 클라우드 환경에서 개발을 진행하고 테스트할 수 있게 되었으며 하드웨어 서비스 개발자들과 동일하게 클라우드로 바로 서비스가 전송이 되어 등록이 되고 최종 사용자는 언제든지 자기가 원하는 서비스가 개발되었는지를 쉽게 확인하고 엣지클라우드로 내려 받아 테스트해보고 사용할 수 있게 되었으며, 보안 문제등의 해결이 가능하다면 외부에서 접속할 수 있도록 클라우드로 데이터를 전송하여 외부에서 모니터링과 제어가 될 수 있다.

추가적으로 서비스에 하드웨어 상태가 모니터링 가능하도록 기록하여 해당 기록 등을 주기적으로 개발자들에게 전송하여 하드웨어 상태를 모니터링 할 수 있게 되었으며 이로 인해 예지정비나 다양한 데이터 기반의 서비스를 개발할 수 있는 환경이 구성되었다

5. 마이크로서비스 기반의 협동 로봇 플랫폼

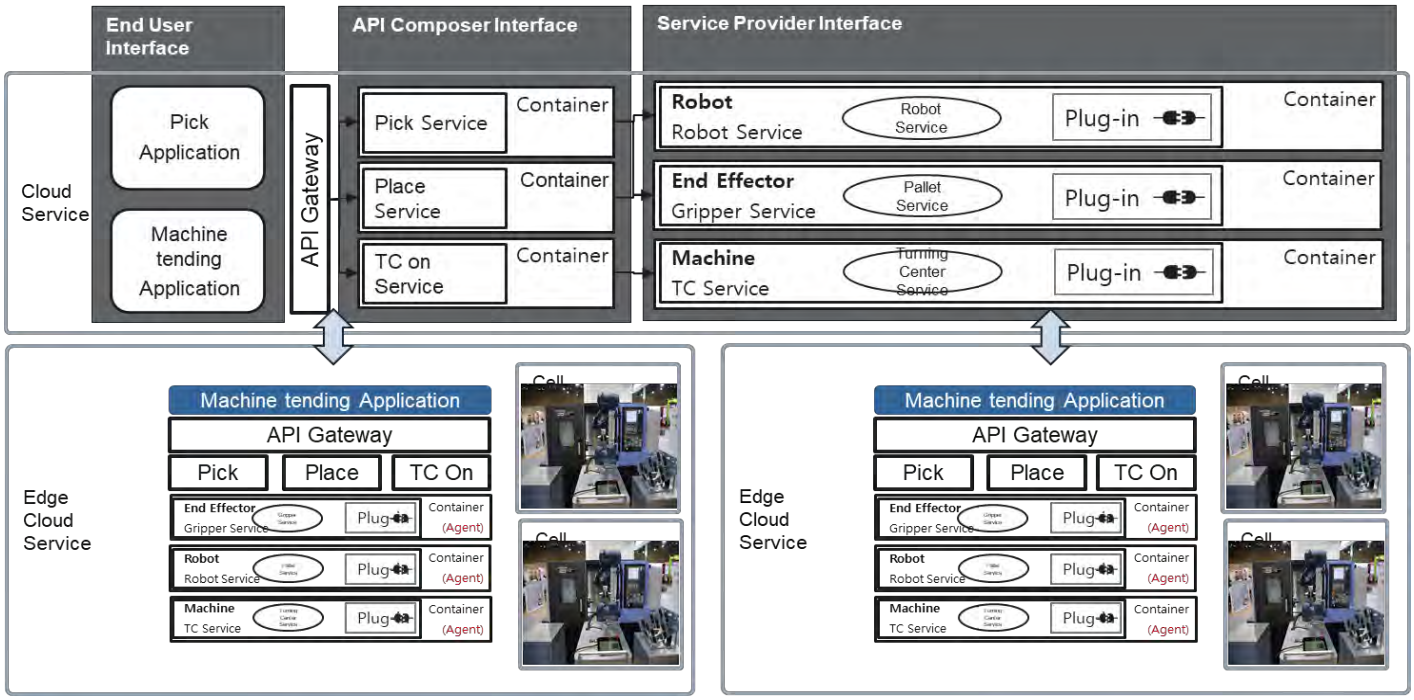
마이크로서비스 아키텍처의 서비스들은 세분화 되어있고 경량 프로토콜을 사용한다. 애플리케이션을 더 작은 여러 서비스로 분해할 때의 장점은 모듈성을 개선하고 애플리케이션의 이해, 개발, 테스트를 더 쉽게 해주고 애플리케이션을 더탄력적으로 만들어 준다[8]. 규모가 작은 자율적인 팀들이 팀별 서비스를 독립적으로 개발하고 규모 확장을 할 수 있게 함으로써 독립적으로 개발할 수 있게 한다[9].

두산로보틱스의 스킬은 [그림 3] 과 같이 로봇을 포함한 주변기기 단위로 서비스가 구분된다. 스킬은 주변기기 기능(function) 조합으로 구성되며 최종적으로 사용자는 Skill을 사용하거나 스킬을 조합하여 새로운 어플리케이션을 만들 수 있다.



[그림 11]Pick Service의 구성

마이크로 서비스 구조로 이를 변환하면 [그림 11]와 같다. 하드웨어 서비스를 제공하는 파트너는 하드웨어 기기 마이크로 서비스를 개발하여 등록하게 되고 스킬을 개발하는 개발자는 API Composer를 통해 하드웨어 개발자가 만든 마이크로서비스를 조합하여



[그림 12] 협동 로봇 마이크로서비스 아키텍처

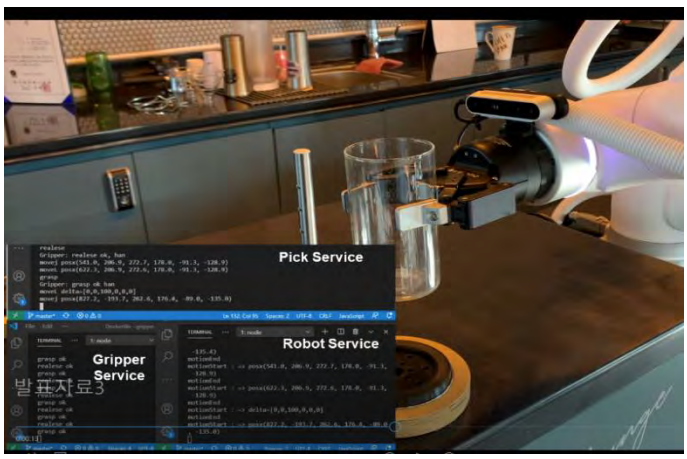
Pick 이라는 서비스를 만들고, 스킬을 조합한 서비스를 만들 수 있다. 최종사용자는 사용자 UI를 통해 등록된 Pick Application을 사용하거나 다른 Service 들과 조합하여 새로운 Application을 만들 수 있다.

[그림11]의API Composition 서비스인 Pick Service는 1,2,3의 위치에 있을 수 있다. 각 위치에 따라 특장점이 있다. 먼저 1의 위치에 있으면 각각의 API 조회 결과를 직접 UI에서 Aggregation 하는 방식이다. API를 처리한 결과를 순차적으로 처리하기 때문에 로딩시간이 많지만 유연하다는 장점이 있다. 2에 위치한 경우 단일 최종사용자와 말단 서비스를 중개하도록 구성하는 방안이다. 이 경우 양쪽의 종속성에서 벗어날 수 있고 API Gateway에 위치해 보안, 인증과 같은 기능을 함께 제공할 수 있게 된다. 3의 경우 서비스의 직접호출로 종속성이 발생하지만 빠른 응답이 가능하다. Pick Service는 Gripper Service와 Robot Service에 종속된 구조로 개발되고 서비스간 빠른 응답이 중요한 로봇 통신의 특성상 3에 위치하도록 하였다.

[그림 12]은 전체 아키텍처를 나타낸다. 하드웨어 파트너는 하드웨어 통신 규약을 맞춘 Plug-in 소프트웨어를 기반으로 기기 서비스추가한 워크셀 아이템을 컨테이너(Container)기반의 개별 마이크로 서비스를 개발해서 등록하고 SI 파트너인 스킬 및 템플릿 개발자들은 등록된 서비스들을 확인하여 워크셀 아이템 개발자들이 등록한 마이크로 서비스를 매시업 하여 최종사용자가 사용할 어플리케이션을 개발 테스트하여 등록하고 되고 최종사용자는 인증 배포된 최종 어플리케이션을 확인 검증하여 사용할 수 있게 된다. API Gateway는 중간에서 발생하는 인증 보안과 같은 공통 작업을 하도록 구성되었다. 모든 사용자들은 각자의 엣지 클라우드(Edge Cloud)가 구축이 되어 있다면 이를 사용할 수 있다.

6. 구현 및 평가

[그림 12]의 아키텍처의 컨셉을 검증하기 위해 로봇의 가장 기본적인 서비스 [그림 3]을 참고하여 Robot Service는 이동하고자 하는 로봇 3차원 좌표계와 호출하면 해당 좌표를 이동하는 기능을 개발하고 Gripper Service는 Grasp와 Release를 개발했다. 로봇 서비스와 그리퍼 서비스 두개의 하드웨어 서비스를 매시업 하여 최종사용자를 위한 [그림 13]와 같이 동작하는 Pick Service 프로토타입을 개발했다. 각 서비스는 별도의 컨테이너에서 node.js로 개발되었다.



[그림 13]개발된 서비스와 어플리케이션

Gripper Service는 Robot Service를 전혀 참조하지 않아도 개발 가능하도록 되어 주변기기 회사는 원하는 시점에 원하는 서비스를 언제든지 개발할 수 있고 로봇 개발사는 주변기기 인증과 배포 절차가 존재하지만 이는 전체 서비스의 인증과 배포가 아닌 권한을 준다면 개발사가 자유롭게 인증과 배포 권한을 갖는다. SI 개발자는 다양한 서비스를 기반으로 도메인 관련 서비스를 개발할 수 있게 되었고 최종 사용자는 다양한 서비스를 손쉽게 사용할 수 있게 되었다.

마이크로서비스 아키텍처 도입으로 협동 로봇 개발사의 잦은 소프트웨어 배포 부담을 줄여 주었고 주변기기 개발사는 로봇에 종속되지 않는 개발을 할 수 있고, 협동 로봇 생태계 확장이 용이해졌고, 클라우드 기반으로 물리적 파일이동을 하지 않아도 되도록 되었다. 결과적으로 컨텍스추얼 디자인을 통해 나타난 최종사용자의 불편함, 개발환경의 갈라파고스와, 클라우드 도입필요 3가지 중 이번 연구에서 다루지 않은 UI와 관련된 직관적인 프로그래밍을 평가할 수는 없지만 스킬로 도메인 지식기반 프로그래밍을 가능하게 한 것으로 본다면 모두 개선된 것을 확인 할 수 있었다.

연구과정 및 결과에 대한 협동 로봇과 관계자들은 “의미 있는 접근이고 다양한 소프트웨어 개발자들의 접근성을 열어줄 수 있는 것 같다", "실제 퍼포먼스가 어떤 지 리얼타임을 어느 정도 수준으로 지원하는지에 대한 실험 결과가 나타나지 않아 아쉽다" "다양한 실험적 클라우드 접근들이 외부에서 감지되고 있는데 구체적 접근 방법에 대해 제시하는 내용이 좋았다"와 같은 피드백을 주었다. 이를 종합해 보면 컨텍스추얼 디자인을 통한 문제점들의 제시와 마이크로서비스 아키텍처도입의 변화 모습에 대해 많은 공감을 했지만 명확한 성능에 대한 결과 제시가 부족하다고 평가할 수 있을 것 같다.

7. 결론

본 연구는 협동 로봇 프로그래밍의 효율성 증대를 위한 마이크로서비스 기반 플랫폼 설계를 목표로 진행된 연구과정에서 다양한 이해관계자가 얽혀 있는 협동 로봇 프로그래밍 생태계의 문제를 풀기 위해 컨텍스추얼 디자인을 통해 이해관계자들의 문제점을 발견하고 해결하기위한 비전을 제시했으며 비전을 기반으로 한 아키텍처를 제안하였고 아키텍처 컨셉 검증을 위한 프로토 타입 개발하였다.

이 연구의 기술적 성과는 협동 로봇 기반의 이기종 디바이스 프로그래밍을 마이크로서비스 기반으로 제공할 수 있다는 것과 도메인 지식을 활용한 매시업을 사용한 프로그래밍을 할 수 있는 가능성을 검증한 것이다. 이는 잦은 통합 배포의 리스크를 낮추고 주변기기나 SI 개발자들의 중복 개발의 문제를 해결가능성을 확인한 것이다. 마이크로서비스 아키텍처의 도입은 협동 로봇 프로그래밍의 이해관계자들의 개발 속도를 높여주고 효율성 증가로 이어져 스마트팩토리를 넘어 스마트 주방 스마트 카페 등의 서비스로 생태계 확장에의 확장 가능성을 높여 더욱 큰 협동 로봇 생태계를 구축할 수 있는 가능성을 확인한 것이라고 할 수 있다.

이후 연구에서는 UI분야를 포함하는 아키텍처 컨셉 검증과 협동 로봇이 적용가능한 어플리케이션을 적용하고 어플리케이션의 QoS 기준을 수립하여 기술 스택의 성능의 비교결과를 도출해 적용가능한 어플리케이션까지 구체적으로 제시될 수 있는 연구를 진행할 예정이다.

Acknowledgement

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2019R1A2C1087430).

8. 참고문헌

[1] Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., ... & Rosson, M. B. The state of the art in end-user software engineering. *ACM Computing Surveys (CSUR)*, *43*(3), 1-44 ,2011.

[2] Zhao, L., Loucopoulos, P., Kavakli, E., & Letsholo, K. J. User studies on end-user service composition: a literature review and a design framework. *ACM Transactions on the Web (TWEB)*, *13*(3), 1-46. 2019.

[3] Chen, G., Wang, P., Feng, B., Li, Y., & Liu, D. The framework design of smart factory in discrete manufacturing industry based on cyber-physical system. *International Journal of Computer Integrated Manufacturing*, *33*(1), 79-101.2020.

[4] Patel, D., & Mohammed, S. Cloud and Edge Computing for Developing Smart Factory Models using a iFogSim Wrapper: Transportation Management System (TMS) Case Study. 2020.

[5] Xia, C., Zhang, Y., Wang, L., Coleman, S., & Liu, Y. Microservice-based cloud robotics system for intelligent space. *Robotics and Autonomous Systems*, *110*, 139-150. 2018

[6] Beyer, H., & Holtzblatt, K. Contextual design. *interactions*, *6*(1), 32-42. 1999.

[7] Han, S.I , KR. Patent No. 10-2017-0015715 ,2019.

[8] Chen, Lianping. Microservices: Architecting for Continuous Delivery and DevOps,2018

[9] Richardson, Chris. “Microservice architecture pattern”

효율적인 웹 정보 수집을 위한 강화학습 기반 스케줄러 설계

정대욱 ○ 백종문

한국과학기술원

dujung@kaist.ac.kr, jbaik@kaist.ac.kr

Design of Reinforcement Learning-Based Scheduler for Efficient Web Crawling

Daeuk Jung○ Jongmoon Baik

KAIST

291, Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea

요 약

국내의 가격 비교 검색 서비스가 활성화되어 있지만, 상품 옵션을 포함한 상세 정보는 제공하지 않고 있다. 특히 신발 관련 사이즈별 재고와 옵션 가격을 검색하는데 있어서 상당한 시간과 노력이 필요한 상황이다. 이를 개선하기 위해, 각 쇼핑몰들의 상세 정보로부터 신발의 사이즈별 재고와 판매가를 주기적으로 수집하고, 이를 검색할 수 있는 정보화 시스템 구축에 대한 요구가 있다. 하지만 수천 가지 각기 다른 신발 모델에 등록된 수천 만개의 개별 사이트로부터 상세 정보를 주기적으로 수집하여 처리할 경우 상당한 트래픽 용량과 컴퓨팅 자원이 필요할 것으로 예상되어 그 실효성에 문제가 있다. 이에 이 논문에서는 상세 검색 서비스 품질을 상당히 높게 유지하면서도, 각 페이지별 수집 주기를 획기적으로 줄일 수 있는 강화학습 기반 스케줄러를 제안하고 있다. 이를 위해 최신 심층강화학습 모델인 IQN (Implicit Quantile Networks) 네트워크를 이용하여, 매 주기마다 페이지별 수집 여부를 결정하는 에이전트와 페이지 수집을 처리하는 작업 스케줄러를 도입하였다. 검증은 국내 온라인 쇼핑몰에서 판매되는 인기 신발 모델들의 상품 페이지를 매 주기마다 실제 수집된 상세 정보를 기초하여, 강화학습으로 수집된 데이터와 비교 진행하였다. 그 결과 강화학습을 이용한 작업 스케줄러가 높은 가격 비교 검색 품질을 유지하면서도 상당히 많은 양의 트래픽을 줄일 수 있음을 보이고 있다.

1. 서 론

인터넷과 스마트폰의 발전으로, 구매자들은 인터넷 검색을 적극적으로 활용하고 있다[1]. 최근 구매자 행동 패턴에 대한 리포트에 따르면 오프라인 매장에 방문하는 구매자들의 50% 이상이 검색 서비스를 통해서 최저가를 검색하고 구매 결정을 하는 것으로 보이고 있다 [2]. 이러한 이유로 판매자 들도 경쟁력 강화를 위해 온라인 검색 서비스를 활용하여 판매 전략을 수집 이용하고 있지만, 낮은 검색 품질로 상당한 어려움을 겪고 있어 이를 개선하고자 연구를 진행하게 되었다.

검색 서비스는 인터넷 URL에 대응되는 정보를 주기적으로 수집하고 분석하여, 그 내용을 사용자가 검색하여 이용할 수 있도록 하고 있다[3]. 그리고, 국내 주요 가격 비교 서비스들도 상품 정보의 최신성을 유지하기 위해 1~2시간 간격으로 상품 정보를 주기적으로 취합하여 검색 인덱싱에 이용하고 있다[4]. 다만, 상품 페이지의 구성을 살펴보면, 상품 기본 정보 외에 추가 옵션 구성을 통해서 실제 구매 여부와 판매가를 페이지 상세 내용 안에서 제공하고 있다. 이러한 옵션 구성 정보는 주로 동적 데이터로 구성되어 있으며 현재의 검색 서비스에서는 제공해주지 않는 데이터로 구성되어 있다. 이러한 이유로 현재의 검색 서비스는 페이지의 기본 정보만으로 검색 내용을 제공해 주고 있으며, 검색 결과의 내용이 실제 페이지에서 제공해 주는 정보와 상이한 문제점이 있다. 이를 해결하기 위해서는 상세 정보를 포함한 상품 페이지를 웹수집기(Crawler)로 직접 수집 후 활용할 수 있는 검색 서비스로 개선할 수 있을 것으로 보이지만 페이지 수와 변동 주기성으로 그 실효성에 시스템 구현의 문제가 있다.

1.1. 데이터 변화 주기의 문제

상품 페이지는 접속 URL를 통해서 접근할 수 있으며,

html 구조 데이터 안에 정적인 부분과 각 옵션별 가격 및 재고를 담고 있는 동적 데이터로 구성된다. 다만, 상품 페이지 내의 동적 데이터들은 판매자 관리 도구와 구매자의 구매 활동에 따라서 그 가격이나 재고수가 시간에 따라서 변동한다. 변화의 정도를 살펴보기 위해 약 1주일 동안 243개의 상품 페이지를 수집하면서 모니터링을 실시하였다. 페이지 내의 동적 데이터의 변동을 추적하면서, 각 시간대 별로 변동된 데이터를 포함하는 페이지의 개수를 측정해 본 결과 아래 표(**그림 1-1**)와 같이 조사됐다. 확인 결과 변화의 정도가 시간대나 기간별로 일정하지 않고 무작위 하게 변화하는 특성이 보인다.

Number of the changed contents (HeatMap/Page#243)

No	Hour	Week Day	Tue 8.4	Wed 8.5	Thu 8.6	Fri 8.7	Sat 8.8	Sun 8.9	Mon 8.10	Tue 8.11	Sub SUM
0	6:00		0	0	0	0	0	0	0	0	0
1	7:00		0	0	5	10	6	2	19	7	49
2	8:00		0	2	23	8	4	0	2	26	65
3	9:00	D	0	3	15	6	3	1	4	27	59
4	10:00	D	0	1	3	7	4	0	2	16	33
5	11:00	D	0	0	5	6	2	0	4	18	35
6	12:00	D	0	0	30	25	2	2	21	14	94
7	13:00	D	0	33	16	17	2	3	17	13	101
8	14:00	D	0	1	29	20	2	1	20	23	96
9	15:00	D	0	2	33	20	3	1	35	28	122
10	16:00	D	0	2	15	15	0	3	23	0	58
11	17:00	D	0	5	26	21	1	2	11	0	66
12	18:00	D	0	5	6	11	2	1	12	0	37
13	19:00		0	176	13	13	4	7	15	0	228
14	20:00		0	13	8	12	1	0	11	0	45
15	21:00		0	4	27	11	2	2	5	0	51
16	22:00		0	5	7	5	1	5	6	0	29
17	23:00		2	5	8	2	1	3	2	0	23
18	0:00		34	14	11	4	42	13	27	0	145
19	1:00		0	0	2	2	0	4	0	0	8
20	2:00		0	2	3	2	0	3	1	0	11
21	3:00		0	0	0	0	0	0	0	0	0
22	4:00		0	0	0	0	0	0	0	0	0
23	5:00		0	0	0	0	0	0	0	0	0
24	6:00		0	0	0	0	0	0	0	0	0
Sub SUM			36	273	285	217	82	53	237	172	1355

그림 1-1. 시간대별 컨텐츠 변경 개수

1.2. 데이터 처리 용량의 문제

국내 상품 비교 검색 서비스에 등록된, 신발 관련 전체 페이지 개수는 약 3천 7백만개 정도로 조사된다. 상품 페이지를 구성하는 html 데이터는 화면 구성을 위한 구조적 부분과 판매 조건과 관련된 동적 데이터가 결합된 형태로 구성된다. 상품 페이지의 기본 정보와 상세 정보를 수집하여 분석하는데 있어서, 웹 수집기(Crawler)가 처리해야할 평균적인 데이터의 크기는 약 250kb정도로 측정되어 진다. 여기에, 시간당 변화하는 데이터의 변화 주기를 고려하면 매 30분 간격으로 수집할 필요가 있다. 이러한 정보 수집 검색 서비스를 만들 경우, 예상되는 전체 데이터 처리 용량은 30분당 약 9.4TB 이 된다(**그림 1-2**). 이를 초당 처리 용량으로 계산할 경우 5GB이다. 일반적인 인터넷 서비스의 전송 속도가 100MB 정도 임을 감안할 때, 최소 50회선 이상이 필요하다. 이처럼 많은 양의 트래픽 처리에 대한 제한이 걸릴 것으로 예상된다.

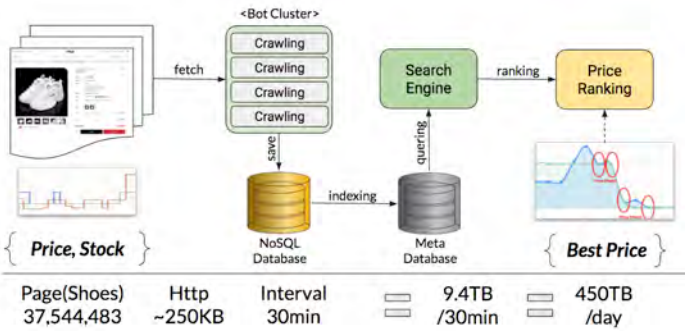


그림 1-2. 데이터 처리 용량 예상치

1.3. 데이터 처리 비용의 문제

관련하여 검색 시스템을 구축할 경우 예상되는 비용을 산정해 보면 아래 표(**그림1-3**)와 같다. 구성에 따라 최적화된 비용은 다소 차이가 있을 것으로 보이지만, AWS (Amazon Web Service) 클라우드 서비스를 이용하여 전체 신발 사이트를 주기적으로 수집할 수 있는 시스템을 예상 평가하였다. 페이지의 동적 정보 수집까지 고려하여, 각 페이지별 평균 수집 처리 시간은 약 3초 정도 걸리는 것으로 측정 되었다. 이를 동시 10개 쓰레드(Thread) 처가 가능한 최소 용량의 EC2기반 인스턴스를 이용하고, 전체 3천 7백만 개의 상품 페이지를 30분 내에 모두 수집 처리할 경우를 평가하였다. 수집 데이터의 인바운드 (InBound) 트래픽 비용은 없으며, 수집 요청에 대한 아웃바운드 (OutBound) 데이터와 인스턴스 가동 비용에 따른 예상되는 월간 최소 유지 비용은 약 4천만원 가까이 됨을 알 수 있다. 이러한 시스템을 유지할 경우 상당한 비용이 예상되며, 그 실효성에 상당한 문제가 있을 것으로 예상된다.

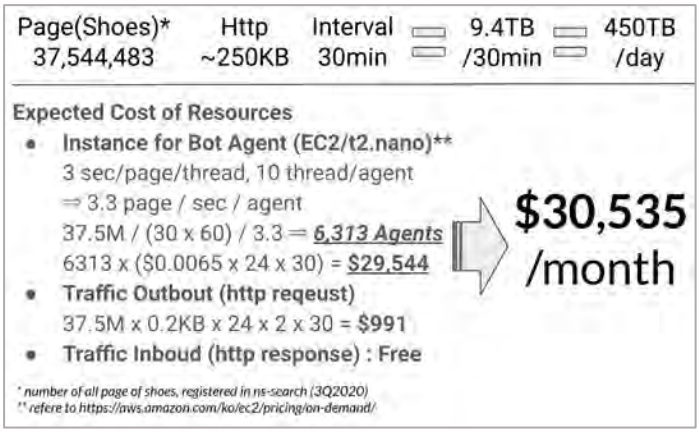


그림 1-3. 월간 데이터 처리 예상 비용

앞서 살펴본 이유로, 이러한 시스템을 구성하는 데에 있어서 데이터 처리 용량 문제와 이에 따른 유지 비용 문제로 효율적인 수집 정책이 문제 해결의 중요한 과제가 되며, 이를 해결하기 위한 연구의 목적이 있다.

2. 관련 연구

주요 검색 서비스들은 사이트의 최신 내용을 유지하는데 있어서 다양한 방법을 이용하고 있으며[5], 최신 정보를 유지하는데 있어서 페이지의 중요도와 변화 정도를 이용하여 효율적인 페이지 수집 방법을 제시하고 이용하고 있다[6,8,9]. 구글의 검색 엔진은 페이지의 중요도를 페이지 랭크(Page Rank) 알고리즘에 기반하여 측정하고, 이와 함께 페이지의 변경 이력 정보로부터 다음에 수집할 주기를 결정하여 검색 서비스를 하고 있다.[7]. HTML페이지의 변경 주기를 예측하는데 있어서 페이지의 구조적 정보와 함께 정적 분석을 활용하여 학습용 데이터를 추출해내고, 이를 인공지능 기술(Random Forest Cluster)을 이용하여 페이지의 변화 주기를 예측하고 다음 웹수집 주기를 결정 짓기도 한다[8]. 한편으로는, 강화 학습(RL) 모델을 이용하여 트래픽 가용 최대 용량(Bandwidth)을 고려하면서도 페이지의 최신을 유지하는 방법이 적용되기도 한다[10]. 하여, 관련 연구를 통해서 문제의 해결책을 고려해 보았다.

2.1. 페이지 중요도에 따른 수집 정책

구글은 페이지 검색 서비스를 위해서 많은 수의 페이지를 수집하여 검색에 이용하고 있다. 2010년도에 발표한 US 특허 정보(US8161033)에 따르면, 구글이 어떻게 페이지 수집에 대한 스케줄러를 이용하는지 알 수 있다 (**그림 2-1**). 페이지의 변화 주기가 높을 수록 더 자주 수집을 해야하는 것으로 예상할 수 있지만, 구글은 여기에 자체 페이지 랭킹 알고리즘에 따른 페이지별 중요도(페이지 랭킹) 정보를 추가 활용하고 있다[6]. 수집 점수(Crawl Score)는 이 페이지 랭크(Page Rank)의 제곱에 비례하도록 계산되어지며, 이 값이 일정한 기준치(Threshold)를 넘을 경우 다음 수집 대상이 된다. 페이지의 변화 정도와 함께 페이지 중요도에 더욱 민감하게 작용하는 스케줄러는 좋은 접근 방법으로 고려되어 진다. 하지만, 본 연구 대상인 상품 페이지의 동적 특성과 가격 비교 우위성에 적용하기에는 부족한 부분이 있어 보인다.

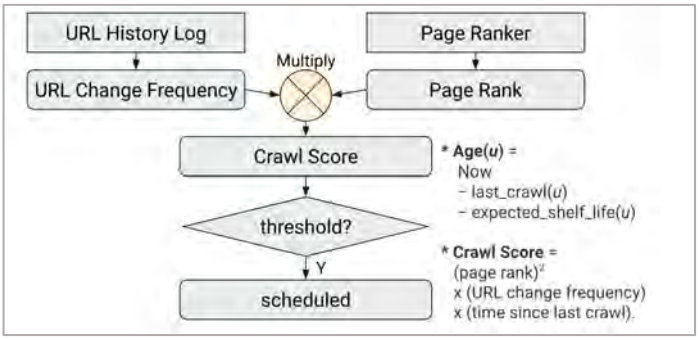


그림 2-1. 구글의 검색엔진 스케줄러 도식도

2.2. 페이지 변경 주기 예측

페이지의 동적 특성을 고려하여 변경 주기를 인공지능 기술을 이용하여 예측하려는 연구도 있다[7]. 페이지 URL로부터 초기 페이지의 구조적 정보를 포함하여 저장하고, 이후 수집된 페이지의 구조적 정보를 서로 비교하였다. 비교되어진 결과는 여러 특성 지표를 만들어 내고, 이를 랜덤 포레스트 (Random Forest) 알고리즘 이용하여 각 페이지별 변화 주기 량을 예측하였다 (**그림 2-2**). 그 결과 예측된 주기는 4시간에서 많게는 72시간으로 평균 정확도는 87.9%로 나왔다. 페이지의 동적 구성까지 고려하여 변화 주기를 예측하는 것은 좋은 접근 방법으로 보이지만, 앞서 보인 상품 페이지의 비 주기성 특성을 고려할 경우, 고정된 변화 주기를 찾는 방법에는 이 이용에 제한이 있을 것으로 판단된다

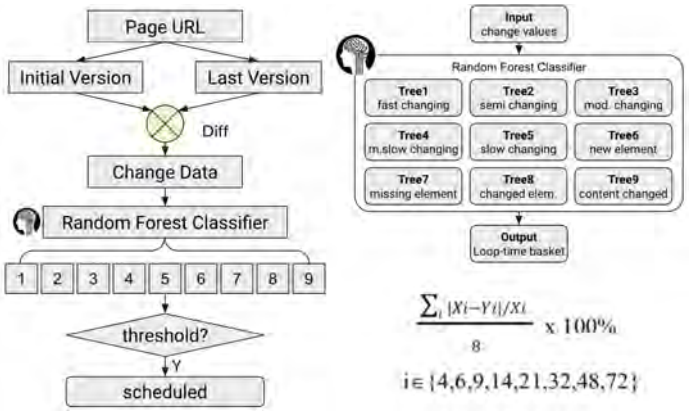


그림 2-2. 랜덤 포레스트 이용 변화 주기 예측

앞서 관련 연구들의 탐구를 통해 페이지의 최신성을 유지하는 효율적인 접근 방법을 고찰하여 보았다. 다만, 가격 비교 우위에 높은 검색 품질을 요구하는 시스템에 적용하기에는 그 한계가 예상되어 보이며, 이에 특화된 시스템을 제안하고자 한다. 하여, 이 논문은 신발 관련 상품 상세 검색 시스템을 구현함에 있어서 일정한 검색 품질(Search Quality)을 유지하면서 효율적인 웹 수집(Efficient Web Crawling) 을 수행하는 시스템에 대한 설계를 제안하고 있다.

3. 제안 시스템

본 논문에서는 상세 정보를 주기적으로 수집하여 검색 인덱싱에 이용하도록 하는 시스템을 구성하고, 효율적인 수집 주기를 결정하기 위해서 강화학습에 기반한 새로운 스케줄러를 제안한다. 웹 수집기(Web Crawler)를 통해서 수집할 내용은 페이지 내의 동적 데이터 내에 포함되어 있는 신발 사이즈 옵션별 판매가(Price)와 재고(Stock)수 정보가 포함된다. 다만

이러한 정보는 직접적인 웹수집을 통해서만 그 변동 여부를 알 수 있는 제약이 있다. 수집 여부를 판단하는 심층 강화 학습(DRL: Deep Reinforcement Learning) 모델은 단일 네트워크로 가장 높은 성능을 보였던 IQN(Implicit Quantile Networks) 네트워크를 이용하며, 강화 학습 기법을 통해서 학습되어 진다. 학습된 수집기는 매 주기마다 페이지별 주기 여부를 판단 결정하게 된다[11].

DRL(심층 강화 학습)은 환경 정보와 행위(action)에 따른 보상(reward)의 상호 관계로 모델이 학습하도록 하고 있다[12]. 딥마인드의 알파고로 그 우수성이 국내에 잘 알려지기도 하였다[13]. 본 논문에서 제안하는 강화학습기반 스케줄러의 성능은 실제 매 30분 간격으로 수집된 상세 정보와 스케줄러를 통해 수집 결정되어진 정보의 최신성 여부를 측정하여 확인하였다. 특히 관심 부분인 재고가 있는 상품 들에서 낮은 가격 순으로 상위 3위의 정보들의 최신성을 검증하였다.

논문은 전체 시스템과 구현 방법에 설명하고, IQN네트워크를 이용한 스케줄러의 구현과 이를 적용한 정보의 비교 검증으로 진행하였으며, 마지막 결론으로 논문의 끝을 맺는다.

3.1. 시스템 모델과 웹수집기

이 연구에서 제시하는 전체 시스템 아키텍처는 웹 수집과 검색 부분에, 추가적으로 RL (Reinforcement Learning)기반 스케줄러를 적용한 아래의 시스템 구성도(**그림 3-1**)와 같다.

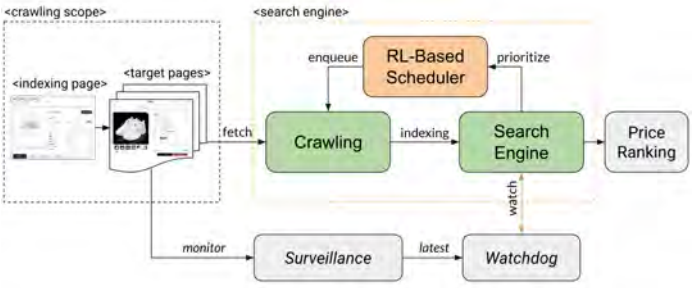


그림3-1. 시스템 구성도

각 상품 페이지는 고유 URL로 접속 가능하며, 현재 가격 비교 서비스를 통해서 대상 상품 페이지의 목록을 가져올 수 있다. 웹수집(Crawling)은 대상 접속 URL를 시작으로 페이지의 상세 정보를 수집 분석하고 서버에 보내게 된다. 수집된 상세 데이터 결과는 검색 엔진(Search Engine)에 색인화(Indexing)후 저장 관리된다. 이후 검색 질의문(Query)을 이용하여 신발의 사이즈 옵션별 재고와 가격 순으로 페이지의 검색 순위를 조회할 수 있다. 가격에 따른 페이지별 순위는 RL기반 스케줄러에서 페이지별 수집 여부를 판단하는 지표로 활용되며, 수집 여부에 따라 스케줄러는 웹 수집기의 작업 대기열에 수집할 것을 알려준다. 본 연구에서는 강화학습에 기반한 인공지능 스케줄러를 이용하고 있지만, 수집하지 않으면 변경 여부를 알 수 없기에 별도로 모니터링을 위한 감시(Surveillance) 부분을 추가로 이용한다. 감시 부분은 매 주기마다 페이지를 랜덤 샘플링 하면서 변경 정보를 추적하며, 제안 시스템의 검색 데이터와 비교하여 스케줄러에 의한 결과가 잘 이루어지는지 확인토록 하였다.

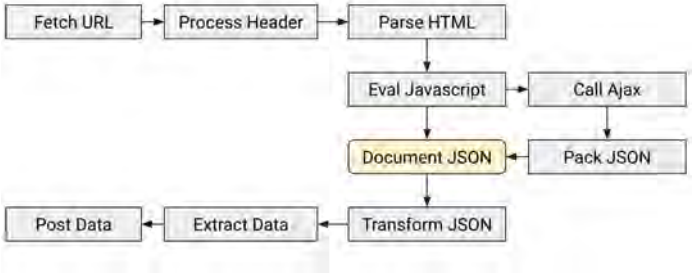


그림 3-2. 동적 데이터 수집 분석 절차

웹수집기(Crawling)은 동적 데이터를 포함한 상세 정보를 수집 분석한다. 온라인 상품 페이지는 인터넷 브라우저로 접속 가능한 각기 고유한 URL 주소가 있으며, 이 URL을 요청하면 HTML 형식의 문자열 데이터가 수신되며, 구조적 분석 및 내용 분석을 통해서 동적 데이터를 추가 수집하여 최종적으로 원하는 데이터를 JSON (JavaScript Object Notation) 형태로 추출하여 서버로 전송되어 진다 (**그림 3-2**). 신발 관련 상품 페이지는 기본 판매가를 포함한 기본 정보와 각 신발 사이즈별 실제 판매가와 재고 여부로 구성된 상세 정보로 구성 되어 있다. 국내/외 상품 검색 서비스들은 이들 상품 사이트로부터 기본 정보만을 주기적으로 수집하여 검색 서비스에 활용하고 있으며, 서비스 이용자가 검색 결과에서 얻어진 상품 페이지 URL 접속 링크를 통해서 직접 상세 내용을 확인하는 방법으로 제시하고 있다. 이를 개선하고자 각 상품 페이지를 직접 수집하는 웹수집기 부분을 만들고, 상세 내용의 신발 옵션 사이즈별 최종 판매가와 재고 여부를 저장/검색 하도록 구성하였다.

상세 페이지는 주로 동적데이터를 포함하고 있으며, JSON과 AJAX (Asynchronous Javascript And Xml) 통신을 통해서 옵션별 상세 정보를 추가로 요청하여 최종 가격과 재고가 포함되도록 하였으며, 상품 페이지 분석은 HTML DOM (Document Object Model) 분석기 (Parser)를 이용하여 동적 데이터 추출과 신발 옵션별 상세 데이터 형식으로 가공하도록 하였다. 최종 처리된 데이터는 JSON형식의 포맷으로 변환되어 서버의 수집 API에 전송하도록 하도록 한다. 수집 서버는 페이지별 수집 성공 여부와 함께, 각 옵션별 데이터를 분석하여 검색 엔진의 색인화 (Indexing)에 쓰일 데이터로 분리하여 저장하도록 하였다.

3.2. 심층강화진습(DRL) 기반 스케줄러

강화학습(Reinforcement Learning)은 행위자(Agent)가 환경(Environment)의 상태(State)에 따라서 선택한 행동(Action)에 따른 보상(Reward)을 이용하여 학습시키는 기계 학습의 한 방법이다[6] (**그림 3-3**). 환경과 서로 상호 작용하면서 얻어지는 기대 보상이 최대가 되도록 행동을 선택하도록 학습하는데, 그 중 심층 강화학습(Deep Reinforcement Learning)은 행동 결정 여부를 심층신경망(DNN: Deep Neural Network)을 이용한다[12].

이번 연구에서의 스케줄러는 이러한 강화학습 기반으로 페이지의 수집 여부를 판단하도록 한다. 에이전트(Agent)가 바라보는 환경은 이전 수집 여부에 따른 페이지의 변화와 신발 상세 데이터 내의 가격/재고의 상태값 들로 표현된다. 주기 시간(t)에 에이전트는 각기 페이지의 상태 정보를 이용하여 다음 주기(t+1)에서 수집 여부에 대한 행동을 선택하도록 하며, 그 결과는 참/거짓이 되도록 하였다. 수집이 결정되면 수집기의 작업 대기열에 수집 예약 되며, 다음 주기(t+1)안에 실제 데이터 수집을 진행하고 그 결과를 페이지별 상태에 반영하게 된다. 이러한 수집 여부 선택과 실제 수집 데이터의 변화에 따라서 보상이 주어지도록 구성하고, 매 주기마다 반복 수행하면서 에이전트가 적절할 행동을 학습할 수 있도록 구성하였다.

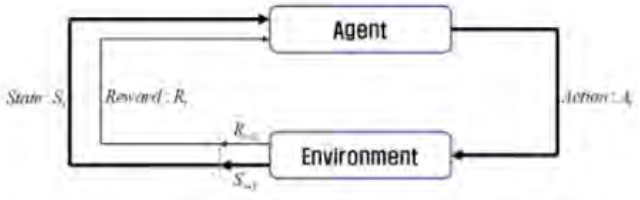


그림 3-3. 강화 학습 모델

DeepMind의 DQN(Deep Q-Networks)의 발표 이후 심층강화학습(DRL) 모델의 다양한 발전이 있었다. 본 논문에서는 이러한 심층강화학습(DRL) 모델 중에서도 단일 네트워크로는 가장 좋은 성능이 보였던 최신의 IQN (Implicit Quantile Networks) 심층 네트워크를 이용한다[11]. 상품 페이지의 상세 정보의 이력과 최신 정보를 저장하는 저장소를 구성하고, 이를 통해서 학습 환경에 필요한 페이지별 상태와 보상 그리고 액션을 저장하도록 하였다. 이러한 스케줄러는 웹 수집기의 수집 작업 대기열(Queue)과 결합시키고 매 주기 시간마다 작동하도록 구성하였다. 이러한 스케줄러의 시스템 연결 구성과 내부 구성 부분은 다음의 전체적인 스케줄러 구성도(**그림 3-4**)와 같다.

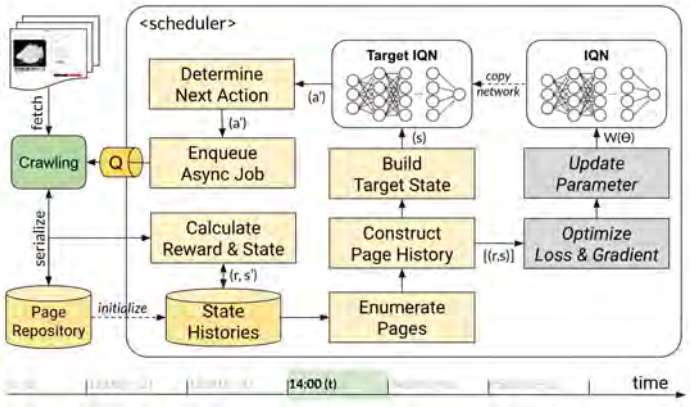


그림 3-4. 강화학습기반 스케줄러 구성도

우선, 스케줄러는 매 주기(30분) 마다 각 상품 페이지별로 다음 수집 정책을 결정 짓게 된다. 각 페이지는 개별 고유 URL이 있으며 이 URL을 통해서 수집된 데이터는 페이지 저장소(Repository)에 시간대 별로 저장된다. 최근 수집된 페이지 정보는 이전 정보와 비교하여 수집 여부에 따른 보상을 계산하거나, 현재 주기에 필요한 상태정보를 도출하게 된다. 변경 이력 정보는 상태 이력 저장소(State Histories)에 모두 기록되어 있으며 이후 수집 여부를 판단하거나, IQN 네트워크의 학습을 위한 데이터로 활용하게 된다. 일정한 주기로 에피소드(Episode) 이력이 충분히 쌓이면 IQN 네트워크를 개선하게 되고, 이를 페이지별 수집 여부를 판별하는 네트워크(Target IQN)에 복사되어 활용된다.

매 주기 활용되는 페이지별 상태 데이터는 아래 **표1**과 같이 정리된다. 페이지의 기본 상태 정보와 날짜 관련 데이터 그리고 신발 사이즈의 옵션별 상태값(가격, 재고)들로 구성된다. 페이지의 변동 여부는 실제 웹수집을 수행할 경우에만 알 수 있으며, 만일 수집을 안하고 지나갈 경우 이전 페이지 이력 정보를 그대로 재활용 하게 된다.

표 1 상태(State) 설정 정보 개요

항목	타입	설명
status	integer	페이지 상태 [ok, error, not-found]
hour	integer	시간 정보 (0~23)
half	integer	시간의 30분 전/후반 여부 (0~1)
week	integer	주간 정보 (0~6)
price*	float[]	각 사이즈별 가격 누적정규분포(CDF) 값.
stock*	integer[]	사이즈별 재고 지표 [0, 1, 2, 3]

* 전체 신발사이즈는 총 39가지 종류로 각 사이즈별 정규화된 값 이용. 사이즈 종류 (120,130,140,150,155,160,165,170,175,180,185,190,195,200,205,210,215,220,225,230,235,240,245,250,255,260,265,270,275,280,285,290,295).

신발 관련 상품 페이지의 경우 각 사이즈 옵션 선택에 따라 판매가와 재고 상태가 다를 수 있다. 각 판매가는 신발 모델과 재수 수준에 따라서 다양하며, 한화 기준으로 수만에서 수십만의 정수 값의 범위를 가진다. 실제로 같은 모델 그룹에 있는 상품 페이들의 각 사이즈 옵션 별 가격 분포를 조사해 보면, 사이즈별로 다른 분포를 보이고 있다(**그림 3-5**). 이러한 상세 페이지 정보로부터 IQN학습에 이용할 상태를 변환하여 이용할 경우, 상대적으로 큰 상품 판매가의 정수(Integer) 범위는 DNN (Deep Neural Network)에 직접 이용하기에는 무리가 있다. 또한 상품 모델에 따라서 가격 평균 분포도가 다르므로, 다양한 가격대의 신발 페이지를 일괄적으로 지원할 수 있는 방법으로 사이즈별 가격에 대한 정규화 처리가 필요하다.

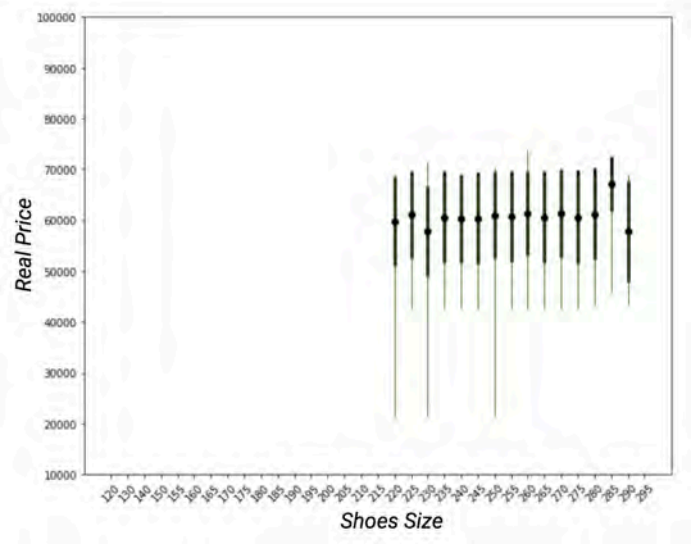


그림 3-5. 신발 사이즈별 가격 분포.

신발 모델 별로 다른 가격 분포를 가짐을 고려하여, 상품 가격은 데이터 정규화(Normalization) 과정을 적용하여 이용하였다. 정규화는 여러가지 목적으로 이용되지만, 이 논문에서는 같은 모델의 상품군 안에서 가격 비교를 통한 가격 랭킹(Price Ranking)이 중요하며, 변환된 결과 또한 가격 순으로 정렬할 경우 동일한 결과를 얻을 수 있어야 할 것으로 보인다.

이를 위해서 동일 모델의 비교 상품군 안에서 재고가 있는 신발 사이즈 옵션별 표준 정규 분포도를 먼저 구하였다. 그리고, 이를 각 페이지의 사이즈 옵션별 가격을 누적 분포 함수(CDF: Cumulative Distribution Function)를 이용하여 변환하여 이용하였다. 낮은 가격이 낮은 값을 유지하면서, 재고 없는 사이즈일 경우 1의 값이 되도록 적용하였다. 그 결과, 앞서 보인 신발 사이즈별 가격 분포는 아래의 차트(**그림3-6**)와 같이 가격 순으로 순위를 유지하면서도 데이터 분포가 [0,1] 사이의 실수값(Real)이 되도록 변환된다. 실제로 신발 260 사이즈의 페이지 검색 결과가 변환 전/후에도 같은 순위의 결과를 얻을 수 있음을 알 수 있다.

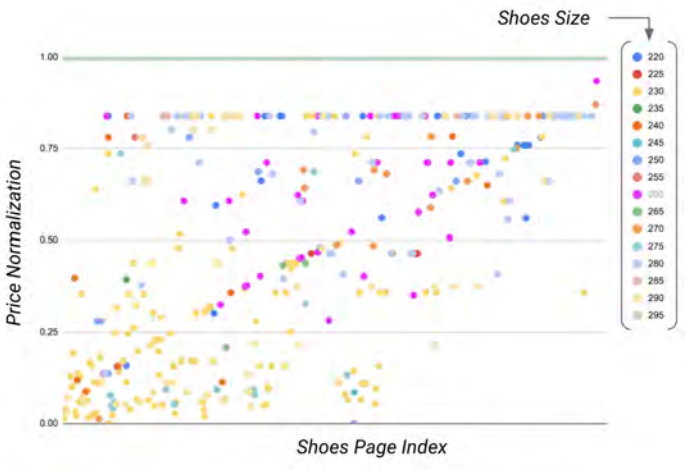


그림 3-6. 신발 사이즈별 변환된 가격 분포도

사용한 정규 분포 함수(PDF)와 누적 정규 분포 함수(CDF)는 각기 아래 수식(**그림 3-7, 그림 3-8**) 와 같은 식으로 구할 수 있다[14]. 다만, 판매 가격은 있지만 재고가 없을 경우 구매할 수 없는 경우이다. 또한 판매가 0인 경우도 특별한 이유가 없는 한 잘못된 경우로, 이러한 경우 변환된 값을 1로 설정하게 된다.

$$\frac{1}{\sqrt{2\pi}\,\sigma} exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$

그림 3-7. 정규 분포 함수

$$\frac{1}{2}\left[1+\mathrm{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]$$

그림 3-8. 누적 정규 분포 함수

재고(stock) 정보를 이용함에 있어서, 사이즈별 재고 지표로 변환하여 이용한다. 재고 값은 0이상의 정수값으로 수집되며, 이 값은 수집된 값의 포함 구간에 따라서 4가지 분류로 변환되어 진다. “0”은 재고 없음, “1”은 1~3까지의 낮은 재고, “2”는 4~8까지의 중간 정도, 그리고 마지막 “3”은 9 이상의 많은 재고 수준을 나타낸다. 다만, 페이지 수집 분석할 때 정확한 재고수를 확인할 수 없을 경우 구매가능하면 “1”, 구매 불가일 경우 “0”으로 대체하여 이용한다.

매 주기(t)에 계산되어 지는 보상은 이전 주기에서 에이전트가 선택한 수집 여부와 페이지별 상태 변화에 따라서 동적으로 주어지도록 하였다. 본 논문에서 이용한 각 보상 정책은 아래 **표2**와 같다. 기본적으로 앞선 주기(t-1)에서 결정된 수집 정책 결과를 확인하면서, 실제 수집 여부에 따른 페이지 내의 상세 정보 업데이트 변화에 따라 주어지는 보상이 달라지도록 설정하였다. 보상은 수집이 성공했으면서도 내용중 업데이트가 있을 경우 최대치(1점)가 되도록

구성하고, 변화가 없을 경우 최소치(-1점)이 되도록 설정하여, 에이전트가 소모적인 수집을 하는 것을 방지하고자 하였다.

표 2 보상(Reward) 설정 정보 개요

항목	값	설명
skipped	-0.001	수집 없이 다음 주기로 넘어갈 경우 기본값
changed	+1.0	수집 후 변화를 감지할 경우 보상값
unchanged	-1.0	수집 결과 변화가 없을 경우 패널티(Penalty)값

4. 검증 및 분석

데이터 표본으로 사용할 데이터는 실제 운영중인 개별 상품 페이지를 접속 URL단위로 추렸다. 어떠한 스케줄러의 연동 작동이 없는 것으로 생각하고, 2020년 10월과 11월간 약 4주 동안 1400개의 상품 페이지에 대해서 매 30분 간격으로 실제 상세 데이터를 수집하고 저장하였다. 이렇게 수집한 데이터의 전체적인 개요는 다음의 **표3** 와 같다.

표 3 표본 데이터 개요

항목	타입	설명
수집기간	4주	10/15 ~ 11/15일 기간
페이지수	1400	개별 온라인 페이지 개수
수집주기	30분	매 수집 주기의 간격
데이터수	2백만개	매 30분 주기의 4주간 데이터

이 데이터를 기반으로 강화학습 학습과 검증을 진행할 수 있는 환경을 구성하였다. 이는 에이전트가 매 주기 페이지별 수집 여부를 판단하면서 그에 따른 보상을 얻을 수 있는, 서로 상호 작용할 수 있는 학습 환경의 구성이다. 학습 환경은 초기 날짜를 시작으로 매 주기의 행위에 따라 시간 간격을 늘려가면서, 실제 수집된 데이터와 보상을 전달하도록 하였다. 각 주기별로 시간(t)이 진행되면서 학습에 필요한 에피소드 묶음이 준비가 되면서 IQN 네트워크가 각기 학습되게 된다. 모아진 에피소드는 단위 페이지 별로 변경 이력과 보상 그리고 행위 정보가 계속 쌓이도록 구성하였으며, 새로운 에피소드가 시작할 경우 다음의 대상 페이지를 무작위로 선택하도록 구성하였다. 이렇게 전체 페이지에 걸쳐 골고루 학습할 수 있도록 하였다.

이러한 환경에 기반하여 실험 데이터에 대해서 강화학습과 검증을 진행 하였고, 그 결과 아래의 그래프(**그림 4-1**) 와 같은 결과를 얻었다

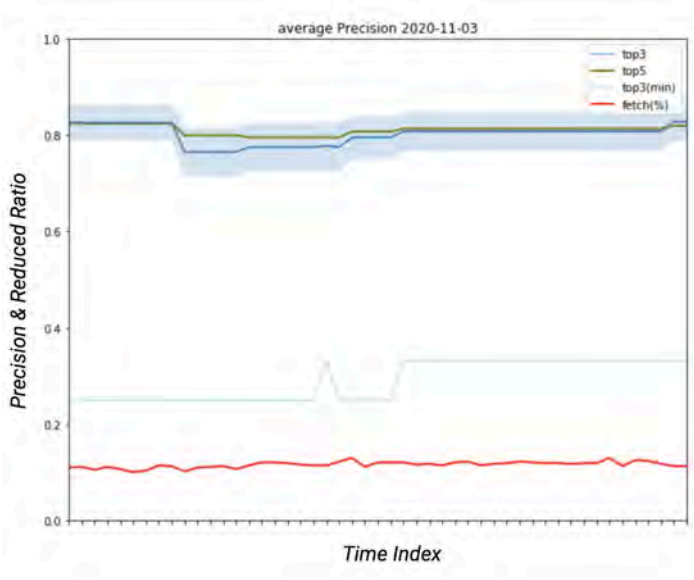


그림 4-1. Top3의 정밀도와 수집 감소율

그래프에서 검색 품질 (Search Quality)는 최저가 순으로 정렬할 경우 상위 3위(Top3)의 정밀도로 측정하였다. 즉 Top3 은 각기 동일 신발 사이즈 별로 구매 가능한 페이지들을 낮은 가격 순으로 정렬 했을 때 상위 3위 안에 유효한 페이지가 포함될 확률 값으로 정밀도(Precision)로 측정하였다. 정밀도는 주어진 결과에서 실제 순위 안에 드는 페이지의 개수로 측정할 수 있다. 이러한 시스템의 결과로 상위 3위 추천 페이지의 정밀도는 평균적으로 82%정도 나왔다. 이를 상위 3위의 검색 추천에 대한 기대되는 적증율(Hit-Ratio)을 계산할 경우 약 99.4% (=1-exp(0.18, 3)) 정도로 예상된다.

그래프에서 수집 효율성(Efficiency)는 매 30분 수집 주기에서 실제 수집 없이 지나간 경우의 수로 측정하였다. 즉, 수집 감소율(Fetch(%))은 최대 수집 주기 대비 절약한(Skipped) 수집 처리의 수로 평균 85% 이상의 감소율을 보이고 있다.

결과적으로 이 연구를 통해 상위 3위의 검색 품질이 99% 이면서도, 85% 이상 트래픽 절약이 가능한 상품 상세 검색 시스템을 제안하고 있다.

5. 결론 및 향후 과제

본 연구에서는 가격에 기반한 웹 페이지 상세 정보 검색 서비스에 활용할 수 있는, 효율적인 웹수집을 위한 강화학습 기반 스케줄러를 제안하였다. 웹수집(Web Crawling)은 페이지 내의 동적 데이터들로부터 각 옵션별 가격(Price)과 재고(Stock) 정보를 분석하여 저장한다. 각기 옵션별 가격은 같은 상품군 내에서 정규분포함수 (PDF)와 누적분포함수 (CDF)를 이용하여 정규화(Normalization) 처리 하였고, 재고 정보는 4가지 클래스로 정규화 처리 하였다. 그리고 스케줄러가

페이지 상태와 상호 작용하면서 학습할 수 있도록, 최신의 IQN(Implicit Quantile Networks) 네트워크 기반의 강화학습(RL) 환경을 구성하고, 페이지별 상태와 수집 여부에 따른 보상을 통해서 수집 정책이 자동으로 개선하도록 학습하였다. 학습된 스케줄러는 표본 데이터를 통해서 검증하였으며, 그 결과 최저 가격 순위 3위에 대한 적증율(Hit-Ratio)이 99%가 되면서도 수집되는 트래픽의 감소율이 85% 이상인 시스템이 가능함을 보였다. 이러한 트래픽 절감 성능은 전체 신발 제품 군의 실시간 수집 검색 시스템 구축에 있어서, 상당히 효율적이고 실용 가능한 시스템이 될 것으로 기대된다.

다만, 본 연구는 신발과 관련된 페이지의 수집 분석 및 효율적인 스케줄러 구성에 대한 내용으로 일반적인 상품 페이지에 적용하기에는 상세 옵션 구성의 다양한 차이성 문제로 일반적으로 이용하기에는 그 한계가 있다. 향후, 동적 데이터를 포함한 상품 페이지 특성에 적합한 학습 모델을 개선하여 전체 수집하는 상품 페이지의 대폭적인 확대를 통해서, 전체 상품 상세 비교 검색 서비스 활용에 크게 기여 할 것으로 기대된다.

6. 참고문헌

[1] Jung Yeon Sung, and KIM HYUN JIN. 2019. Analysis of Typology and Competition of Domestic Online Shopping - Focused on Naver shopping. JDMR, Vol.22, No.1, pp.35-46. DOI: 10.17961/jdmr.22.1.201902.35

[2] Vishal Mishra. February 14, 2018. Artificial Intelligence, Blog.https://sokrati.com/impact-online-marketing-offline-stores

[3] C. Castillo. 2005. Effective web crawling. ACM SIGIR Forum, 39 (1), 55-56, 2005.

[4] Naver Corp.. Guidance of Naver Shopping Engine Page https://saedu.naver.com/help/faq/ncc/view.nhn?faqSeq=389

[5] K. Desai, V. Devulapalli, S. Agrawal, P. Kathiria and A. Patel. Web Crawler : Review of Different Types of Web Crawler, Its Issues, Applications and Research Opportunities. International Journal of Advanced Research in Computer Sci, 8 (3), 1199-1202, 2017.

[6] Keith H. Randall. 2010. Scheduler for search engine crawler. US7725452B1.

[7] Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, MA, USA, 1998.

[8] Lakmal Meegahapola, Vijini Mallawaarachchi, Roshan Alwis, Eranga Nimalarathna, Dulani Meedeniya, and Sampath Jayarathna. 2018. Random Forest Classifier based Scheduler Optimization for Search Engine Web Crawlers. In Proceedings of the 2018 7th International Conference on Software and Computer Applications (ICSCA 2018). Association for Computing Machinery, New York, NY, USA, 285-289

[9] L. Meegahapola et al., Adaptive Technique for Web Page Change Detection using Multi-threaded Crawlers. In 7th international conference on Innovative Computing Technology, London, 2017.

[10] Andrey Kolobov, Yuval Peres, Cheng Lu, and Eric Horvitz. 2019. Staying up to Date with Online Content Changes Using Reinforcement Learning for Scheduling. Advances in Neural Information Processing Systems 32 (NeurIPS 2019)

[11] W. Dabney, G. Ostrovski, D. Silver, R. Munos, 2018, Implicit Quantile Networks for Distributional Reinforcement Learning, ICML 2018.

[12] R. S. Sutton, A. G. Barto, 1998, Reinforcement Learning: An Introduction, MIT press

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, Marc G. Bellemare, A. Graves, M. Riedmiller, Andreas K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Feb 2015, Human-level control through deep reinforcement learning, Nature, Vol. 518, No. , pp. 529-533

[14] Cumulative distribution function, Wikipedia, https://en.wikipedia.org/wiki/Cumulative_distribution_function

PAT 기반 반례로부터 테스트 시퀀스로의 체계적 변환*

B. Zelalem Mihret O, Lingjun Liu, 지은경, 배두환
한국과학기술원
{zelalem, riensha, ekjee, bae}@se.kaist.ac.kr

A Systematic Translation from PAT-based Counterexamples to Viable Test Cases*

B. Zelalem Mihret O, Lingjun Liu, Eunkyoung Jee, Doo-Hwan Bae
School of Computing, Korea Advanced Institute of Science and Technology (KAIST)

Abstract

Model checking approach has become popular as it provides the capabilities of exhaustively exploring the state space of the modeled system and generate counterexamples for properties specified over the model. The generated counterexamples can be used to derive test cases. However, counterexamples only show states, transitions and the values of their parameters. In addition, its semantics are also dependent on model specification languages and trace representation notations. In this paper, we present a focused test case generation approach from PAT model checker. The focus is driven by specific and putative attack behaviors. To this end, we devised test case specification rules to translate counterexamples to test cases. We demonstrate our approach by using air traffic control system (ATC) with a goal of minimizing safety risks due to cyberattacks during aircraft landing operation.

1. Introduction

The task of creating test cases manually is tedious, prone to error and time-consuming. There are several different approaches to address these problems. The approaches vary depending on several factors including the distinct characteristics of application domains, system development phases at which the test is planned to be conducted, abstraction level of the system, and intended goals to be achieved by the test. Due to such complexities, to a significantly larger extent, researches both in academia and industries focus on automatic generation of test cases from certain models of a system [1].

In this paper, we present our investigation and results obtained on focused test case generation using model checking technique. The focus is driven by specific and putative attack behaviors. For example, intercepting communication between systems and modifying leaked data is one of the common attack for several systems that rely on message communications. We model such attack behaviors as part of the system under study. The modeling includes permitting attacker behavior to access local variables of the system as one of the legitimate processes, and study the combined system behaviors for specific properties (such as starvation or collision properties). This is a focused approach in that the generated counterexamples show the system behavior specifically impacted by the attacker actions. Systematically we refine the counterexamples to formulate viable test cases.

*This research was partly supported by CybWin Project (No. 287808), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2019R1I1A1A01062946), and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2015-0-00250, (SW Star Lab) Software R&D for Model-based Analysis and Verification of Higher-order Large Complex System).

For the purpose of demonstrating the feasibility of our approach, we use a prototypical case from the aviation domain, air traffic control system (ATC). ATC is composed of several autonomous systems such as flight deck systems (FD), air traffic service provider (ATSP), control towers, different level human operators, etc. [2]. In this paper, we particularly focus on ATC constituent systems that play roles during landing operations of an aircraft. The system list includes: FD, surveillance data processing (SDP, ATC decision component), and short term conflict alert system (STCA). ATC system is huge and complex. We abstracted only a few functionalities in each constituent systems. For example, an aircraft represents request for landing permission, SDP checks the status of runway, current requests, and assign aircrafts to use runways, and STCA handles warning and alerts that may result in safety and security constraints. The three systems are networked, hence there is a risk of cyberattacks. We model the interaction behavior of the collaborating systems using CSP modeling language, and verify properties (for example detecting collision or starvation during landing) using PAT model checker [3].

PAT is a model checking tool. By design, it is not directly tailored for the purpose of test case generation. We demonstrate how to work around this by using test case specification rules that can be used to transform counterexamples into a set of actions and their sequences. With these constructing elements, we further refined it in a systematic way to formulate a concrete test case.

The remaining part of the paper is organized as follows. A survey on related works is presented in section 2. The overall approach is discussed in section 3. Section 4 presents illustrative example to show the application of the proposed approach, results obtained from the implementation, and shares lessoned learned in the process. Section 5 is devoted for conclusion and

future work.

2. Related work

There are several model-based test case generation methods [4]. The distinction between them lies in, among other things, the addressed modeling paradigm, application domains, the test generation methods, and the supported coverage criteria. Model checking is one of the techniques considered for test case generation [5] [6].

Despite the similarities in the technique, which is model checking, there is major difference in the way we consider input models for the model checking process. Specifically, our approach is different in that we consider the attacker behavior as input model for the model checking process. This gives chances to generate focused test cases that can address actual concerns. In this regard, we have not seen a related work that tries to exploit putative attack behavior as input model. Thus, the related works discussed below show model checking technique in general for test case generation.

S. Mohalik et al. [7] use the model checking technique to generate automatic test sequences. A test sequence represents paths in the system transition model. This sequences are used to conduct the tests. A. Armando et al. [6] deal with generation of putative attack from model checker and automatic testing on real implementations for security protocols using test execution engine. Putative attack in this work is used to represent selected and refined counterexamples based on domain expert knowledge from the counterexample pool.

3. Overall Approach

The overall approach is used to show major concepts and how the concepts are related in the processes of generating test cases from model checking. As shown in Fig. 1, an attacker behavior is considered as input model for the model checking process in our approach. This makes the approach more focused to the specific properties modeled in the attacker behavior. Major components of the overall approach are discussed as follows.

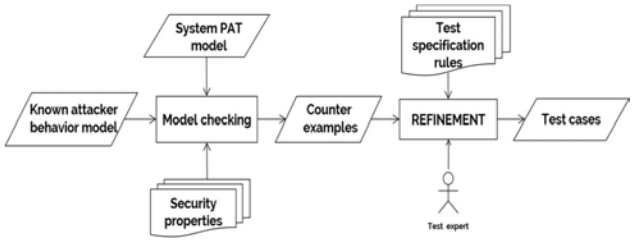


Figure 1. Overall approach

3.1. System PAT Model

PAT system model is a representation of a system behavior using a dedicated module in PAT. We use CSP module for modeling the system behavior as we are dealing with concurrent systems. The basic components of such modeling module are set of states, transitions between states, and rules of transitions. At run time, the operational semantics of the PAT model translates the behavior of the model into LTS (Labeled Transition System) which can be automatically explored by the verification algorithm [3]. One of the advantages of PAT is that it is designed with features that facilitate effective incorporation of domain knowledge with formal verification [8]. For example, complex computation and domain specific knowledge can be designed separately by using a standard programing language (for example C#) and can be integrated into the model checking process.

3.2. Putative Attack Behavior Model

Recently, information about cyberattacks including their techniques, targets and patterns are being identified and organized. For example, in the air traffic management (ATM) domain studies have identified putative cyberattacks (common to similar application domain [9]). For feasibility study, we investigate a specific and putative attack type in ATC domain.

3.3. Counterexample

A counterexample is a verification engine output that gives the simulation run leading to the state where a specified property is violated. PAT model checker has a verification engine that invokes a procedure to generate a Buchi automaton that corresponds to the negation of LTL property specification. Then the a Buchi automaton is composed with the internal model so as to determine whether the LTL formula is true for all system execution [3]. A counterexample, therefore, is generated when the search returns false.

3.4. Test Case Specification Rules

Basically a test case has components that describe inputs, actions, expected responses and explicit step by step instructions in order to determine if a system correctly deliver certain functions. In this regard, the test case specification rules we propose aims at identifying these constructs from counterexamples. The test case specification rules are set of rules to translate and refine counterexamples to test cases.

Counterexamples contains sequenced set of events that have an initial event and end events. The end event may form a loop i.e. contain more than one event. We proposed two phased step by step test case specification rules, presented as follows.

3.4.1. Test case data extraction from counterexamples

This phase is used to identify relevant events for test case synthesis, and distinguish attacker event and system responses.

(1) Step 1: Start from system event that happened immediately before the first attacker event in the counterexample, label this event as START. If the counterexample begins with an attacker event, take the first attacker event as START.

(2) Step 2: Record variables and their values both found in the START and the next event

(3) Step 3: Trace the counterexample forward skipping all system events until (3.1) a system event appears just before an attacker event, or (3.2) attacker event appears.

(4) Step 4: If (3.1) is the case from step 3, add to the record the newly discovered system event and the next attacker event parameters and their values, then repeat step 3. If (3.2) is the case from step 3, add to the record variables and their values of the attacker event.

(5) Step 5: Repeat step 3 and 4 until there is no more event to explore, or an event or set of events that create loops. Label the last event or set of events as END.

Between START and END events, collect all variables (local and global) that show change in their values at least once. These variable can be used as test inputs both for attacker events and system responses. All other variables and their values will be used to maintain sequence of events.

Fig. 2 shows counterexample system and attacker event transitions. The transition from START to END should satisfy the following two conditions to be considered for test case specification. First, a minimum one transition should exist from system event to attacker event, if START is a system event. Similarly, a minimum one transition from attacker event to system event if START is an attacker event. Second, a specific END event can be at system event or attacker event.

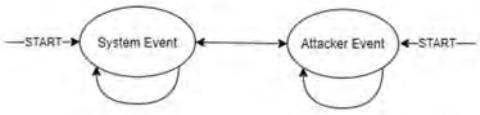


Figure 2. Counterexample event transition diagram

3.4.2. Test case synthesis

Test case synthesis systematically analyze the test case data record to drive test cases. The synthesis decides preconditions, test actions, test inputs and expected responses. It analyses the test data record based on the following steps:

(1) If START is an attack event, no precondition required to start test, otherwise the starting system event will be precondition to start test.

(2) Event's values that change at least once between START and END is used to construct test inputs. The END event represents the test case result.

A summarized form of a test case extracted from a counterexample is shown as follows.

Table 1.Test case components description

Counterexample data record	<current-event>
Precondition	Global or local variables and their values
Test inputs	Parameter values that have been changed if <current-event> is attacker event
System response	System state values if system event follows from <current-event> If <current-event> is not END, then it will be precondition for next event

4. Case study

To illustrate the feasibility of our proposed approach, we selected a small, but complete and non-trivial practical scenario from ATC. Ensuring safety during landing operations is one of the critical concerns in ATC. A test case that can be used to check that collaborating constituent systems properly function together and achieve safety goals such as no collision or starvation is one of the demanding requirements.

4.1. System PAT model

The following code snippets is taken from CSP models representing the three constituent systems: FD, SDP, and STCA. The model represent mainly processes and decisions while collaborating each other.

```
FD(i) = [rqstChnn[i] == notmade && rspnsChnn[i] !=
granted && ntfChnn[i] ==
ready]AIRCRAFT_makeRQST.i{ rqstChnn[i] = rqst}->
FD(i) [] [(ntfChnn[i] == ready || ntfChnn[i] ==
released) && alertsignal[i] == GREEN &&
rqstChnn[i] !=
notmade ]AIRCRAFT_landing.i{ ntfChnn[i] =  inaction}-
> FD(i) ...
SDP() = [state == listening && flg ==
DOWN ]ATC_checkrequest{var i = 0; var rqcntr = 0; var
gcntr = 0;  while(i < numberOfAirCraft){
if(rspnsChnn[i] == granted ){ gcntr = gcntr + 1;};
i++}; i = 0; while(i < numberOfAirCraft){
if(ntfChnn[i] == released){rspnsChnn[i] = ...
undecided };  i++}; ...
STCA() = [flg == UP && stat == 1]STCA_notifyalert{var
i = 0;  while(i < numberOfAirCraft){ if(rspnsChnn[i]
== granted){ alertsignal[i] = GREEN; stat = 0}
else{alertsignal[i] = RED}; i++};} -> STCA()
[] [stat == -1]STCA_resetalert{var i = 0; var cntr =
0; ...
```

As stated in the introduction and overall approach section, our goal is a focused test case generation driven by specific and putative attack. We model an intruder by considering a putative attack behavior in ATC domain. Communication links between collaborating system such as STCA and aircraft system are targets for cybersecurity attack. For this feasibility study, we consider an eavesdrops attack that aims at altering notification messages. A snippet showing the intruder behavior model is included as follow.

```
INTRDR() = [force == 0 && flg == UP]ATTCK_listen{var
i = 1; CNTR = 0;
if(alertsignal[0] == RED ){CNTR = 1};
if(CNTR == 1) {force = 1} else {force = 2}} ->
INTRDR() // evasdrope
[][force == 1]ATTCK_mdfy{alertsignal[0] = GREEN;}->
Skip();...
```

4.2. Property verification

One of the important safety properties in ATC domain, particularly during landing operation, is guarantying non-existence of collision i.e. more than one aircraft should not be landing on a run way simultaneously. We specified this property as follows for three FDs requesting for landing.

```
// check collision problem
#define collision (ntfChnn[0] == inaction &&
ntfChnn[1] == inaction ) ||(ntfChnn[0] == inaction &&
ntfChnn[2] == inaction)||(ntfChnn[2] == inaction &&
ntfChnn[1] == inaction);
```

4.3. Test case generation

PAT supports multiple verification options including first witness trace using DFS, shortest witness trace using BFS, and others [3]. PAT generates the counterexample events along with the variables and their values. The verification result from PAT model checker is shown in Fig 3.

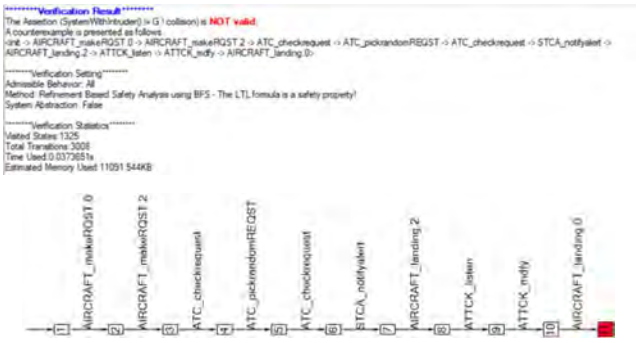


Figure 3. PAT model checker counterexample

By following the test case specification rules (test data extraction steps), we populate test case data record. The test synthesis process, finally, produces the test case. Table 2. shows the test case generated from the counterexample.

Table 2. Generated test case

Counterexample data	Precondition	Test inputs	System response
AIRCRAFT_landing.2	ntfChnn=[ready, ready, inaction]; alertsignal=[RED, RED, GREEN]; rspnsChnn=[undecided, undecided, granted]; rqstChnn=[rqst, notmade, rqst];	not applicable	none
ATTCK_listen	none	none	not applicable
ATTCK_mdfy	none	alertsignal=[GREEN, -, -];	not applicable
AIRCRAFT_landing.0	none	not applicable	ntfChnn=[inaction, ready, inaction]; alertsignal=[GREEN, RED, GREEN]; rspnsChnn=[undecided, undecided, granted]; rqstChnn=[rqst, notmade, rqst];

4.4. Result analysis

Using the generated counterexamples and the test case specification rules described, we showed how we develop a test case for aircraft landing operation. The two most popular challenges of using model checking technique for test case generation are (1) the quality of a model checking result is entirely dependent on the quality of input models, hence test case quality inherits similar limitations by association, and (2) counterexample generated based on test cases don't always map to real case concerns. This is mainly because counterexamples are results of exhaustive search for unsatisfied conditions in the state space without any bound as such for soundness or closeness check with respect to actual concerns. Our approach has constituents to attest for real case specific attack concerns because putative attack behaviors are constructed from real case reports.

Despite an attacker behavior is introduced into the system model, there is no direct way to make conclusion about which properties of the system can be violated. Therefore, the counterexamples can reveal what can go wrong even without a putative attack included in the model. In addition to this, a test case expert can use the generated test cases for a focused vulnerability analysis.

5. Conclusion and Future work

A viable test case can be synthesized systematically from counterexamples generated using PAT model checker. In this paper, we present a feasibility study on the possibility of translating PAT-based counterexamples into viable test cases that can be used for focused system behavior analysis, particularly attacker driven. Despite the challenges and limitations that exist in the model checking approach in general, and the modeling of an attacker behavior in particular; it can be seen that viable test cases can be synthesized that will reduce test expert's effort to convert it into a concrete test cases. Automating the translation process is our future work. This work can be extended to address vulnerability analysis of complex systems, and also runtime monitoring using the counterexamples as test sequences.

References

[1] S. Anand, E. K. Burke, T. Y. Chen, J. Clark, M. B. Cohen, W. Grieskamp, M. Harman, M. J. Harrold and P. McMinn, "An orchestrated survey of methodologies for automated software test case generation," in *Journal of Systems and Software*, 2013.

[2] G. Carrozza, M. Faella, F. Fucci, R. Pietrantuono and S. Russo, "Engineering air traffic control systems with a model-driven approach," in *IEEE software*, 2013.

[3] J. Sun, Y. Liu, J. S. Dong and J. Pang, "PAT: Towards flexible verification under fairness," in *Springer*, 2009.

[4] W. Li, F. L. Gall and N. Spaseski, "A Survey on Model-Based Testing Tools for Test Case Generation," in *Springer International Publishing*, 2018.

[5] O. Tkachuk, M. B. Dwyer and C. S. Pasareanu, "Automated environment generation for software model checking," in *IEEE*, 18th IEEE International Conference on Automated Software Engineering, 2003. Proceedings, 2003.

[6] A. Armando, G. Pellegrino, R. Carbone, A. Merlo and D. Balzarotti, "From model-checking to automated testing of security protocols: Bridging the gap," in *International Conference on Tests and Proofs*, Springer, 2012.

[7] S. Mohalik, A. A. Gadkari, A. Yeolekar, K. Shashidhar and S. Ramesh, "Automatic test case generation from Simulink/Stateflow models using model checking," in *Software Testing, Verification and Reliability*, Wiley Online Library, 2014.

[8] Hoare and CAR, "Communicating Sequential Processes Prentice Hall Int." London, 1985.

[9] E. P. Enoiu, A. Causevic, T. J. Ostrand, E. J. Weyuker, D. Sundmark and P. Pettersson, "Automated test generation using model checking: an industrial evaluation," in *International Journal on Software Tools for Technology Transfer*, Springer, 2016.

[10] M. E. Ruse, "Model checking techniques for vulnerability analysis of Web applications," 2013.

[11] E. Harison and N. Zaidenberg, "Survey of Cyber Threats in Air Traffic Control and Aircraft Communications Systems," in *Springer*, 2018.

[12] D. Beyer, A. J. Chlipala, T. A. Henzinger, R. Jhala and R. Majumdar, "Generating tests from counterexamples," in *IEEE*, 2004.

CCTV 영상 무결성 지원을 위한 대용량 이미지 블록체인 저장 방안

김태영 ○, 홍준기, 송성한, 강민구, 김순태

전북대학교 소프트웨어공학과

{rlaxodud1200, rlwns012, 201314106, 201314090, stkim} @jbnu.ac.kr

Large-capacity Image Blockchain Storage Approach for CCTV Video Integrity

Taeyoung Kim○ , Joongi Hong, Seounghan Song, Mingu Kang , Suntae Kim

Dept. of Software Engineering, Jeonbuk National University

요 약

본 연구에서는 사회의 많은 범죄를 해결하기 위해 많이 사용되는 CCTV의 보안관리 부실과 내외부의 악의적 해킹공격으로 인한 영상정보 조작을 방지하기 위해 사설 블록체인을 기반으로 대용량 이미지 블록체인의 무결성을 지원하기 위한 방안을 제시한다. 첫 째로 Hyperledger Fabric을 기반으로 사설 블록체인을 구축하고 영상의 원본을 저장하기 위해 IPFS를 블록체인과 함께 운영한다. 다음으로 대용량 영상의 트랜잭션 최적화를 위해 영상의 I-Frame을 추출 후 이미지용 해쉬 알고리즘과 기존의 해쉬 알고리즘을 조합하여 Hash화를 수행한다. 그 후 스마트 컨트랙트와 IPFS를 통해 비교하려는 영상의 조작 여부 확인 및 조작 된 부분을 알려주어 영상의 무결성 및 영상 조작의 설명을 가능하게 한다.

1. 서 론

사회의 다양성과 복잡성으로 살인이나 강간과 같은 다양한 범죄들이 증가하고 수법도 극렬해지고 있는 추세이다. 이를 해결하기 위하여 많은 기관들은 CCTV를 이용하여 공공안전에 위협적인 문제를 해결하고자 하고 있으며 범죄의 핵심 증거로 사용되고 있다[1,2,3,4]. 하지만, CCTV 영상정보에 대한 보안관리 부실과 내외부의 악의적 사용자들의 해킹 공격에 의한 영상정보 조작 및 사고가 발생되어 기술의 발달에도 불구하고 범죄율은 지속적으로 증가하고 있으며 특히 CCTV에 찍힌 영상 데이터의 무결성이 위협받고 있다.

데이터의 무결성을 해결하기 위해 블록체인을 기반으로 데이터의 무결성을 보장하는 방법이 있다. 블록체인은 중앙 집중 형 시스템의 데이터 무결성 문제점을 해결하기 위해 P2P와 같은 분산 네트워크를 기반으로 블록이라는 데이터 구조에 데이터를 입력하고 모두가 공유하는 분산 데이터베이스 시스템이다. 현재, 이러한 데이터의 무결성을 보장해주는 블록체인 기술을 통해 CCTV의 영상 무결성을 보장하기 위한 연구들이 존재한다.

Bela Gipp[5]는 영상정보와 Timestamp를 공용 블록체인 플랫폼인 비트코인에 저장하여 특정 시점의 영상의 조작을 방지한다. Mingda Liu[6]는 중국 대학교의 CCTV 영상 조작 방지를 위해 사설 블록체인을 도입하여 사용하고 있으며 M.Kerr[7]의 경우 동일하게 사설 블록체인을 사용하고 있지만 영상에 Water Mark를 사용하여 영상에 추가적으로 영상의 무결성을 지원하고 있다. 하지만, 공용 블록체인을 사용하여 영상의 무결성을 보장해주는 연구들의 경우 Privacy에 민감한 영상 정보가 저장되는 문제가 있으며 추가적으로 코인을 지불해야 데이터가 저장이 되는 문제가 있다. 또한, 기존의 사설 블록체인을 기반으로 한 연구들의 경우 스마트 컨트랙트를 제공하지 않아 특정 CCTV 영상의 조회 등의 기능의 부재가 존재하며 원본 영상을 통해 영상이 어떻게 조작이 되었는지에 대한 설명이 불가하다. 따라서, 본 논문에서는 이러한 문제들을 해결하고 영상의 무결성 보장 및 조작 방지를 위해 블록체인을 기반으로 대용량의 영상 정보를 관리하기 위한 접근방안을 제안한다.

CCTV 영상 조작 방지를 위해 대표적인 사설 블록체인인 HyperLedger Fabric과 영상의 원본을 분산 저장하기 위한 IPFS를 사용하여 CCTV 무결성 지원 네트워크를 구성한다. 또한 대용량 CCTV 영상의 고속 트랜잭션을 블록체인에 저장하기 위해 영상으로부터 I-Frame을 추출하고 이를 다중의 Hash와 조합하여

트랜잭션 발행량을 최적화 한다. 마지막으로 영상의 조작 조회 서비스를 지원하기 위해 Hyperledger Fabric의 Chaincode를 활용하여 스마트 컨트랙트를 구현한다.

2. CCTV 영상 무결성 지원을 위한 대용량 이미지 블록체인 지원 방안

다음의 그림 1은 CCTV 영상의 무결성 지원을 위한 대용량 이미지 블록체인 플랫폼의 구조를 보여주고 있다. 우선 CCTV에 영상이 저장된 영상 기록장치로부터 영상 정보를 받아 DApp으로부터 I-frame을 추출한다. 다음으로 해당 I-frame들의 원본 데이터는 IPFS에 저장한 후 각 I-Frame을 Hash화 한다. Hash화 된 각 I-frame들은 여러 Hash 조합으로 구성된 Merkle Tree에 저장되어 트랜잭션으로 변환되며 이를 사전에 배포된 스마트 컨트랙트를 통해 블록에 저장 후 각 노드에 Broadcast를 통해 해당 영상의 트랜잭션 값을 공유하도록 하여 무결성을 지원하게 한다. 마지막으로 영상의 조작 검증을 위하여 사설 네트워크의 스마트 컨트랙트를 통해 각 I-frame의 해쉬값이 블록체인의 해당 I-frame 해쉬값에 있는지 확인하고 IPFS로부터 해당 원본 이미지를 가져와 이 둘을 비교하여 조작 되었는지 확인한다.

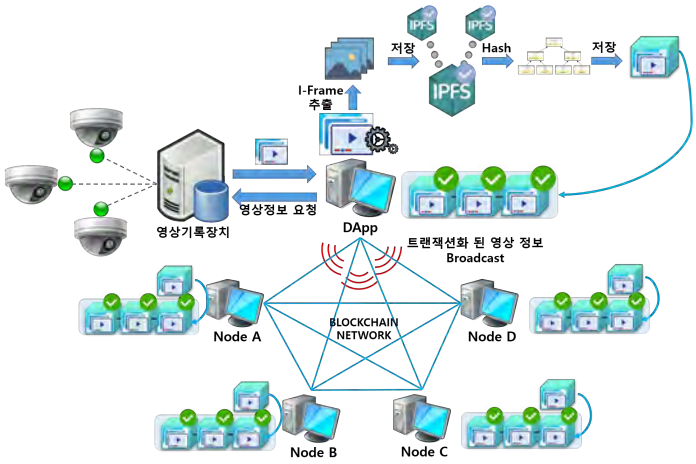


그림 1. CCTV 영상 무결성 지원을 위한 대용량 이미지 블록체인 지원 플랫폼

2-1. IPFS 기반 영상 무결성 CCTV 사설 네트워크 운용 환경 구성

영상 무결성 CCTV 사설 네트워크를 운용하기 위해 우리는 대표적인 사설 네트워크 지원 플랫폼인 Hyperledger Fabric을 사용하였다. 아래의 표1은 Hyperledger Fabric을 기반으로 구성한 컴포넌트를 보여준다. Orderer는 블록을 만들고 순서에 따라 다른 노드에 전파하는 역할을 한다. 즉 다른 노드가 동작하지 않을 경우 동작이 가능하게 하기 위해 5개를 사용하였다. 또한, 2개의 Endorser를 배치하여 영상 트랜잭션의 검증을 하도록 하였으며 지자체 기관과 기업 그리고 내부 기관 등의 3개 그룹을 구성하기 위해 3개의 CA와 Commiter를 구성한다.

VMS로부터 영상 핵심 정보를 추출하여 안전하게 보관하기 위해 각 노드마다 IPFS를 설치한다. IPFS(InterPlanetary File System)는 peer-to-peer 기반의 분산 파일 시스템으로써 사용자에게 DDos나 해킹으로부터 안전하게 파일을 공유하도록 지원한다[8]. 이러한 특성을 기반으로 우리는 대용량 영상을 IPFS에 저장 및 영상에 대한 조작 검증을 위해 IPFS를 블록체인과 같이 배치한다.

Component	# of Component
Orderer	5
Endorser	2
Commiter	3
Certificate Autority(CA)	3

표 1. Hyperledger기반 CCTV 사설 네트워크 노드 구성

2-2. CCTV 영상 Hash화 및 트랜잭션 최적화

CCTV 영상은 많은 데이터양을 보유하고 있어 이를 블록체인에 전 부 저장하여 모든 노드가 공유하게 되면 데이터 용량과 처리속도에서 성능의 저하가 올 수 있다. 다음의 표 2는 영상 해상도에 따른 1분당 영상의 File Size를 보여준다. Full HD의 (30FPS)의 경우 1분당 124mb의 크기를 가지며 UHD의 경우 348mb를 가지고 있다. 그리고 일반적으로 대부분의 CCTV는 Full HD(30FPS)의 해상도를 가지고 있다. 만약 영상기록장치가 CCTV 영상을 10분 단위로 저장 된다고 하면 평균적으로 1.24GB를 가지게 된다. 즉 이를 블록체인에 그대로 저장하게 될 경우, 트랜잭션마다 1.24GB를 가지게 되며 이에 따른 처리 부하속도가 증가하게 된다.

따라서 우리는 우선 영상의 핵심 정보가 되는 I-Frame들을 우선적으로 추출한다. 그 후 해당 I-Frame을 이미지 해쉬 알고리즘을 통해 해쉬화 하였으며 각 I-frame의 해쉬값은 Merkle Tree에서 Sha-256 해쉬 알고리즘과 조합 후 전체 영상에 대한 하나의 트랜잭션으로 생성하여 해당 트랜잭션 정보를 블록체인의 스마트 컨트랙트에 저장하여 블록체인의 저장되는 양을 줄인다. 또한, 각 I-frame은 IPFS에 저장하여 비교할 영상의 I-frame들과 비교 시 사용된다.

Video Resolution	File Size
Full High Definition (30FPS)	124mb
Full High Definition (60FPS)	205mb

Quad High Definition (QHD)	180mb
Ultra High Definition (UHD)	348mb

표 2. 1분당 영상 해상도에 따른 파일 크기

2-3. 스마트 컨트랙트를 통한 CCTV 영상 조작 방지 및 조작 설명

이전 단계에서 나온 CCTV 영상의 트랜잭션 값을 기반으로 우리는 아래 그림2. 와 같이 CCTV 영상 조작에 대한 설명을 표현하고 있다. 먼저 영상기록장치로부터 DApp을 통해 특정 영상의 조작 검증을 요청하게 되면 Restful API를 통해 특정 영상의 해쉬화를 수행 후 해당 해쉬 값을 블록체인에서 스마트 컨트랙트를 통해 해당 영상이 블록체인에 저장되어 있는지 확인한다. 만약 저장이 되어 있다면 영상의 각 I-frame을 해당 해쉬 값을 통해 IPFS에서 가져온 후 원본과 비교하려는 영상의 I-frame의 해쉬 값과 비교하여 영상의 조작 구간을 파악 후 조작된 구간을 알려준다.

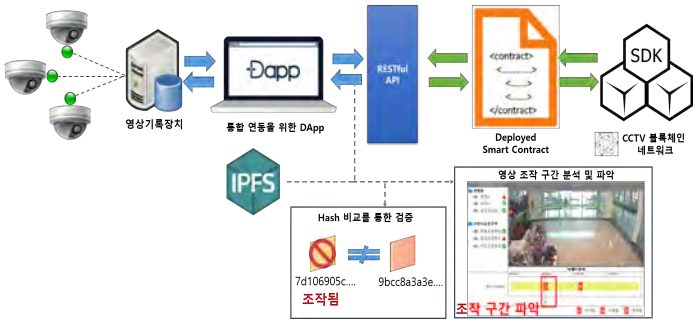


그림 2. 스마트 컨트랙트를 통한 CCTV 영상 조작 설명방식

스마트 컨트랙트를 통해 영상의 조작 여부, 영상의 해쉬 값 저장 및 특정 구간 영상 조회를 가능하게 하기 위해 우리는 스마트 컨트랙트를 다음 표 3.과 같이 3가지의 기능으로 정의한다.

기능	입력값	출력값
영상 저장	영상 해쉬값	
영상 검증	비교 영상 해쉬값	Boolean
특정 구간 영상 조회	시작시간, 종료시간	여러 영상의 Hash값

표 3. CCTV 영상 조작 검증을 위한 스마트 컨트랙트 기능

영상 저장의 경우 저장하려는 영상의 해쉬 값을 입력 값으로 넣어 이를 블록체인에 저장하고 있으며 영상 검증은 비교하려는 영상을 해쉬화 하고 이를 조회하여 블록체인에 있는 경우 True값을 없는 경우 False값을 출력하도록 한다. 마지막으로 특정 구간 영상 조회의 경우 입력 값으로 보고자 하는 영상의 시작과 종료시간을 입력하여 이에 대한 영상의 구간에 해당하는 해쉬 값들을 출력하도록 한다. 만약 영상을 3분 단위로 저장하는 경우 9분짜리 영상을 구간을 요청하게 된다면 3개의 영상의 Hash값을 출력하게 한다.

3. 결론

대용량 영상의 저장과 무결성 그리고 조작 여부 등의 설명을 제공하기 위해 본 연구에서는 블록체인 사설 네트워크를 기반으로 한 CCTV 대용량 영상 무결성 지원 방안을 제안했다. 첫 번째로, Hyperledger Fabric과 IPFS를 결합하여 대용량 CCTV 영상을 저장하기 위한 블록체인 사설 네트워크를 구성하였으며 다음으로 대용량 영상의 트랜잭션 최적화를 위해 영상용 I-Frame 해쉬화 와 기존의 해쉬 알고리즘을 결합하여 이를 Merkle Tree로 트랜잭션화 하였다. 마지막으로 영상을 검증 하고 IPFS의 원본 영상과 비교하여 조작의 설명을 가능하게 하기 위해 스마트 컨트랙트의 기능을 정의하였다.

향후 연구로는 제안된 방안을 기반으로 시 기관 및 사업체와의 협력을 통한 블록체인 네트워크 구축 그리고 대용량의 트랜잭션에 따른 성능과 검증 성능을 평가할 계획이다.

Acknowledgement

“본 연구는 전북테크노파크 우수아이디어제품화연구개발지원사업의 지원으로 수행되었음(과제번호 JBTPSW-2020-A-01)”

4. 참고문헌

[1] M. Kerr and R. van Schyndel, "Adapting Law Enforcement Frameworks to Address the Ethical Problems of CCTV Product Propagation," in *IEEE Security & Privacy*, vol. 12, no. 4, pp. 14-21, July-Aug. 2014, doi: 10.1109/MSP.2014.61.

[2] U. Nedim, “What are the problems with using cctv evidence in court?” Oct. 2019. [Online]. Available: https://nswcourts.com.au/articles/ what-are-the-problems-with-using-cctv-evidence-in-court/

[3] A. Martin, “Advice on using footage as evidence in court,” Oct. 2019. [Online]. Available: https://blinkforhome.co.uk/blogs/news/ advice-on-using-footage-as-evidence-in-court

[4] E. Buchanan, “Relevance and admissibility (vic),” Oct. 2019. [Online]. Available: https://www.gotocourt.com.au/criminal-law/vic/ relevance-admissibility/

[5] Gipp, B., Kosti, J., & Breitinger, C. (2016). Securing Video Integrity Using Decentralized Trusted Timestamping on the Bitcoin Blockchain. *MCIS*.
[6] Liu, M., Shang, J., Liu, P., Shi, Y., & Wang, M. (2018). VideoChain: Trusted Video Surveillance Based on Blockchain for Campus. *ICCCS*.
[7] Kerr, M., Han, F., & Schyndel, R.V. (2018). A Blockchain Implementation for the Cataloguing of CCTV Video Evidence. *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 1-6.
[8] https://docs.ipfs.io/concepts/what-is-ipfs/

스테이트리스 컴퓨팅 환경에서의 서버리스 클러스터의 설계*

정대욱 ○ 배민지 범준석 송기혁 최예람 백종문 지은경
한국과학기술원
{dujung, minjibae, junseok.beom, kihyouk.song, yeram.choi, jbaik}@kaist.ac.kr
ekjee@se.kaist.ac.kr

Design of Serverless Clusters with stateless computing environment

Daeuk Jung○, Minji Bae, Junseok Beom, Kihyouk Song, Yeram Choi,
Jongmoon Baik, Eunkyoung Jee
KAIST
291, Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea

요 약

AWS(Amazon Web Service) Lambda와 같은 서버리스(Serverless) 환경은, 물리적 서버의 운영 관리가 필요 없으며, 클라우드 서비스에서 소스 코드를 실행할 수 있도록 해주지만, 실행 환경에 대한 상태(State)를 유지할 수 없는 특성이 있다. 데이터 분산 처리를 위해서 실행 디바이스들을 소켓으로 연결하고, 클러스터를 구성할 수 있다. 본 연구에서는 이러한 분산 클러스터링을 스테이트리스(Stateless) 컴퓨팅 환경에서 웹 소켓을 이용하여 서로 연결하고 통신할 수 있는 방법에 대해서 제시한다. 이를 통해서 클러스터 노드들의 상태를 실시간으로 모니터링하고, 비동기식 프로토콜을 이용하여 효과적인 분산 작업 처리를 지원하도록 하였다.

1. 서론

AWS(Amazon Web Service) Lambda[1] 와 같은 서버리스(Serverless) 환경은, 물리적 서버의 운영 관리가 필요 없으며, 클라우드 서비스에서 소스 코드를 실행할 수 있도록 해주지만, 실행 환경과 관련된 네트워크 연결 소켓(socket) 정보를 저장할 없는, 즉 상태(State)를 유지할 수 없는 특성이 있다(Stateless). 이는 서버 구성을 단순화하여 확장성을 높여주지만, 클라우드 서비스를 개발하여 이용할 경우 이러한 제약 사항을 고려해 주어야 한다. 클러스터링(Clustering)은 데이터 분산 처리를 목적으로 컴퓨팅 자원을 소켓으로 연결하여 서로 실시간 통신하도록 구성할 수 있다.

본 논문은 서버리스 컴퓨팅에서의 클러스터링을 구성하는 방법에 대해서 제시한다. 2장에서는 쉬운 서버 설치 방법과 노드들의 실시간 모니터링 구성, 그리고 동기화 요청 및 비동기식 다중 분산 처리 방법을 제시하며 그에 따른 성능 결과를 보여준다. 3장에서는 연구 결과의 정리로 마무리한다.

*본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT 연구센터지원사업(IITP-2020-2020-0-01795)과 2019년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. NRF-2019R1I1A1A01062946)

2. 서버리스 클러스터 구현

AWS Lambda[1]는 실행 코드만으로 서버 구성을 쉽게 이용하도록 하고 있다. AWS의 CloudFormation 서비스를 이용하며 클라우드 구성을 프로그램하여 이용할 수 있는데, 구조가 복잡하여 직접 작성하여 이용하기에는 다소 어려움이 있다. 다행히 오픈소스 프로젝트인 Serverless 프레임워크[2]를 이용하면 보다 쉬운 방법으로 클라우드 자원을 설치 구성할 수 있다. 이를 활용하여 아래 그림 1과 같이 단일 명령어 실행으로 5분 이내에 서버 구성이 완료하도록 개발하였다. 모든 관련 코드는 오픈 소스로 개발되어 직접 확인할 수 있도록 하였다 (Serverless Clusters, MIT LICENSE, lemoncloud.io https://github.com/lemoncloud-io/serverless-clusters).

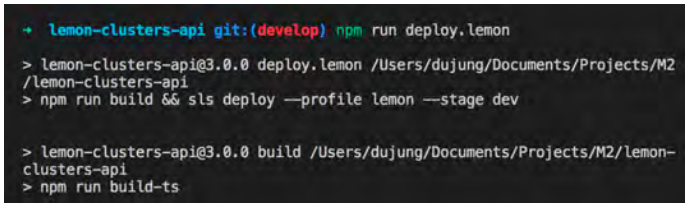


그림 1. 실행 명령으로 서버 구성 자동 완료

2.1. 클러스터 모니터링

클러스터에 연결된 노드들의 CPU나 메모리 등의 상태를 실시간으로 모니터링하고 관리할 필요가 있다. 모니터링을 위한 웹 화면을 만들고, 브라우저에서 웹 소켓으로 서버와 통신하도록 구성하였다. 그리고, 각 연결된 노드들은 매 주기별로 상태 정보를 웹 소켓을 통해 서버로 보내게 된다. 전송된 상태 정보들은 NoSQL데이터베이스에 저장되고, 이후 변경된 데이터를 추출하여 모니터 노드들에 전송 되어진다. 전체적인 구성은 다음의 그림 2와 같이 설계하였다.

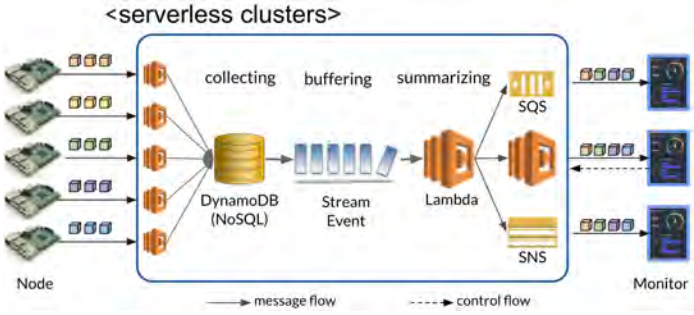


그림 2. 클러스터 모니터링 구조도

각 연결된 노드에서 보내는 상태 정보를 NoSQL 형태의 메인 저장소인 DynamoDB 에 저장되며 업데이트 정보는 DynamoDB의 업데이트 Stream Event를 통해서 수신된다. 이를 서버 실행 Lambda 에서 분석하고 모니터링 노드에 취합된 결과를 실시간 전송하게 된다. 모니터들 또한 동시에 다중으로 연결되며, 모두 동일한 데이터를 수신하게 된다. 노드에서 상태 정보를 송신하여 모니터에서 수신할 때까지의 시간을 측정한 결과 아래 그림 3과 같다. 측정 결과 상태 정보 수신까지의 걸리는 시간은 평균 **745ms** 정도이다.

회수	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	평균
시간(ms)	666	674	834	742	737	712	759	718	791	715	863	694	782	754	778	679	976	563	731	745.7

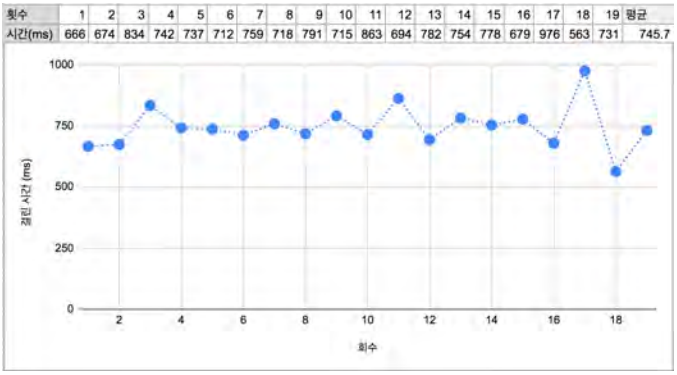


그림 3. 상태 수신까지 걸린 시간(ms)

2.2. 동기화 요청 실행

클러스터에 연결된 노드들의 개별적인 관리와 원격 명령 수행 처리를 지원할 필요가 있다. 이를 위해서, 클러스터에 연결된 노드들에 수행 요청을 클러스터 서비스를 통해서 보내고 그 실행 결과를 수신할 때까지 기다리는 동기화 요청(Synchronous Request)을 지원하도록 하였다. 이를 지원하기 위해서 다음의 그림 4와 같이 서버리스 서버 환경에서 공유 저장소를 이용하여 처리할 수 있도록 구성 개발하였다.

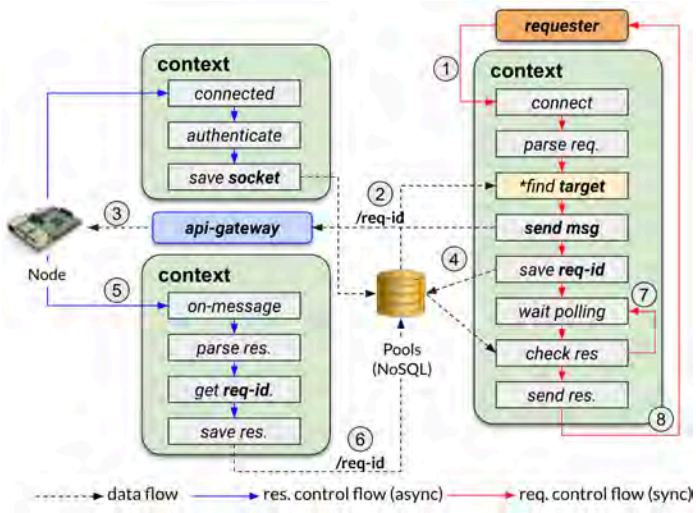


그림 4. 노드 연결과 동기화 요청 구성도

컨텍스트(Context)는 서버리스 환경에서 요청에 의해서 잠시 실행되는 실행 환경을 말한다. 이는 각 요청에 대한 처리를 하고 바로 종료가 되는 특성이 있다. 각 노드가 웹 소켓을 통해서 접속하게 되면, 연결 소켓 정보를 NoSQL형태의 저장소에 저장해 준다. 그리고 이후 이 노드에 명령을 보내기 위한 요청을 요청자(Requester)가 다른 실행 컨텍스트를 통해서 시작하게 된다(①). 간단한 요청 확인후 연결 노드의 접속 정보를 저장소에서 얻는다(②). 이 연결 정보로 소켓 통신의 데이터는 API-Gateway를 통해서 전달하게 되며(③) 관련 요청 정보를 저장소에 저장해 준다(④). 이후 해당 노드에서는 요청을 처리하고 요청ID와 함께 서버로 전달해주면, 이를 받은 서버 컨텍스트는 이를 요청ID에 해당하는 데이터에 업데이트 해주게 된다(⑤,⑥). 요청 서버에서는 시간을 가지고 요청이 완료되었는지 체크하게 되며(⑦), 응답이 완료된 경우 해당 정보를 저장소에서 읽고 이를 최종 응답으로 돌려주어(⑧) 요청에 대한 실행 컨텍스트를 종료하게 된다.

특정 노드에 대한 요청은 http 기반 restful api 방식으로 가능하다. http 요청을 시작으로 응답까지 걸린 총 시간을 autocannon [3] 측정 도구로 측정한 결과는 아래 그림 5와 같다.

```
→ lemon-clusters-api git:(v2) ✗ autocannon -c 8 -a 8 -t 30 -m POST -b '{}' -H '
Content-Type: application/json' localhost:8121/clusters/1000323/execute
worker_threads is not available
Running 8 requests test @ http://localhost:8121/clusters/1000323/execute
8 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	428 ms	440 ms	652 ms	652 ms	513.75 ms	99.84 ms	652 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	8	8	8	8	8	0	8
Bytes/Sec	5.8 kB	5.8 kB	5.8 kB	5.8 kB	5.8 kB	0 B	5.8 kB

```
Req/Bytes counts sampled once per second.
8 requests in 1.02s, 5.8 kB read
```

그림 5. 요청에서 응답까지 걸린 시간

동시 8개의 실행 쓰레드(Thread)로 각 8번의 요청에 대한 평균적인 응답 시간은 약 **513ms** 으로 측정되었다. 특이사항으로는 응답의 표준 편차가 99.84ms으로 측정되는데, 이는 응답 폴링(Polling) 주기를 100ms로 설정한 결과이다.

2.3. 다중 비동기식 요청 처리

요청들에 대한 비동기식 요청(asynchronous request)을 처리하는 방법은 아래 그림 6의 구성과 같다.

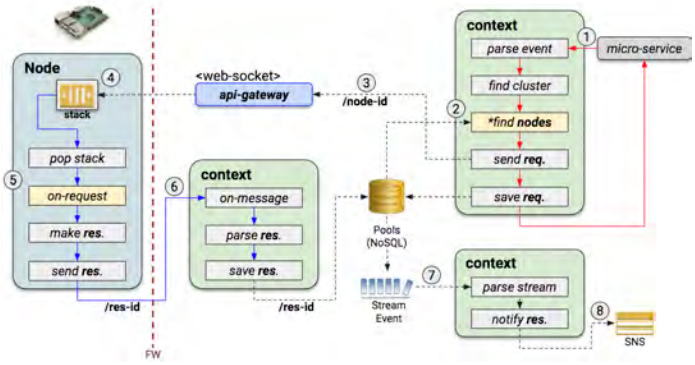


그림 6. 다중 요청 비동기식 요청 처리

비동기식 요청은 처리 응답에 대한 결과를 바로 알 수 없으며, 처리 완료 이벤트 수신으로 알게 된다. 요청자는 클러스터 서비스에 관련 요청 정보를 준비하여 호출하게 된다(①). 이를 처리할 노드 선택은 연결된 노드 들에서 무작위로 선택하게 된다(②). 그리고는 할당된 노들에 관련 요청들을 웹 소켓을 통해서 전달하게 된다(③). 각 노드에는 수신된 요청을 스택(stack) 저장소에 우선 저장되고(④), 이후 타이머를 이용하여 스택에 저장된 요청을 꺼내어 요청을 하나씩 수행하게 된다(⑤). 이후 수행된 결과는 요청ID 정보를 포함하여 서버에 웹 소켓을 통해서 전송되며(⑥), 수신부 서버에서는 해당 요청ID에 결과와 완료를 설정하게 된다. 요청에 대한 응답 변경 이벤트를 스트림을 통해서 추출하게 되며(⑦), 이를 최종적으로 SNS 이벤트를 발생시켜 요청자에게 알려 주도록 하였다(⑧).

성능 검사를 위해서, 다중 요청을 비동기식으로 처리하는 데에 소요되는 전체 시간을 측정하였다. 라즈베리 파이(Raspberry PI 3) 기기에 NodeJS기반 웹 소켓 클라이언트 5개를 준비하여 클러스터를 구성하였다. 단위 요청은 소수(Prime Number) 여부를 계산하도록 하고, 기기마다 일정한 처리 시간이 걸리도록 구성하였다 (2.76 sec/request). 총 60개의 다중 요청을 여러 노드들에 분산 처리 요청할 경우, 총 걸린 시간을 측정한 결과는 아래 그림 7과 같다. 그 결과 1개, 2개, 5개 노드들을 이용하여 분산 처리한 결과는 각각 165초, 84초(50% 감소), 33초(80%감소)로 나왔다. 분산 처리에 참여한 노드들의 개수에 따라 전체 시간이 줄어듦을 알 수 있다. 이후 6개부터는 실제 연결된 노드들이 5개 이므로, 총 걸린 시간에는 영향이 없음을 알 수 있다.

그림 7. 다중 요청 분산 처리에 걸린 총 시간

3. 결론

본 연구는 서버리스 클라우드 실행 환경에서 웹 소켓을 이용하여 클러스터를 구성하여 이용하는 방법을 제시하였다. 이를 위해서 여러 오픈 소소들을 이용하고, 간단한 명령어 실행으로 바로 서비스 구성 가능하도록 오픈 소스로 개발 제시하였다. 구성된 클러스터는 노드들의 상태를 실시간으로 모니터링 할 수 있도록 하면서도, 클러스터 연결 노드들에 동기화 명령 전달이 가능토록 설계 및 개발하였으며, 그리고 다중 요청들에 대한 비동기식 처리가 가능하도록 각 노드에 스택 메모리를 이용하여 수행하도록 개발하였다. 그 결과 서버리스 컴퓨팅의 확장성과 안정성을 최대한 이용하면서, 연결된 클러스터 노드들의 관리를 용이하게 처리할 수 있게 되었다. 이는 분산 데이터 처리를 위한 클러스터 구성을 클라우드 환경에서 쉽게 구성하여 이용할 수 있도록 하며, 다중 노드들의 효과적 연결 관리를 통해 그 활용도가 매우 클 것으로 기대된다. 향후 개별 노드의 특성을 고려하여 대규모 분산 처리 클러스터를 효율적으로 운영하는 방안에 대한 추가 연구가 필요할 것으로 생각한다.

4. 참고문헌

[1] AWS Lambda Help
https://docs.aws.amazon.com/ko_kr/lambda/
[2] Serverless Open Source Framework. MIT License.
https://www.serverless.com/open-source/
[3] fast HTTP/1.1 benchmarking tool written in Node.js. MIT License.
https://github.com/mcollina/autocannon

eBPF 기반 System Monitoring Application 조사 연구

송소민 ○
경북대학교
Sominsong97@knu.ac.kr

A Study on the eBPF-based system Monitoring Applications

Somin Song ○
Kyungpook National University

요 약

본 연구에서는 컨테이너의 실시간 모니터링 기법들에 대한 전반적인 이해를 위해 알려진 기법들의 기능과 제약사항을 상세하게 분석한다. 분석 결과의 검증을 위하여 성능을 비교하고 장단점들을 파악한다. 본 연구의 결과는 현재 존재하는 컨테이너 동작 모니터링 기법의 장단점에 대한 분석을 통해 주어진 환경에 맞는 보안 기법을 선택하여 사용하는 결정에 활용할 수 있다. 본 연구를 통해 조사된 eBPF 기반의 컨테이너 모니터링 기법들의 장단점과 성능 분석을 통해 현재 기법들의 이해도를 높이고 컨테이너 보안의 보안성을 높이는데 기여할 것이다.

1. 서 론

컨테이너 기술이 IT업계의 전반으로 확산됨에 따라 기업들의 경쟁력 확보를 위해 컨테이너 기술의 도입은 필수로 자리 잡았다. 하지만 컨테이너의 도입이 확산될수록 컨테이너 보안 문제의 중요성이 인식되어 다양한 연구가 진행되고 있다 [1,2,3,4]. 주요 컨테이너 보안 관련 연구는 시스템 콜 개수를 차단하여 공격 표면을 감소하는 것을 목표로 한다 [5,6]. 이를 적용하여 보안성이 강화된 gVisor, Kata Container, Lupine, Nabla Linux, 다수의 Unikernel 계열들도 시스템 콜 개수의 차단에 주력하고 있다.

하지만 이러한 전략만으로 컨테이너 보안 문제를 모두 해결하기에는 충분하지 않으며 컨테이너의 동작을 실시간으로 감시하는 방법이 병행되어야 한다.

본 연구에서는 효과적인 실시간 컨테이너 모니터링 기법을 조사하는 것을 목표로 한다. 특히, 현재의 여러 실시간 컨테이너 기법들과 최신 커널 모니터링 기법인 eBPF [7]를 사용하여 현재의 실시간 컨테이너 기법이 가지는 문제점들을 분석한다. 이러한 분석 결과와 성능 비교 실험을 통해 본 논문에서는 eBPF를 사용하는 각 모니터링 어플리케이션에 대한 체계적인 성능 분석을 통해 어떠한 부분이 개선의 여지가 있는지를 분석하여 제시한다.

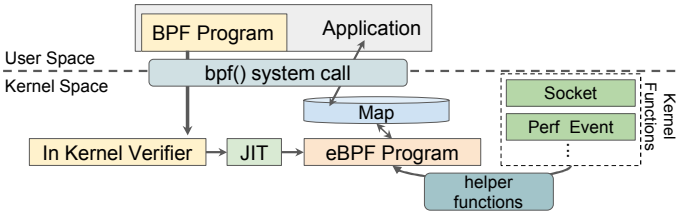
2. 배경 지식

2.1 eBPF 개요

Extended BPF (Berkeley Packet Filter)의 약자이다. 일련의 코드를 커널 내부에서 안전하게 실행시키는 도구이며, 커널 모듈이 아닌 코드를 실행하는 커널 내부의 가상 머신이다. BPF는 Packet Filter라는 이름처럼 처음에는 패킷 필터링을 수행하는 커널 내부 코드 실행을 위해 제작되었지만, 현재 security, auditing, tracing, monitoring 등 다양하게 활용되고 있다 [8,9].

2.2 eBPF 구조

eBPF 프로그램은 유저 어플리케이션에서 bpf() 시스템 콜을 통해 커널로 전달된다. 그 후, 전달된 프로그램이 안전한 프로그램인지 확인하는 In-Kernel Verifier가 프로그램을 체크하고, JIT(Just-in-time) 변환을 통해 해당 프로그램 중 일부 코드를 kernel native code로 컴파일하여 최적화한다. 최적화된 eBPF 프로그램은 실행되면서 Map을 통해 유저 애플리케이션과 통신하고, helper function을 통해 일부 커널 함수들을 사용한다. 이후 해당 정보에 대하여 Map을 통해 유저 애플리케이션과 통신하고, helper function을 통해 일부 커널 함수를 사용한다. 이러한 eBPF 프로그램 구조는 (그림 1)에 도식화되어 있다.



(그림 1) eBPF 구조

3. Monitoring Tool 분석

3.1 BPFtrace 기술 상세 분석

BPFTrace [10]는 eBPF를 사용한 시스템 콜, 커널 function 호출을 tracing 할 수 있는 도구이며, 제한적이지만 출력문을 조정할 수 있다. eBPF 프로그램 컴파일 없이 명령어만을 통해 원하는 기능을 수행하도록 제작이 가능하다.

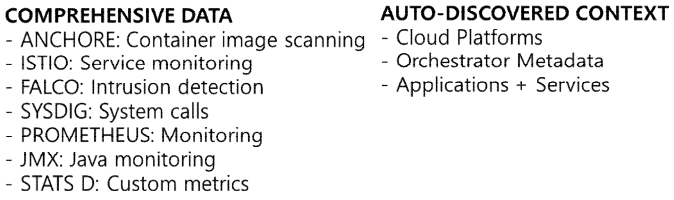
직접 스크립트를 작성하여, 모니터링 및 커널 동작에 따른 이벤트를 발생시킬 수 있다는 특성으로 인해, 비교적 자유로운 활용이 가능하나, 사용자가 직접적으로 스크립트를 제작하여야 하기 때문에, 일정 수준 이상의 이해도가 필요하다는 단점이 존재한다.

3.2 Sysdig 기술 상세 분석

Sysdig [11,12]는 라우터에서 패킷을 수집하는 것이 아닌 End-point에 더욱 가까운 위치에서 데이터를 모아 관찰하는 모니터링 툴로써 제작되었다. 그리고 현재는 모니터링 툴의 기능을 넘어서 (그림 2)의 항목과 같이 여러 종류의 추적 어플리케이션에서 모은 데이터를 통합하는 기능을 포함하는 하나의 플랫폼이 되는 것을 목표로 개발이 되고 있다.

Sysdig는 eBPF를 사용하여 커널에 probe를 삽입하고 시스템 콜을 수집하여 수집된 시스템 콜을 활용하여 여러 목적으로 분석할 수 있게 한다. 서비스를 컨테이너에서 수행하는 경우 Monitoring, Trouble-shooting, Security 관리를 해결할 방법을 찾는 목적으로 사용할 수 있으며, Sysdig는 쿠버네티스 혹은 클라우드 기반 서비스를 사용하는 큰 기업들에 서비스를 제공하는 기반 기술이 되는 것을 목표로 한다. 일반적인 eBPF 모니터링 툴은 런타임에 발생하는 이벤트들을 분석하는 것이 주요 목표이지만, Sysdig는 컨테이너 빌드 시에 확인해야 하는 보안 위배 요소들을 검사하는 기능도 포함되어 있다.

Sysdig는 모니터링 대상에 따라 자동으로 알맞은 metric을 선택하여 출력하므로 특정 어플리케이션을 위한 plug-in 설치와 같은 추가 작업이 필요하지 않으며, 데이터 수집이 하나의 프로세스에서 이루어지므로 여러 종류의 다양한 사용하는 것보다 프로세스를 오버헤드가 상대적으로 적다는 장점이 있다. 그러나, 설치 시 Kernel Header의 설치가 필요하며 Opensource 이외에 Product가 따로 개발되어 모든



(그림 2) Sysdig Data Integration

기능을 사용하며 테스트할 수 없다는 문제점이 존재한다.

3.3 Tracee 기술 상세 분석

Tracee [13]는 eBPF를 이용하여 컨테이너 및 시스템을 tracing할 수 있는 tool이다. Tracee의 tracing 과정은 시스템 콜과 실시간 시스템 이벤트에 대해 이루어진다. 이 때, 네임스페이스 정보도 출력하여, 같은 파드의 다른 컨테이너를 구분하여 관찰이 가능하다. 또한, Tracee가 생성된 후, 새로 생성되는 프로세스나 컨테이너를 tracing할 수 있다.

Tracee는 기존 eBPF trace tool과 달리, 출력된 이벤트 기록에서 각 컨테이너에서 발생한 이벤트를 구별 가능하다. Kernel 영역에서 container id를 직접적으로 얻을 방법은 없지만, Docker와 Kubernetes의 경우 대개 container id를 UTS 네임스페이스로 지정하기 때문에, 사용자는 이를 통해 컨테이너를 구별할 수 있다. 또한 옵션을 통해 Tracee의 추적 시작 이후 생성된 컨테이너의 이벤트만 필터링할 수 있다. Container 고유의 PID 네임스페이스 안에서 생성된 첫 번째 프로세스는 ID를 1로 갖는 점을 활용하여 PID 네임스페이스가 다르고, 이벤트가 발생한 프로세스의 ID가 1인 경우 새로 생성된 container로 판단하고 기록한다.

개발 현황으로는 기존에 python을 통하여 제작된 frontend가 Go를 통하여 재작성되었다. Go 언어는 Cloud 환경에서 활발하게 사용되어, Tracee를 각 시스템에 integration 하기가 용이하다는 장점을 가지고 있다. Go 언어로 제작하여 build된 Tracee는 python로 개발된 기존 Tracee보다 실행속도가 더 빠르다. 그러나 eBPF를 통한 event 기록 시 ring buffer를 사용하여 그 정보를 저장하고, front-end에서 가져와 처리할 때, 그 과정에서 event 정보가 기록되는 속도가 읽는 속도보다 빠르면 event lost가 발생하는 현상이 나타난다는 문제점이 있다.

4. 모니터링 툴 간 성능 비교

4.1 실험 개요

각 시스템 콜 tracing tool들은 호스트에서 시스템 콜이 발생 시, 실행 흐름을 어플리케이션으로부터

<표 1> Tracing Tool 별 차이점 비교

	BPFTrace	Sysdig	Tracee
Tracing 범위	Events, Kernel function, System call	Events, Kernel function, System call	Events, System call
별도의 eBPF 프로그램	필요 없음	필요 없음	필요 없음
출력 양식	제한적으로 제작 가능	Table, Graphic	Table, JSON
Tracing 단위	Process	Process	Container, Process
특정 이벤트 지정 지원	O	O	O
특정 프로세스 지정 지원	O	O	O

전달받고(Hook) 각기 다른 이벤트 핸들러를 통해 기록 작업을 수행하고 다시 어플리케이션에게 실행 흐름을 반환한다. 이때 시스템 콜 tracing tool 간의 각기 다른 이벤트 핸들러 구현으로 인해 각기 다른 hooking overhead를 가지게 된다. 따라서 본 절에서는 각 tracing tool들의 hooking overhead를 비교하기 위하여 특정 어플리케이션을 각 tracing tool들과 함께 실행하였을 경우와 수행 시간을 비교 분석한다.

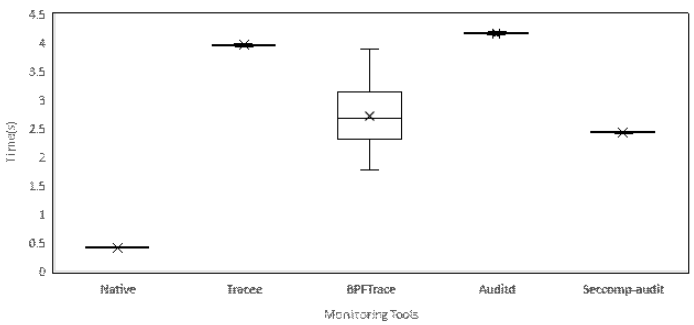
4.2 실험 구성

본 연구에서 구성한 실험 조건은 아래와 같다.

- H/W 사양
  - CPU: intel i7-4770 3.50GHz
  - Memory: 8GB
  - Network: 1000Mbps
- S/W 버전
  - OS: Ubuntu 20.04
  - Kernel: 5.4.0-52-generic
  - Tracee: v0.1.0, BPFTrace: v0.11.0, Auditd: v2.8.5
  - Docker: v19.03.12

Tracee와 BPFTrace는 공통적으로 Perf ring buffer를 사용하여 커널 스페이스의 데이터를 유저 스페이스로 전달하는데, 이때 충분하지 않은 ring buffer 사이즈를 지정하면 이벤트 손실이 발생하여, 해당 Tracing tool이 해당 이벤트에 대한 처리를 못할 수도 있다. 따라서 이를 방지하고자 65,536 pages로 ring buffer 크기를 충분히 할당하였다.

위의 실험 환경에서 getpid() 시스템 콜을 100만 회 호출하는 프로그램을 실행시켜 모니터링 툴과 함께 수행 시 측정되는 수행 시간을 비교하여 나타낸다.



(그림 3) getpid() 시스템 콜 100만 회 호출 실험결과

4.3 실험 결과

Getpid 시스템 콜을 반복 실행하는 실험을 수행한 결과는 (그림 3)와 같이 나타났다. Tracee는 Native 대비 956.6% 많은 소요시간을 보이는 것을 확인할 수 있고, Seccomp-audit, BPFTrace, Tracee, Auditd 순으로 적은 소요시간을 보였다. 또한 BPFTrace의 실험결과가 높은 분산을 보이는 것을 확인할 수 있는데 이는 BPFTrace가 단일 스레드로 동작하는 프로그램이기 때문에 해당 프로세스의 CPU 점유율이 쉽게 100%를 차지하므로 실험결과가 나타난 것으로 판단된다.

<표 1>은 각 모니터링 어플리케이션 사이의 차이점을 정리한 표이다. 이를 통해 모두 eBPF를 활용한 Tool임에도 불구하고, 각기 다른 특성을 가지는 것을 확인할 수 있다. BPFTrace는 이벤트와 kernel function, 그리고 시스템 콜에 대한 tracing을 수행할 수 있지만, Tracee의 경우 이벤트와 시스템 콜에 대한 tracing만을 지원한다. 세 가지의 tracing tool 모두 특정 이벤트만을 지정하여 tracing이 가능하지만, 특정 프로세스를 지정하여 tracing하는 기능은 BPFTrace만 지원한다

5. 결론

컨테이너의 보안성을 고수준으로 높이기 위해서는 컨테이너 보안 런타임을 사용함과 동시에 실시간 모니터링 및 추가적인 보안 정책에 의한 접근 제어 및 시스템 콜 호출 제어를 병행하여야 런타임의 방어 범위를 벗어나는 공격에 대해서도 인지 및 대처가 가능하다.

현재 최신의 실시간 모니터링 및 추가적인 보안 정책을 적용할 수 있는 모듈의 경우, 최신 Linux 커널에 추가된 기술인 eBPF 기법이 주로 사용되고 있다. 본 과제에서는 eBPF 기법을 이용하는 여러 모니터링 및 보안 기법들에 대하여 기술적 분석을 진행하였다.

5. 참고 문헌

[1] Neves, Francisco, Ricardo Vilaça, and José Pereira. "Black-box inter-application traffic monitoring for adaptive container placement." Proceedings of the 35th Annual ACM Symposium on Applied Computing. 2020.
[2] Bélair, Maxime, Sylvie Laniepce, and Jean-Marc Menaud. "Leveraging Kernel Security Mechanisms to Improve Container Security: a Survey." Proceedings of the 14th International Conference on Availability, Reliability and Security. 2019.
[3] Jian, Zhiqiang, and Long Chen. "A defense method against docker escape attack." Proceedings of the 2017 International Conference on Cryptography, Security and Privacy. 2017.
[4] Lin, Xin, et al. "A measurement study on linux container security: Attacks and countermeasures." Proceedings of the 34th Annual Computer Security Applications Conference. 2018.
[5] Ghavamnia, Seyedhamed, et al. "Temporal system call specialization for attack surface reduction." 29th {USENIX} Security Symposium ({USENIX} Security 20). 2020.
[6] Ghavamnia, Seyedhamed, et al. "Confine: Automated system call policy generation for container attack surface reduction." 23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020). 2020.
[7] https://kubernetes.io/blog/2017/12/using-ebpf-in-kubernetes/
[8] Cassagnes, Cyril, et al. "The rise of eBPF for non-intrusive performance monitoring." NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2020.
[9] Levin, Joshua. "ViperProbe: Using eBPF Metrics to Improve Microservice Observability." (2020): 13.
[10] https://github.com/iovisor/bpftrace
[11] https://sysdig.com/
[12] https://www.brighttalk.com/webcast/16287/351084
[13] https://github.com/aquasecurity/tracee

Variational AutoEncoder를 활용한 여행자 후기 기반 관광지 추천 알고리즘

조원희[1 O], 김창영[2], 김수은[3], 김윤민[4], 양형정[1 *]
전남대학교 인공지능융합학과[1], 전남대학교 IoT 인공지능융합전공[2],
전남대학교 소프트웨어공학과[3], 전남대학교 빅데이터금융공학융합전공[4]
{206465, 170544, 175767, 175973, hjyang}@jnu.ac.kr

Tourist Recommendation Algorithm based on Traveler Reviews using Variational AutoEncoder

Won-Hee Jo[1 O], Chang-Yeong Kim[2], Su-Eun Kim[3], Yoon-Min Kim[4], Hyung-Jeong Yang[1 *]
Department of Artificial Intelligence Convergence, Chonnam National University[1]
IoT Artificial Intelligence Convergence Major, Chonnam National University[2]
Department of Software Engineering, Chonnam National University[3]
Big Data Financial Engineering Convergence Major, Chonnam National University[4]

요 약

본 연구에서는 기존의 추천 시스템 구현 방법들이 보이는 많은 양의 메모리 요구, 낮은 정확도, 정보 손실 등의 한계를 해결하기 위해 관광지 추천 모델에 VAE 기반의 모델을 도입하였다. 추천 시스템의 정확도를 높이기 위해 여행자 후기 데이터의 감성 정보를 관광지 방문 데이터에 병합하여 관광지 방문 데이터 만으로는 알기 어려운 여행자의 선호 정보를 반영한 관광지 추천 알고리즘을 개발하였다. VAE 기반의 모델을 사용하여 기존 추천 방식들의 한계를 해결하고 여행자 후기 데이터의 감성 정보를 적용함으로써 추가적인 성능 향상을 이루었다.

1. 서 론

최근 인터넷 산업이 발전함에 따라 추천 시스템에 대한 산업계의 수요가 증가하고 있다. 추천 시스템의 구현 방법 중 하나인 협업 필터링은 사용자와 항목 간의 상호작용을 기반으로 가장 관련도가 높은 항목을 추천해주는 기법이다. 전통적인 협업 필터링 기법 중 하나인 메모리 기반 협업 필터링의 경우 구현이 단순하지만 많은 양의 메모리를 요구하며 단순히 사용자의 선호 정보만을 고려한다는 단점이 있다. 이에 반해 모델 기반 협업 필터링의 경우 잠재 변수를 통해 사용자의 특성을 압축하기 때문에 상대적으로 적은 메모리를 요구하며 높은 정확도를 보이지만 잠재 요인을 추출하는 과정에서 정보가 손실된다는 단점이 있다.

[1, 2, 3]는 추천 모델에 딥러닝과 통계적 기법을 결합한 VAE(Variational AutoEncoder)를 적용하여 정보의 손실을 최소화하면서 구조적인 특성을 추출하기 위한 방법을 제안하였다. [4, 5]는 추천 시스템의 성능을 개선하기 위해 SNS(Social Network Service)에서 텍스트 형태의 후기 데이터를 수집하여 감성분석을 통해 긍정과 부정으로 분류한 뒤 좋아요 수나 팔로워 수 등에 비례하는 영향력 파라미터를 측정하여 선호도에 사용자의 영향력을 반영하는 방법을 제안하였다.

본 논문에서는 VAE 모델 기반 추천 시스템[1, 2, 3]의 성능을 관광지 방문 데이터를 활용하여 향상시키는 방법을 제안한다. 여행자 후기 데이터에서 텍스트를 추출하여 감성분석을 수행한 후 감성 정보를 관광지 방문 데이터에 병합하여 각 개인의 기호를 반영한 맞춤형 관광지 추천 알고리즘을 개발하고자 한다. 실험 결과 VAE 모델이 기존 협업필터링 추천방법보다 재현률(Recall)과 NDCG(Normalized Discounted Cumulative Gain)의 성능이 대폭 개선된 것을 보이며 여행자 후기 데이터의 감성 정보를 가중치로 적용하였을 때 추가적인 성능향상을 보인다.

2. 제안 방법

관광지 추천 시스템에서 여행자 후기 데이터는 규모가 작으며 정보량이 희소하기 때문에 관광지 방문 데이터를 모델의 입력 데이터로 사용하고 여행자 후기 데이터는 감성 정보를 분석한 후 관광지 방문 데이터에 병합하여 보조적으로 사용한다. 추천 모델의 출력은 각 사용자의 각 관광지에 대한 선호도로 볼 수 있으며 최종 추천 목록은 최종 선호도를 크기 순으로 정렬하여 사용자에게 관광지를 추천해주게 된다. 그림 1은 전체적인 시스템 구성도를 보이고 있다.

그림 1 여행자 후기 기반 VAE 추천 시스템 구성도

VAE 모델은 기존의 AutoEncoder 모델에서 입력 데이터가 통계적 과정을 통해 생성되었다고 가정하여 구조적인 잠재요인을 추출하는 방식으로 그림 2는 VAE 모델의 아키텍처를 나타낸다. VAE 모델의 인코더는 입력 데이터를 잠재변수 z의 평균과 분산 파라미터로 인코딩하며 두 파라미터를 모수로 하는 정규분포에서 무작위로 하나의 샘플을 추출하여 디코더는 이를 입력 데이터로 복원한다. 이 과정에서 VAE 모델은 입력 데이터에 대한 구조적이고 연속적인 잠재공간을 학습한다.

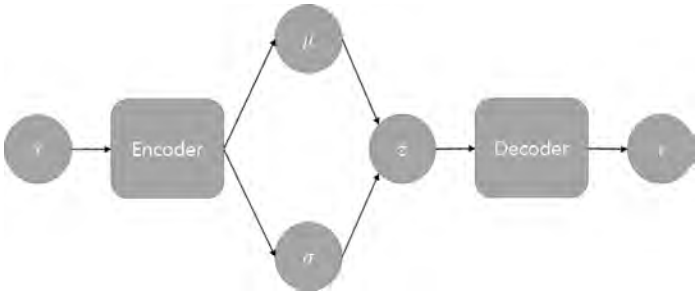


그림 2 VAE 모델 아키텍처

[1]은 VAE 모델을 추천 시스템에 적용하기 위해 기존의 정규분포로 가정하던 사전분포를 다항분포로 변경하고 기존의 VAE 모델의 손실함수 (1)에서 규제 손실 부분에 정규화의 강도를 제어하기 위한 β 파라미터를 추가하여 (2)와 같이 변형하였다.

$$\mathcal{L}_{VAE} = \mathbb{E}_{q_\phi}(z|x)[\log p_\theta(x|z) - KL(q_\phi(z|x) \parallel p(z))] \quad (1)$$

$$\mathcal{L}_{\beta-VAE} = \mathbb{E}_{q_\phi}(z|x)[\log p_\theta(x|z) - \beta \cdot KL(q_\phi(z|x) \parallel p(z))] \quad (2)$$

Multi-VAE[1]를 기반으로 한 RecVAE[2]는 Encoder 아키텍처에 정규화와 잔차 연결을 추가하고 사전분포와 베타 파라미터를 변형함으로써 성능을 향상시킨 모델이다. H+Vamp Gated[3]는 [1]에 보다 유연한 사전분포를 사용하고 게이팅 메커니즘을 도입하여 성능을 향상시킨 모델이다. 본 연구는 VAE모델 중에 H+Vamp Gated를 사용하고 결과를 RecVAE 및 기존 협업필터링 모델과 비교하였다.

여행자 후기 데이터의 감성 정보는 긍정, 중립, 부정 3개의 범주로 나누어지며 분류된 감성 정보를 모델의 입력에 적용하기 위해 긍정은 1의 값을 부여하고 중립과 부정은 0의 값을 부여하였다. 여행자 후기 데이터의 감성 정보를 모델의 입력에 병합하여 사용함으로써 관광지 방문 데이터 만으로는 알 수 없는 선호 정보가 모델의 학습에 반영되어 상위 추천 목록에 사용자가 보다 선호할 수 있는 항목이 추천된다.

3. 실험

3.1 데이터 셋

실험에 사용한 데이터는 스탬프투어 모바일 앱에서 수집한 데이터로 여행자가 남긴 후기 데이터와 관광지 방문 데이터로 이루어져 있다[6]. 표 1에 수집한 데이터의 통계를 보이고 있으며 희소성이 높은 것을 알 수 있다. 여행자 후기의 감성 정보는 4명의 연구자가 독자적으로 분류한 뒤 과반수로 선택된 범주로 정하였다. 그림 3에서 보이는 것과 같이 긍정이 약 76%이고 부정과 중립은 약 24%로 보이고 있다.

표 1 스탬프투어 데이터 통계

스탬프투어 데이터 통계			
전체 사용자		28335	
전체 관광지		1543	
여행 후기 데이터		관광지 방문 데이터	
전체 리뷰 수	3979	전체 방문 수	140463
리뷰를 남긴 사용자	1053	관광지를 방문한 사용자	15695
리뷰가 있는 관광지	758	사용자가 방문한 관광지	1325
희소성	0.499%	희소성	0.675%

그림 3 여행 후기 감성분석

추천 시스템을 개발 및 평가하기 위해 관광지 방문 데이터를 이진 데이터 형태의 사용자와 관광지 간의 희소 행렬로 만든다. 사용자 수 기준으로 학습 데이터와 평가 데이터로 샘플링하며 평가 데이터는 각 사용자별로 값이 1인 항목 중 20%를 무작위로 선택하여 입력 데이터와 타겟 데이터로 나누어 사용한다.

3.2 평가지표

평가지표로는 검색 시스템 등에서 랭킹 기반 지표로 널리 사용되는 Recall과 NDCG를 사용하였다. Recall은 사용자가 실제 긍정적으로 평가한 항목이 상위 K개의 추천된 목록에 얼마나 포함되어 있는지에 대한 비율로

식은 다음과 같다. rel은 실제 긍정적으로 평가한 항목의 집합이며 ret_K은 상위 K개의 추천된 항목의 집합이다.

$$Recall@K = \frac{rel \cap ret_K}{min(K, rel)}$$

NDCG는 log 정규화를 통해 랭킹 순서에 가중치를 주어 상위 랭킹을 얼마나 잘 예측하는지를 평가하기 위한 지표로 식은 다음과 같다. rel_i는 i번째 순위로 추천된 항목의 실제 긍부정 값으로 분모에 로그를 적용하여 상위 랭킹에 보다 비중을 주게 된다. IDCG(Ideal Discounted Cumulative Gain)는 가능한 최상의 DCG(Discounted Cumulative Gain) 값으로 이를 DCG 값에 나눠주어 [0,1] 범위로 정규화 해준다. Recall과 NDCG를 사용자별로 계산한 평균 값을 최종 지표로 사용한다.

$$NDCG@K = \frac{DCG@K}{IDCG@K}$$

$$DCG@K = \sum_{i=1}^{K} \frac{rel_i}{log_2(i+1)} \qquad IDCG@K = \sum_{i=1}^{K} \frac{rel_i^{opt}}{log_2(i+1)}$$

3.3 실험결과

본 실험에서는 전통적인 메모리 기반 협업 필터링 기법 중 하나인 IBCF(Item Based Collaborative Filtering)의 3가지 메소드[7]와 VAE 모델 기반의 RecVAE와 H+Vamp Gated을 비교하며 여기에 본 연구에서 제안한 방법을 적용하였을 때의 성능을 비교한다.

표 2는 각 평가지표 별 가장 높은 성능을 보이는 모델(볼드체)과 두 번째로 높은 성능을 보이는 모델(밑줄)을 나타낸다. 본 연구에서 제안한 방법으로 여행자 후기 데이터의 감성 정보를 관광지 방문 데이터에 병합하여 학습시켰을 때 관광지 방문 데이터만 사용하였을 때 보다 전체적으로 0.01~0.02 정도의 성능이 향상되었으며 K값이 10일 때를 제외하면 H+Vamp Gated가 RecVAE 보다 Recall과 NDCG의 성능이 높은 것을 알 수 있다. RecVAE의 경우 감성 정보를 병합하였을 경우 K값이 10일 때와 20일 때 Recall이 떨어지고 NDCG가 상승하였는데 이는 상위 항목을 보다 잘 추천하는 것으로 해석할 수 있다.

표 2 실험결과

	NDCG@5	NDCG@10	NDCG@20	Recall@5	Recall@10	Recall@20
H+Vamp Gated + apply review	**0.75540**	0.78316	**0.80045**	**0.82361**	0.89555	0.94897
RecVAE + apply review	0.75030	**0.78331**	0.79803	0.81703	0.90119	0.94831
H+Vamp Gated	0.74728	0.78117	0.79696	0.81217	0.89952	0.94656
RecVAE	0.73360	0.77339	0.79109	0.79919	**0.90156**	**0.95691**
IBCF (Cosine)	0.56014	0.61159	0.64346	0.68139	0.80885	0.90721
IBCF (AdjCosine)	0.57586	0.62941	0.66072	0.68417	0.81677	0.90841
IBCF (Pearson)	0.62970	0.67191	0.69725	0.72517	0.83375	0.91285

4. 결 론

본 연구에서는 VAE 모델 기반 협업필터링에 관광지 방문 데이터와 여행자 후기 데이터의 감성 정보를 병합하여 모델의 입력 데이터로 사용하는 방법을 제안함으로써 이를 적용하였을 때 성능의 개선을 보였다. 협업 필터링은 기본적으로 신규 사용자나 신규 아이템 등 상호작용 정보가 없는 경우 추천이 불가한 콜드 스타트 문제와 사용자의 관심이 저조하거나 상호작용 정보가 부족한 아이템은 추천 목록에 노출되지 않고 소수의 인기 있는 아이템만 추천 목록에 주로 노출되는 롱테일 문제 등이 있다.

향후 연구방향으로 협업 필터링에 내용 기반 필터링을 결합한 하이브리드 추천 시스템을 적용하는 연구와 상호작용 정보가 부족하더라도 사용자가 만족할만한 관광지가 추천 목록에 노출될 수 있도록 하는 보다 유연한 감성 정보 적용 방법 개발 등이 있다. 또한 본 연구에서는 후기 데이터의 감성 정보를 수작업으로 분류하였으나 향후 유사한 다른 데이터에서 학습된 감성분석 모델을 적용하여 보다 실용적인 시스템을 구성할 수 있을 것이다.

감사의 글

본 과제(결과물)는 교육부와 한국연구재단의 재원으로 지원을 받아 수행된 사회맞춤형 산학협력 선도대학(LINC+) 육성사업의 연구결과입니다.

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음 (IITP-2020-2016-0-00314)

참고 문헌

[1] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, Tony Jebara, “Variational Autoencoders for Collaborative Filtering“, ACM WWW ’18, 2018

[2] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, Sergey I. Nikolenko, “RecVAE: a New Variational Autoencoder for Top-N Recommendations with Implicit Feedback“, ACM WSDM ’20, 2020

[3] Daeryong Kim, Bongwon Suh, “Enhancing VAEs for Collaborative Filtering: Flexible Priors & Gating Mechanisms“, ACM RecSys ’19, 2019

[4] 정회윤, 양형정, “트위터 내의 감성 분석을 이용한 여행지 추천 시스템“, 한국정보통신학회논문지, 2014

[5] 박기성, “소셜 네트워크 데이터 분석과 협업 필터링 알고리즘을 통한 영화 추천 시스템“ 석사학위논문, 전남대학교 대학원, 2017

[6] 댓츠잇 주식회사, “스탬프투어 모바일 앱“, https://stamptour.modoo.at/

[7] Michael Leben, “Applying Item-based and User-based Collaborative Filtering on the Netflix Data“, HPI Uni Potsdam, 2008

국지적 경로 공간 탐색 문제를 완화하기 위한 다방향 Concolic 탐색 전략

최한솔, 홍신

한동대학교 전산전자공학부
{21831008, hongshin}@handong.edu

Multi-directional Concolic Testing Search Strategies for Mitigating Path-space Local Search Problem

요 약

본 논문은 다양한 경로 공간으로 탐색 가능성이 있을 것으로 예상되는 복수의 경로를 번갈아 가며 탐색함으로써 동시에 여러 방향으로의 탐색이 진행되게 하는 다방향 Concolic 탐색 전략을 제안하고, 이를 실제 테스트 생성에 적용하여 분기 커버리지 달성 정도를 평가한 결과를 소개한다. 다방향 Concolic 탐색 전략은 기존에 개발된 Concolic 탐색 전략을 사용하여 새로운 경로를 탐색하는 과정에서 발견된 경로를 평가하여 새로운 경로 공간으로 탐색할 가능성이 있을 것으로 보이는 경로를 저장하고, 향후 저장했던 경로부터 탐색을 다시 시작하는 방법으로 다양한 공간이 탐색되도록 유도한다. 본 논문에서는 C 프로그램을 위한 Concolic 테스팅 툴인 CREST에 다방향 Concolic 탐색 전략을 적용하여 4가지 프로그램(Vim, Grep, Busybox Awk, Busybox Sed)에 실험하였으며, 제안한 탐색 기법이 기존의 탐색전략보다 더 높은 분기 커버리지(평균 4.1% 향상)를 더 빠르게 달성하는 것을 확인할 수 있었다.

1. 서론

Concolic 테스팅[1,2]은 프로그램 코드를 분석해서 얻은 정적 실행 경로 정보(symbolic execution)와 실제 프로그램 실행 런타임에서 관찰한 동적 실행 경로 정보(concrete execution)를 결합하여 검증 대상 프로그램의 다양한 경로를 탐색하는 테스트 입력을 생성하는 자동 테스팅 기법이다. Concolic 테스팅은 기호실행 기능을 구현한 탐침을 테스트 대상 프로그램에 삽입한 후, 이를 실제 테스트 입력으로 실행함으로써 실행된 경로를 심볼릭 입력에 대한 조건식으로 표현하고, 다음 실행을 위해 조건식을 이전 실행과 다른 경로를 실행하는 조건식으로 변형한 뒤, SMT 해법기를 통해 변형된 조건식을 해결함으로써 새로운 실행 경로에 도달하는 테스트 입력을 지속적으로 생성하는 방식으로 동작한다.

Concolic 테스팅에서 탐색 전략(Search Strategy)[3]이란 이전 입력에 해당하는 경로에서 추출된 조건식을 변형하여 어떤 경로를 실행하도록 결정하는 방식을 뜻하며, Concolic 테스팅의 성능에 중요한 기술요소로 알려져 있다. 기존의 Concolic 테스팅 연구에서는 제한된 시간 내에 최대한 다양한 경로를 실행하기 위해 높은 커버리지의 빠른 달성을 유도하는 휴리스틱 탐색 기법들[3,4,5]이 제시되었다.

Concolic 테스팅 탐색 기법 휴리스틱에 대한 많은 연구가 진행되었음에도 불구하고, 기존의 탐색 전략은 테스트 대상 프로그램 코드 혹은 테스트 케이스의 구조적 특성이 탐색 전략이 기반하는 가정에 맞지 않을 경우 테스트 효용성이 낮다는 한계점을 갖는다. 이로 인해, 특정 테스트 대상 프로그램에 대해서는 테스트 실행 시간이 많이 주어지더라도 한정된 경로 공간만을 탐색하는데 그쳐 커버리지 달성에 효과적이 못하는 상황이 자주 발생한다.

본 논문은 이와 같은 Concolic 테스팅 탐색 기법의 문제점을 효과적으로 해결하기 위해 Concolic 테스팅을 위한 다방향 탐색 전략(Multi-directional Search Strategy)을 제안하고, 이를 C 프로그램 대상 Concolic 테스팅 도구에 구현한 결과를 소개한다. 다방향 Concolic 테스팅 탐색 전략은 기존의 Concolic 테스팅이 단일한 실행에 대해서만 연쇄적으로 경로조건식 변경을 발생시켜 한 방향으로 탐색을 진행하는 것과 달리, 다양한 경로 공간으로 탐색 가능성이 있을 것으로 예상되는 복수의 실행을 저장하여 여러 방향으로의 탐색이 진행되도록 Concolic 테스팅 과정을 운영한다.

다방향 Concolic 테스팅 탐색 전략은 경로 선택 과정에서, 이전 입력이 실행 경로에 해당하는 분기의 반대 분기(alternative branch)가 아직 실행되지 않은 경우에, 해당 분기를 통해 새로운 경로 공간을 탐색할 수 있다고 판단하여, 해당 실행으로부터 출발하는 Concolic 탐색 방향을 신규로 생성하여 큐에 저장한다. 그리고 Concolic 테스팅 전반에 걸쳐 큐에 저장된 여러 개의 탐색 방향으로부터 일정한 횟수만큼 테스트 입력을 생성하고, 큐의 다른 탐색 방향으로 전환하는 방식을 통해 다방향 Concolic 탐색을 운용한다.

본 연구에서는 다방향 Concolic 탐색 전략의 분기 커버리지 달성의 효과성과 효율성을 확인하기 위해, C

본 논문의 연구는 한국연구재단의 지원(NRF-2020R1C1C1013512, NRF-2017M3C4A7068179)을 받아 수행됨.

```
01  func (int x, int y, int * arr) {
02    int i = 0, count = 0;
03    if (x==1) // then b1 else b1'
04      for (i = 0; i < 100; i++)
05        if (arr[i] == 0) // then b2 else b2'
06          count++;
07    else
08      if (y==1) // then b3 else b3'
09        abort();
10  }
```

Figure 1. Motivating Example

그림1. 국지적 탐색 문제가 발생할 수 있는 예제 코드

프로그램을 위한 Concolic 테스팅 툴인 CREST[6]에 포함된 2개의 탐색 전략인 CFG Directed Search과 Random Negation에 적용하였으며, 4가지 실제 프로그램에 (Vim, Grep, Busybox Awk, Busybox Sed)에 대해 실험하였다. 실험 결과, 다방향 탐색 전략이 기존 탐색 전략보다 더 높은 분기 커버리지(평균 4.1% 향상)를 더 빠르게 달성하는 것을 확인할 수 있었다.

2. 다방향 Concolic 탐색 전략

2.1 기존 Concolic 탐색 전략의 한계

기존의 Concolic 테스팅 탐색 전략에서 발생할 수 있는 국지적 영역에 탐색이 제한되는 문제는 그림 1에서의 예제 프로그램을 Random Negation 탐색 전략을 사용하는 상황을 통해 설명할 수 있다.

그림 1의 `func` 함수는 심볼릭 변수인 `x` 와 `y`, 사이즈가 100인 심볼릭 변수 배열인 `arr`를 입력으로 받는다. 목표로 하는 분기는 `abort()` 함수를 실행하는 `b3` 분기이다. 이 때 모든 변수의 초기값은 0으로 가정하겠다. 초기 입력에 의해 `b1'`, `b3'` 분기가 실행되며, 다음과 같은 심볼릭 조건식이 도출된다:

$$\phi_1 = (x \neq 1) \wedge (y \neq 1)$$

이 때 Random Search에 의해 50%의 확률로 첫 번째 조건절이 선택되었다고 가정하겠다. `x`의 값은 1로 업데이트되어 분기 `b2`가 실행되며 다음과 같은 심볼릭 조건식이 도출된다:

$$\phi_2 = (x = 1) \wedge (arr_0 \neq 0) \wedge (arr_1 \neq 0) \wedge \ldots \wedge (arr_{99} = 0)$$

목표로 하는 분기 `b3`은 반드시 분기 `b1'`이 먼저 실행되어야 하므로, 이를 위해 ϕ_2 의 첫번째 조건절이 탐색전략에 의해 선택되어야 한다. Random Negation 탐색 전략은 모든 경로조건식을 동일한 확률로 선택하므로, 약 99.0%의 확률로 첫 번째 조건절이 아닌 반복문에 해당하는 나머지 조건절들이 선택되게 된다. 즉, 반복문에 해당하는 조건절이 높은 확률로 다시 선택되게 되어 경로탐색이 국지적 공간에 제한되는 효과가 발생한다.

```
Input: target program P, initial path φ0
φ ← φ0; C ← Cov(φ0); Q ← empty queue
Q.add((φ, GetUniqueBranch(φ, Q)))
while ¬timeout()
  foreach (φ, U) ∈ Q
    for n ∈ [1, N]
      I ← NextInput(φ)
      φ ← Run(P, I)
      C ← C ∪ Cov(φ)
      U' ← GetUniqueBranch(φ, Q)
      if U' ≠ ∅ then
        Q.add((φ, U'))
      end if
    end for
  end foreach
end while
```

Figure 2. Multi-directional Concolic Search Algorithm

그림2. 다방향 Concolic 탐색 전략 알고리즘

2.2 다방향 Concolic 탐색 전략

본 논문에서는 Concolic 테스팅 과정에서 발생할 수 있는 국지적 탐색 문제를 완화하기 위해 여러 방향으로의 탐색이 진행되도록 하는 탐색 전략을 제안한다. 다방향 탐색전략에서 하나의 방향은 경로조건식과 목표 분기 집합의 순서쌍으로 정의되며, 탐색 과정에서 고유한 분기가 발견될 때 탐색 방향이 큐에 저장되어 각 방향의 경로 조건식으로 탐색을 시작하여 지정된 횟수만큼의 테스트 입력을 생성하게 된다.

그림1의 예제 코드 경로 탐색에서 발생할 수 있었던 국지적 탐색 문제는 다방향 탐색 알고리즘을 적용하여 효과적으로 해결됨을 보일 수 있다. 2.1장에서와 같이 함수 `func`의 초기 입력으로 심볼릭 변수 `x` 와 `y`, 사이즈 100인 심볼릭 변수 배열 `arr`가 입력되며, 모든 변수의 초기값은 0으로 가정하겠다. 이때 도출되는 조건식은 다음과 같다:

$$\phi_1 = (x \neq 1) \wedge (y \neq 1)$$

제안하는 다방향 Concolic 탐색 전략은 실행한 분기의 반대 분기인 `b1`, `b3`가 이전에 실행된 적이 없으므로, 해당 Concolic 탐색 방향을 큐에 저장한다. 현재 방향의 테스트 입력 생성 횟수가 종료된 이후 해당 방향의 순서가 되어 ϕ_1에서 시작하는 탐색이 시작되어 입력이 생성된다. 즉, 해당 경로를 도출하는 입력이 다시 생성되어 실행되지 않더라도, 미리 큐에 저장했던 방향에서 시작하는 탐색을 통해 `b3`을 실행하는 경로가 생성될 수 있게 된다.

그림 2는 기본적인 Concolic 테스팅 알고리즘의 구조[3]를 바탕으로, 본 논문이 제안하는 다방향

Table 1. The average number of branches covered at the end of test generation
표1. 분기 커버리지 달성(개수)

Target Program	Search Strategy	Base	Multi-Directional
Vim	CFG	8610.5	**9312.9**
	Random Negation	9113.2	**9946.9**
Grep	CFG	2288.0	**2349.9**
	Random Negation	1978.4	**2035.6**
Awk	CFG	496.2	**512.1**
	Random Negation	**465.6**	458.2
Sed	CFG	505.0	**508.6**
	Random Negation	473.0	**521.2**

Concolic 테스팅 알고리즘을 나타낸 알고리즘이다. Concolic 테스팅은 입력으로 검증대상 프로그램(P)과 초기 실행 경로(p)를 받는다. 전체 탐색 기간 동안 달성한 분기들의 집합(C)을 입력된 경로가 실행하는 분기를 반환하는 함수 Cov를 통해 초기화 한다. 그리고 방향을 저장하는 큐(Q)를 초기화한다(1행). 그리고 초기 입력의 실행을 통해 생성된 방향을 Q에 추가한다. 함수 GetUniqueBranch는 파라미터로 입력된 경로가 실행하는 분기의 반대 분기(alternative branch)들 중 큐에 포함된 방향들의 타겟 분기들에 대해 고유한 분기들의 집합을 반환한다. 실행한 분기의 타겟 분기를 GetUniqueBranch 함수를 실행하여 초기 실행 방향을 추가한다(2행).

다방향 Concolic 탐색은 탐색 시간이 종료시간에 도달할 때까지 Q에 있는 모든 방향을 순회하며 일정한 횟수 (N)만큼의 테스트 입력을 생성한다(3~15행). 각각의 테스트 입력 생성마다 가장 먼저 Q에서 시작하는 경로 조건식이거나 앞서 실행되었던 경로조건식 ϕ에 대해 기반 탐색전략을 사용하여 다음 입력을 생성한다(4행). 그리고 생성된 입력으로 타겟 프로그램을 실행하여 경로 조건식을 추출하고 달성한 분기를 C 에 추가한다(5~6행). 다음으로 앞서 실행한 입력이 달성하는 고유한 분기의 집합(U')을 도출하여 고유한 분기가 있다면 새로운 탐색 방향을 Q에 추가한다(9~12행).

3. 실험을 통한 평가

3.1 실험 설계

연구 질문. 제안한 Concolic 테스팅 탐색 전략으로 인한 효용성과 효과성 향상을 평가하기 위해서 다음의 두 가지 연구 질문에 대한 실험을 수행하였다:

- RQ1. 기반 Concolic 탐색 전략을 사용한 경우에 비하여, 해당 탐색 전략에 대해서 다방향 Concolic 탐색 전략을 적용하는 경우 분기 커버리지의 달성이 얼마나 향상되는가?
- RQ2. 기반 Concolic 탐색 전략을 사용한 경우에 비하여, 해당 탐색 전략에 대해서 다방향 Concolic 탐색 전략 최대 반복 횟수를 제한한 탐색 전략을 추가로 적용할 때 경우, 높은 분기 커버리지를 달성하는 데 소요되는 테스트 입력 값 생성 시간이 얼마나 단축되는가?

Table 2. The average time required to achieve 90% of the maximum branch coverage at the end of test generation (second)
표2. 최대 분기 커버리지의 90%달성 소요시간(초)

Target Program	Search Strategy	Base	Multi-Directional
Vim	CFG	1198.2	**751.5**
	Random Negation	1176.2	**589.2**
Grep	CFG	594.5	**297.9**
	Random Negation	83.5	**65.2**
Awk	CFG	9.8	**9.5**
	Random Negation	8.3	**8.0**
Sed	CFG	35.9	**34.7**
	Random Negation	526.5	**18.2**

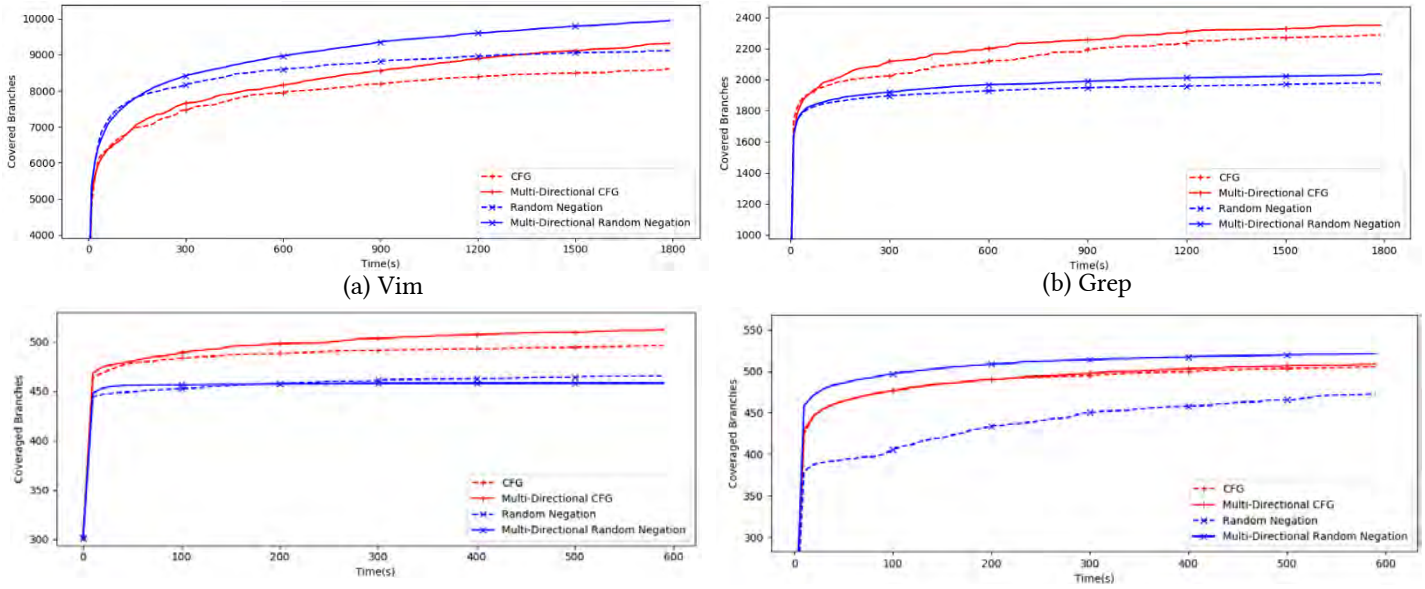
Concolic 테스팅 탐색 기법. 본 실험에서는 기반 탐색 알고리즘으로 기존 연구[3]에서 제안한 CFG Directed Search, Random Negation Search을 사용하였다. 그리고 각 기반 탐색 알고리즘에 본 논문에서 다방향 탐색 전략을 적용한 Multi-Directional CFG, Multi-Directional Random Negation을 구현하여, 총 4개의 서로 다른 기법 간의 결과를 비교하였다. 본 실험은 Concolic 테스팅 도구인 CREST[6]를 SMT해법기로 Z3 4.8.4[7]를 사용하도록 수정한 도구인 CRESTIVE[1]를 이용하였다.

테스트 대상 프로그램. 본 실험에서는 기존 연구[3]의 실험에서 사용한 Grep 2.2(13339 LoC, 7289 분기)와 Vim 5.7(144577 LoC, 90644 분기), Busybox version 1.30.1[8]의 Awk(26733 LoC, 8072분기)와 Sed(25220 LoC, 6237 분기)를 테스트 대상으로 사용했다. 각 테스트 대상에 대해 사용자 명령 및 파일 입력을 심볼릭 입력 변수로 정하는 심볼릭 드라이버를 각각 정의하여 사용하였으며, 심볼릭 입력 변수의 개수는 Vim의 경우 20개, Grep은 경우 33개, Awk는 35, Sed는 40개를 사용하였다.

실험 수행 및 측정. 반복 횟수 제한 탐색 전략의 사용에 따른 커버리지 달성 효과성(RQ1)과 효율성(RQ2)을 평가하기 위해 총 4개의 탐색 전략을 각각 Concolic 테스팅에 적용하여 Vim은 과 Grep은 1800초, Awk와 Sed는 600초 간 테스트 입력 생성을 수행했다. 테스팅 시간은 사전조사를 통해 시간 연장이 결과에 유의미한 차이를 보이지 않는 지점을 선택하였다.

탐색 전략에 따른 커버리지 달성 효과성(RQ1)을 평가하기 위해, 최종적으로 생성된 테스트 입력이 한 번이라도 커버한 분기의 수를 측정하였다(서로 다른 실행에서 같은 분기를 커버한 경우, 중복해서 계수하지 않음). 또한, 효율성(RQ2)을 비교하기 위해, 각 탐색 전략 별로 최종적으로 커버한 최대 분기 개수의 90%를 달성하기까지 소요된 시간의 평균을 측정했다. 예를 들어, Vim에 대한 Multi-Directional CFG와 CFG 간의 효율성 비교를 위해, 둘 중 최종적으로 더 많은 분기를 커버한 Multi-Directional CFG의 결과인 9312.9의 90% 지점, 즉 8381.6이 달성되는 데 Multi-Directional CFG와 CFG가 각각 소요한 테스팅 시간을 구했다. 본 실험에서 사용한

[1] CRESTIVE. https://github.com/arise-handong/crestive



(a) Vim (b) Grep

(c) Awk (d) Sed

Figure 3. Branch coverage achievement over testing time

그림 3. Vim, Grep, Awk, Sed의 분기-커버리지 그래프

기반 탐색 전략에는 난수 발생 요소가 있으므로 동일한 실험을 30회 반복한 결과의 평균을 구했다.

3.2. 실험 결과

RQ1: 커버리지 달성 효과성. 표1은 총 4개의 테스팅 기법이 최종적으로 달성한 분기 개수를 나타낸다. Baseline은 다방향 탐색 전략을 적용하지 않은 경우이며, Multi-Direction은 본 논문이 제안한 다방향 탐색 전략을 적용한 것을 각각 나타낸다. 표1의 결과에 따르면, 다방향 Concolic 탐색 전략을 적용한 경우(Multi-Directional)가 그렇지 않은 경우(Baseline)보다 더 많거나 동일한 수준의 분기 커버리지를 달성한다. Vim의 경우, 모든 경우에서 반복 횟수 제한 탐색 기법이 기반 탐색 기법보다 최대 9.2% (Random) 높은 커버리지를 달성했다. Grep의 경우, Multi-Direction CFG, Multi-Direction Random Negation이 기반 탐색 전략에 대비하여, 각각 2.7%, 2.9% 많은 분기를 커버하였다. Awk의 경우, CFG는 3.2%, Random의 경우, Baseline이 1.6% 더 많은 분기를 커버하였다. Sed의 경우에는 CFG는 0.7%, Random은 10.2% 더 많은 분기를 커버하였다.

RQ2: 커버리지 달성 효율성. 표2는 표1에서의 탐색 전략 별 최대 분기 커버리지의 90%를 달성하기까지 걸린 테스트 생성 시간이다. 표2는 모든 테스트 대상 프로그램에서 다방향 Concolic 탐색 전략을 적용한 경우(Multi-Direction)에 그렇지 않은 경우(Baseline)보다 더 짧은 시간 내에 90% 지점을 달성하여 (평균 25.5%의 시간 단축) 다방향 Concolic 탐색 전략이 높은 분기 커버리지를 더 빠르게 달성하였다.

그림3은 실험대상인 4가지 테스트 대상 프로그램에서 시간에 따라 분기 커버리지 달성을 나타낸 그래프이다. 다방향 Concolic 탐색 전략(Multi-Directional)은 실선으로, 비교 대상인 기반 탐색 전략(Base)은 점선으로 표현하였다. 테스팅 전반에 걸쳐 다방향 Concolic 탐색 전략이 기반 탐색 전략에 비해 커버리지를 더 높게 달성하며, 테스팅 시간 추가에 따라 지속적인 커버리지 증가를 달성하는 경우가 있음을 확인할 수 있다.

4. 결론 및 향후 연구

본 논문에서는 다양한 경로 공간으로 탐색 가능성이 있을 것으로 예상되는 복수 개의 경로를 번갈아 가며 탐색함으로써 동시에 여러 방향으로의 탐색이 진행되도록 Concolic 테스팅 과정을 운영하는 다방향 Concolic 탐색전략에 대해 제안하였다. 기존에 개발되었던 탐색전략인 CFG, Random Negation에 제안한 탐색 기법을 적용한 결과, 분기 커버리지 달성의 효과성과 효율성이 향상되는 것을 실험적으로 확인할 수 있었다. 향후에는 방향 별로 입력 생성 횟수를 가변적으로 스케쥴링하여 탐색 전략의 분기 커버리지 달성 성능을 높이는 연구와, 제안한 탐색 전략을 CGS[4], CarFast[5] 등 다른 탐색 휴리스틱에 복합적으로 적용하는 연구를 수행할 계획이다.

참조문헌

[1] R. Kannavara, C. J. Havlicek, B. Chen, M. R. Tuttle, K. Cong, S. Ray, F. Xie, Challenges and Opportunities with Concolic Testing, NAECON 2015

[2] K. Sen, D. Marinov, G. Agha, CUTE: a concolic unit testing engine for C, ESEC/FSE, 30(5), 263-272, 2005

[3] J. Burnim, K. Sen, Heuristics for Scalable Dynamic Test Generation, ASE 2008

[4] H. Seo, S. Kim, How We Get There: A Context-Guided Search Strategy in Concolic Testing, FSE 2014

[5] S. Park, B. M. Hossain, I. Hussain, C. Csallner, M. Grechanik, K. Taneja, Q. Xie, CarFast: achieving higher statement coverage faster, FSE, 2012

[6] CREST, https://github.com/jburnim/crest

[7] Z3, https://github.com/Z3Prover/z3

[8] Busybox, https://busybox.net

데이터 증강 기법을 활용한 심층 신경망 강건성 평가 방법

최영원, 이영우, 채흥석
부산대학교 정보컴퓨터공학부
fenrir92@pusan.ac.kr amorepooh@pusan.ac.kr hschae@pusan.ac.kr

Method to Estimate the Robustness of Deep Neural Networks with Data Augmentation

Young-Won Choi, Youngwoo Lee, Heung-Seok Chae
School of Computer Science & Engineering, Pusan National University
fenrir92@pusan.ac.kr amorepooh@pusan.ac.kr hschae@pusan.ac.kr

요 약

심층 신경망이 발달함에 따라 여러 도메인에서 DNN을 접목하고 있으며, 이미지 분류 모델의 경우 자율주행자동차를 비롯한 안전필수시스템에도 사용되고 있다. 그러나 이미지 분류 모델은 적대적 공격과 데이터 증강에 대하여 취약함을 드러내고 있으며, 해당 공격 기법에 대해 이미지 분류 모델이 강건성을 가지도록 여러 기법들이 연구되었다. 기존 DNN의 강건성에 대한 연구는 적대적 공격 기법과 그에 대한 방어 기법이 주를 이루고 있다. 본 논문에서는 DNN의 강건성에 관한 논문들을 Attack과 Metric으로 분류하고, 데이터 증강 기법을 활용한 DNN의 강건성 테스트 방법을 제안한다.

1. 서 론

심층 신경망(DNN)이 발달함에 따라 사물인식과 음성인식 등 여러 분야에서 DNN을 접목하고 있으며, 자율주행자동차를 비롯한 안전필수시스템에도 이미지 분류 모델이 사용되고 있다. 그러나 이미지 분류 모델은 적대적 공격[1]과 데이터 증강[2]을 적용한 변형된 데이터에 대해 취약점을 드러내고 있다[1, 2, 3]. 안전필수시스템이 DNN의 취약점으로 공격을 받는 경우 막대한 금전적 손실이 발생한다. 그러므로 적대적 공격과 데이터 증강에 대한 DNN의 취약점을 개선할 방법이 필요하다.

적대적 공격과 데이터 증강 기법을 적용한 데이터에 대해 이미지 분류 모델이 강건성을 가지도록 여러 방어 기법들이 연구되었다. 이미지 분류 모델이 탐지하기 어려운 적대적 공격 기법을 고안하거나[3, 4, 5, 6, 7], 변형된 데이터에 대해 DNN이 강건성을 가지도록 하는 방어 기법[8, 9, 10, 11], DNN의 강건성을 정량적으로 측정하는 메트릭[12, 13, 14, 15]에 대하여 연구가 진행되었다.

DNN의 강건성에 대한 연구에서 데이터 증강 기법은 적대적 공격보다 적게 활용되었다. 적대적 공격은 DNN의 취약점을 공격하기 위해 인공적인 데이터를 생성하지만, 밝기 변화, 회전, 등 데이터 증강 기법들은 현실에서 발생할 수 있는 이미지를 생성한다. 또한 밝기 변화의 경우 이미지 내 전체 픽셀의 값이 변경되지만 모델이 분류할 원본 이미지 내 객체 형태를 유지한다. 따라서 이미지 분류 모델은 적대적 공격 기법을 적용하여 미세하게 변화된 이미지와 데이터 증강 기법을 적용하여 크게 변경되면서 객체의 형태를 유지하는 이미지 모두에 대해 강건성을 가져야 한다.

본 논문에서는 DNN의 강건성과 관련된 기존 논문에서 강건성을 측정하기 위해 사용하는 방법들을 Attack, Tolerance, Metric 3가지 항목으로 분류한다. 그리고 DNN의 강건성 테스트 방법으로 데이터 증강을 사용한 이미지 생성 방법을 제안한다.

2. Robustness of DNN

본 절에서는 기존 DNN의 강건성에 대한 연구들을 Attack과 Metric으로 분류한다. Attack은 DNN이 강건성을 가지는지 확인하기 위한 이미지 변형 기법으로, Attack은 Adversarial Attack(적대적 공격)과 Data Augmentation(데이터 증강) 2가지 세부 항목으로 분류한다.

Adversarial Attack은 이미지 데이터의 미세한 변화로 이미지 분류 모델이 올바르게 분류하지 못하도록 하는 공격이다. 모델이 이미지를 제대로 분류하지 못하면서 사람도 이미지의 변화를 인지하지 못해야 하므로 적용 시 이미지 값 변화량을 고려한다.

Data Augmentation은 밝기 조절, 회전, 노이즈 적용 등 이미지 변형 기법들로, 모델의 다양한 학습 데이터를 생성하기 위해 적용하지만[2], 테스트 데이터 생성에도

표 1. Attack과 Metric에 따른 기존 연구 분류

Attack \ Metric		NC	KMNC	NBC	Accuracy	Success Probability	CLEVER	mCE
Adversarial Attack	FGSM	-	[13]	[13]	[9]	-	[14]	
	JSMA	-	[13]	[13]	[9]	-	-	-
	C&W	-	[13]	[13]	[9]	[7]	[14]	-
	BIM	-	[13]	[13]	[9]	-	-	-
	DeepFool	-	-	-	[9]	-	-	-
Data Augmentation	Brightness	[12]	-	-	-	-	-	-
	Noise	-	-	-	-	-	-	[15]

사용된다. 주로 단일 기법을 이미지에 적용하지만 2가지 이상 기법들을 조합하여 적용하기도 한다. Data Augmentation은 Adversarial Attack과는 달리 이미지의 변화량에 대해 고려하지 않는다.

Metric은 DNN의 강건성을 측정하는 척도이다. Attack 기법을 적용한 이미지에 대해 정확도가 높다는 것은 모델의 취약점을 공격한 이미지를 올바르게 분류한다는 의미이므로 모델의 정확도를 모델의 강건성 메트릭으로 사용한다[9]. 소프트웨어 공학의 테스트 기법들에 영감을 받아 만들어진 메트릭도 존재한다[12, 13].

표 1은 기존 DNN의 강건성 연구를 Attack 기법과 Metric으로 분류한다. Attack은 Adversarial Attack과 Data Augmentation으로 나누어 분류한다. 실험에 따라 Tolerance를 적용하지 않거나, Metric으로 성능 지표인 정확도를 사용한다.

N. Carlini and D. Wagner[7]는 사람이 인지하기 어렵도록 이미지를 최소한으로 변경하기 위해 원본 데이터와의 데이터 변화량을 최소화하는 이미지를 생성하는 적대적 공격 기법을 제안하였다. 이미지 분류 Softmax 값을 이미지 클래스 분류 결과값으로 학습시키는 Defensive Distillation(DD)[8]을 적용한 모델을 대상으로 실험하여 생성한 적대적 공격 데이터들이 DNN 공격에 성공하여 잘못 분류된 확률 Success Probability를 측정하였다.

K. Pei et al.[12]는 DNN 내부 뉴런의 Activation Value를 통해 뉴런 활성화 정도를 Neuron Coverage(NC)를 측정하여 강건성. 밝기 조절 등 데이터 증강 기법들을 적용한 이미지 데이터를 여러 모델에 입력하여 정확도와 Neuron Coverage를 측정하였다.

L. Ma et al.[13]는 학습 시 측정된 DNN 내부 각 뉴런의 활성화값 범위를 k개로 나누어 커버리지를 측정하는 k-multisection Neuron Coverage(KMNC)와 경계값 분석처럼 학습 시 뉴런 활성화값 범위 바깥 구간을 테스트 시에 커버리지 범위로 측정하는 Neuron Boundary Coverage(NBC) 등을 제안하였다. FGSM과 BIM, JSMA, C&W를 적용한 데이터를 사용하여 제안한 커버리지 매트릭을 측정하여 비교하였다.

T. W. Weng et al.[14]은 DNN의 강건성을 측정하기 위한 매트릭으로 적대적 공격을 적용한 이미지의 변화량 최소 L_p Norm 거리의 Lower Boundary를 모델의 강건성 지표로 사용하는 CLEVER Score를 제안하였다. Defensive Distillation을 적용한 CNN과 Bounded ReLU(BReLU)[10]의 CLEVER Score를 측정하기 위해 FGSM과 C&W를 적용한 데이터를 사용하였다.

W. Xu et al.[9]은 적대적 공격을 적용한 이미지를 탐지하는 방어 기법으로 색상 표현 범위를 줄이는 방법과 이미지를 흐릿하게 하는 방법을 적용하여 이미지 식별에 필요하지 않는 특성을 줄이는 Feature Squeezing을 제안하였다. Feature Squeezing을 적용한 DNN에 각각 FGSM[3], BIM[4], DeepFool[5], JSMA[6], C&W[7]를 적용한 이미지 데이터를 입력하여 정확도를 측정하였다.

R. G. Lopes et al.[11]은 데이터 증강을 적용한 데이터 학습 시 발생하는 정확도와 강건성 간의 반비례 관계를 극복하기 위해 Cutout[16]과 Gaussian Noise 2가지 데이터 증강 기법을 조합한 Patch Gaussian을 제시하였다. Patch Gaussian을 적용한 데이터로 학습한 모델에 Gaussian Noise를 적용한 이미지를 입력하여 정확도와 mean Corruption Error(mCE)[15]를 측정했다.

기존 연구에서는 데이터 증강에 대한 DNN의 강건성 테스트가 부족하다. 소프트웨어를 대상으로 동적 테스트를 수행할 때 소프트웨어의 구조와 테스트 방식, 테스트 단계에 따라 취약점을 공격할 테스트 데이터를 생성한다. 그러나 기존 DNN의 강건성 연구에서는 DNN에 대한 적대적 공격 기법들이 제안된 후, 적대적 공격 기법에 대응하는 방어 기법과 측정할 수 있는 메트릭을 고안한다. 그로 인해 DNN의 강건성 테스트를 수행할 때 데이터 증강 기법보다 적대적 공격 기법들이 더 많이 적용되고 있다. DNN의 강건성 테스트를 위한 테스트 데이터 생성 시 적대적 공격 기법은 C&W와 FGSM, JSMA 등 특정 기법들이 사용되고 있는 반면, 데이터 증강 기법은 DNN의 강건성 테스트에 적합한 기법이 제시되지 않고 있다.

적대적 공격 기법들은 사람이 인지하지 못하는 미세한 이미지 변화를 적용한 이미지를 생성한다. 반면에 데이터 증강 기법은 이미지 변화가 확연히 표현되지만 이미지 내 객체의 형태를 유지하고,

현실적으로 발생할 수 있는 이미지를 생성한다. 밝기 변화, 날씨 효과 등 기법들은 현실에서 발생할 수 있는 이미지를 생성한다. 노이즈가 존재하는 이미지 또한 데이터 통신 상태에 따라서 발생한다. 그러므로 DNN의 강건성 테스트를 위한 테스트 데이터 생성에 데이터 증강 기법을 활용해야 한다.

이미지 분류 모델은 데이터 증강 기법을 적용한 이미지의 변화값이 크더라도 객체의 형태가 유지되면 올바르게 분류해야 하며, 강건성 테스트 시 적대적 공격을 적용한 데이터뿐만 아니라 데이터 증강을 적용한 데이터에 대한 모델의 강건성도 측정해야 한다. 따라서 데이터 증강 기법을 활용한 DNN의 강건성 테스트가 필요하다.

3. Robustness with Data Augmentation

본 논문에서는 DNN의 강건성을 테스트하기 위해 데이터 증강을 적용한 DNN의 강건성 테스트 제안한다. 데이터 증강 기법으로는 밝기 조절, 회전, 좌우/상하 반전, 노이즈, 날씨 효과 등 이미지 내 객체의 형태를 유지하는 기법들을 적용한다.

테스트 데이터를 다양하게 하기 위해 데이터 증강 기법을 1개 적용하는 단일 데이터 증강과 2가지 기법을 합성하여 적용하는 합성 데이터 증강을 적용한다. 데이터 증강 기법 2개를 조합한 합성 데이터 증강은 이미지 변화량이 더 커지지만 다양한 이미지들을 생성할 수 있다. 그림 1은 합성 데이터 증강을 적용하여 테스트 데이터를 생성하는 과정을 나타낸다. td0은 원본 테스트 데이터이며, td1, td2, td3은 각각 단일 데이터 증강 기법 DA1, DA2, DA3을 적용하여 생성된 데이터이다. Td4와 td5는 각각 원본 데이터에 DA1과 DA2의 합성 데이터 증강 DA4를, DA2와 DA3의 합성 데이터 증강 DA5를 적용하여 생성된 데이터이다.

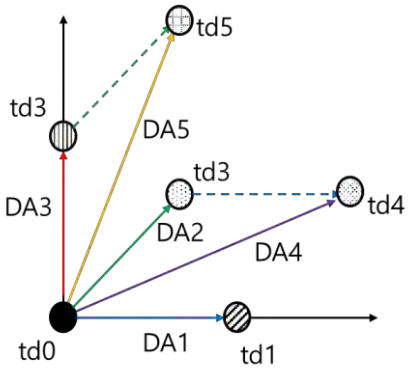


그림 1. 합성 데이터 증강 테스트 데이터 생성 과정

데이터 증강 기법 중 밝기 조절, 노이즈 등 기법들은 극단적인 강도를 적용하는 경우 객체의 형태가 사라질 수 있다. 합성 데이터 증강 또한 2가지 데이터 증강 기법을 적용함으로 인해 객체의 형태가 크게 손상될 수 있다. 따라서 데이터 증강 기법으로 이미지를 생성할 때 이미지 변화량이 사람이 육안으로 객체를 올바르게 분류할 수 있는 최대 변화량 이하가 되도록 데이터 증강 기법 별로 적용 세기를 설정해야 한다.

데이터 증강을 적용한 이미지가 사람이 구분하기 힘든 강도로 적용하는 것을 예방하기 위해 이미지 변화량을 측정해야 한다. 이미지 변화량을 측정하는 매트릭으로 SSIM[17]과 L_p Norm 거리(L_p Norm Distance)가 존재한다. SSIM은 휘도, 대비, 구조 3가지 측면에서 두 이미지를 비교하여 이미지 간 유사도를 계산하며 값이 클수록 두 이미지가 유사함을 나타낸다. L_p Norm 거리는 p 값에 따라서 두 이미지 간의 거리를 계산하며, C&W의 이미지 생성과 CLEVE Score 측정에 활용된다. C&W는 원본 데이터와의 데이터 변화를 L_p Norm 거리로 측정하고, L_p Norm 거리를 최소화하는 이미지를 생성한다[7]. CLEVER Score는 이미지 변화량 L_p Norm 거리 최소값의 Lower Boundary를 모델의 강건성 지표로 사용한다[14]. 기존 연구와 이미지 변화량 비교를 위해서 이미지 변형 시 값 변화량을 L_p Norm 거리로 측정하도록 한다.

4. 결론

DNN의 강건성에 대한 사례 연구로 DNN에 대한 공격 기법과 방어 기법, 강건성 측정 매트릭을 제안한 논문들을 선정하여 각 논문에서 활용한 기법들을 Attack, Tolerance, Metric으로 분류해 보았다. Attack은 Adversarial Attack과 Data Augmentation 두 세부항목으로 분류하였다. DNN의 강건성에 대한 연구는 적대적 공격 위주로 연구가 진행되었고 데이터 증강에 대한 강건성 테스트가 부족하였다. 적대적 공격 기법은 DNN 테스트를 위한 기법들이 정형화되었지만, 데이터 증강은 DNN의 강건성 테스트를 기법이 부족하다. 데이터 증강은 적대적 공격과는 달리 이미지 변화가 확연히 드러나지만, 이미지 내 객체의 형태는 유지한다. 또한 일부 데이터 증강 기법은 현실적으로 생성될 수 있는 이미지를 생성한다. 따라서 이미지 분류 모델은 적대적 공격과 데이터 증강 두 방식에 대해 강건해야 한다.

본 논문에서는 데이터 증강을 활용한 DNN 모델 강건성 테스트 방법을 제안하였다. 이미지 내 객체의 형태를 유지할 수 있는 데이터 증강 기법들을 적용하며, 다양한 이미지를 생성하기 위해 2가지 데이터 증강 기법을 적용하여 테스트 이미지를 생성하도록 한다. 그리고 데이터 증강 기법이 이미지 내 객체의 형태를 손상시키지 않도록 이미지 변화량을 측정하여 일정 이미지 변화량 이내의 데이터를 생성하도록 해야 한다.

참고 문헌

[1] C. Szegedy et al., “Intriguing Properties of Neural Networks,” in *International Conference*

on Learning Representation, 2013.

[2] C. Shorten and T. M. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 60, 2019.

[3] I. J. Goodfellow et al., "Explaining and Harnessing Adversarial Examples," in *International Conference on Learning Representation*, 2015.

[4] A. Kurakin et al., "Adversarial Examples in the Physical World," in *International Conference on Learning Representation*, 2015.

[5] S. M. Moosavi-Dezfooli et al., "DeepFool: a Simple and Accurate Method to Fool Deep Neural Networks," in *Proceedings of the IEEE Conference of Computer Vision and Pattern Recognition*, pp. 2574-2582, 2016.

[6] N. Papernot et al., "The Limitations of Deep Learning in Adversarial Settings," *IEEE European Symposium on Security and Privacy*, pp. 372-387, 2016.

[7] N. Carlini and D. Wagner, "Towards Evalutationg the Robustness of Neural Networks," *IEEE Symposium on Security and Privacy*, pp. 39-57, 2017.

[8] N. Papernot et al., "Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks," *IEEE Symposium on Security and Privacy*, 2016.

[9] W. Xu et al., "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks," *arXiv preprint arXiv:1511.03034*, 2017.

[10] V. Zantedeschi et al., "Efficient defenses against adversarial attacks," *arXiv preprint arXive:1707.06728*, 2017.

[11] R. G. Lopes et al., "Improving Robustness Without Sacrificing Accuracy with Patch Gaussian Augmentation," *arXiv preprint arXiv:1906.02611*, 2019.

[12] K. Pei et al., "DeepXplore: AutomatedWhiteBox Testing of Deep Learning Systems," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 1-18, 2017.

[13] L. Ma et al., "DeepGauge: Multi-Granularity Testing Criteria for Deep Learning Systems," The 33rd IEEE/ACM International Conference onf Automated Software Engineering, 2018.

[14] T. W. Weng et al., "Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach," in *International Conference on Learning Representations*, 2018.

[15] N. Ford, "Adversarial Examples Are a Natural Consequence of Test Error in Noise," *arXiv preprint arXiv:1901.10513*, 2019.

[16] T. DeVries et al., "Improved Regularization of Convolutional Neural networks with Cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[17] Z. Wang et al., "Image Quality Assessment: from Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, 2004.

비상호적 차분 프라이버시 모델의 노이즈 파라미터 분배 기법

김성민 O, 이원석*

O 연세대학교 컴퓨터과학과

*연세대학교 컴퓨터과학과

k620314@naver.comO, leewo2001@gmail.com*

Noise parameter distribution technique of non-interactive differential privacy model

Sung-Min KimO , Won-Suk Lee*

ODept. of Computer Science, Yonsei University

* Dept. of Computer Science, Yonsei University

요 약

비상호적 차분 프라이버시 모델은 원본 데이터 셋에 노이즈를 삽입해 안전한 배포용 데이터 셋을 만드는 개인정보 보호 기법이다. 본 연구에서는 원본 데이터 셋을 속성별로 분리해 차분 프라이버시를 적용할 때 필요한 노이즈 파라미터 분배 방식을 제안한다. 노이즈 파라미터를 각 속성에 분배하는 여러 방식을 제시하고 실험을 통해 그 결과를 비교함으로써 비상호적 차분 프라이버시 모델의 데이터 유틸리티를 향상시키고자 한다.

1. 서 론

차분 프라이버시 모델은 데이터베이스 상에서 수행되는 질의 결과에 의한 개인정보 재식별을 방지하기 위해 그 결과에 노이즈를 추가하는 개인정보 보호 기법[1]이다. 개인정보 재식별 위험성을 수학적으로 정의했다는 점에서 많은 반향을 불러일으켰으며 최근 활발한 연구가 이루어지고 있다.

차분 프라이버시 모델의 한 종류인 비상호적 차분 프라이버시 모델에서는 데이터 보유자가 원본 데이터 셋 자체에 차분 프라이버시 기반의 노이즈를 삽입해 결과 데이터 셋을 만든다. 이렇게 생성된 결과 데이터 셋은 개인정보 재식별이 불가능한 것으로 간주되어 제3자에게 공유 및 배포가 가능하다. 하지만 비상호적 차분 프라이버시 모델로 생성된 결과 데이터 셋에는 지나치게 많은 노이즈가 삽입되어 데이터 유틸리티가 떨어진다는 문제점이 제기되었다. 데이터 유틸리티를 향상시키는 방향으로 여러 연구가 진행되어 왔지만 여전히 실제로 배포하기에는 데이터 유틸리티가 낮다는 한계점[2]이 있다.

2. 관련 연구

초창기 비상호적 차분 프라이버시 모델[3]은 n개의 레코드를 갖는 원본 데이터 셋에 대해 레코드 선택 질의를 n번 적용해 결과 데이터 셋을 만들었다. 원본 데이터 셋에 정렬과 같은 별도의 전처리 과정을 거치지 않았기 때문에 레코드 순서에 대한 기준이 없었다. i번째로 선택한 레코드와 (i+1)번째로 선택한 레코드 사이에 유사성이나 연관 관계가 존재하지 않았고 이는 과도한 노이즈가 삽입되는 결과로 이어졌다. 이후 진행된 비상호적 차분 프라이버시 연구들[4][5][6]은 레코드에 클러스터링을 수행해 유사한 레코드끼리 모으고 각 클러스터에 차분 프라이버시를 적용하는 방식을 사용하였다.

초기 클러스터링 기반 비상호적 차분 프라이버시 모델[4]은 다변량(multivariate) 선택 질의를 사용하였다. 이는 질의의 결과가 레코드 단위인 것으로 속성이

m개이고 레코드가 n개인 결과 데이터 셋을 만들기 위해서는 총 n번의 질의를 사용해야 했다.

이후 진행된 연구는 단변량(univariate) 선택 질의를 사용[5][6]하였다. 질의의 결과가 레코드의 한 속성 값으로, 속성이 m개이고 레코드가 n개인 결과 데이터 셋을 만들기 위해서는 총 m*n번의 질의를 사용해야 했다. 단변량 방식으로 만들어진 결과 데이터 셋의 데이터 유틸리티가 다변량 방식의 데이터 유틸리티보다 좋다는 것이 실험을 통해 입증되어 최근 연구들은 단변량 방식을 통해 진행되고 있다.

지금까지 발표된 비상호적 차분 프라이버시 모델들 중 가장 데이터 유틸리티가 좋은 것은 2018년에 발표된 '최적 단변량 마이크로집계(optimal univariate microaggregation)'[6]로 단변량 방식을 사용하고 있다. 위 연구에서는 원본 데이터 셋을 속성별로 분리하고 각각을 정렬한 뒤 그래프 알고리즘을 사용해 클러스터링한다. 각 속성 값들을 그래프의 노드로 나타내고 어떤 두 속성 값이 한 클러스터에 포함되었을 때 그 클러스터의 SSE(Sum of Square Error)를 두 속성 값에 해당하는 노드 간 간선의 길이로 나타낸다. 이후 그래프 상에서 가장 작은 속성 값 노드에서 가장 큰 속성 값 노드까지 이동하는 최소 비용 경로를 찾으면 그것이 곧 최적 클러스터링 결과에 해당한다. 이렇게 구해진 각 클러스터에 대해 클러스터 내의 레코드들을 클러스터의 중심값으로 치환하고 차분 프라이버시를 적용해 노이즈를 삽입한다.

3. 제안하는 기법

차분 프라이버시 모델에서는 개인정보 재식별 기준을 노이즈 파라미터 ϵ를 통해 결정한다. 노이즈 파라미터가 작다는 것은 차분 프라이버시의 기준이 엄격함을 의미한다. 이 경우 개인정보 재식별을 피하기 위해 큰 노이즈가 삽입되는 경향을 보인다. 반대로 노이즈 파라미터가 크다는 것은 차분 프라이버시의 기준이 낮다는 것으로, 상대적으로 작은 노이즈가 삽입된다. 개인정보 재식별 위험성과 데이터 유틸리티를 균형 있게 만족시키는 최적 노이즈 파라미터 값이 얼마인지에 대한 연구 결과는 아직 나온 바가 없지만 통상적으로 0.1 이상 10 이하의 값을 노이즈 파라미터 값으로 사용한다.

레코드를 속성별로 분리하는 단변량 방식에서는 한 레코드를 만들기 위해 속성의 개수만큼 차분 프라이버시를 적용해야 한다. 이 경우 차분 프라이버시의 순차 정리(sequential theorem)에 의해 각 속성에 차분 프라이버시를 적용할 때 사용되는 속성별 노이즈 파라미터들이 총합이 데이터 셋에 대한 노이즈 파라미터와 같아야 한다. 즉, 노이즈 파라미터를 각 속성마다 분배해야 하는 것이다. 이 노이즈 파라미터 분배는 결과 데이터 셋의 데이터 유틸리티에 큰 영향을 미칠 수 있지만 이에 대한 연구는 진행되지 않고 있다. 노이즈 파라미터를 각 속성에 균등하게 분배하는 균등 분배 방법[5]과 각 속성의 도메인 크기에 비례하여 분배하는 도메인 크기 비례 분배 방법[5][6] 두 가지만 사용되었다.

본 논문에서는 노이즈 파라미터를 각 속성에 분배하는 여러 방식을 제시하고 실험을 통해 그 결과를 비교하고자 한다. 그 방식들은 다음과 같다.

ⅰ) 균등 분배 방식 (기존)
ⅱ) 도메인 크기 비례 분배 방식 (기존)
ⅲ) 속성값 총합 비례 분배 방식: 각 속성별로 그 속성에 속하는 모든 값들의 총합에 비례하여 노이즈 파라미터를 분배하는 방식이다.
ⅳ) 속성값 제곱합 비례 분배 방식: 각 속성별로 그 속성에 속하는 모든 값들의 제곱의 합에 비례하여 노이즈 파라미터를 분배하는 방식이다.
ⅴ) 유격 총합 비례 분배 방식: 속성별로 정렬한 후 각 속성에 속하는 각 값들의 유격을 구하여 유격의 합에 비례하여 노이즈 파라미터를 분배하는 방식이다.
ⅵ) 유격 제곱합 비례 분배 방식: 속성별로 정렬한 후 각 속성에 속하는 각 값들의 유격을 구하여 유격의 제곱의 합에 비례하여 노이즈 파라미터를 분배하는 방식이다.

4. 실 험

본 장에서는 실험을 통해 앞서 제시한 6가지 노이즈 파라미터 분배 방식 별 데이터 유틸리티를 비교한다. 실험에 사용한 데이터는 인구 조사 데이터 셋과 캘리포니아 주택 데이터 셋이다. 지금까지 진행된 대부분의 비상호적 차분 프라이버시 모델 연구가 위 두 데이터 셋을 사용했기 때문에 본 연구에서도 동일하게 두 데이터 셋을 사용했다. 실험 환경과 데이터 셋의 상세 정보는 다음과 같다.

표 1. 실험 환경

CPU	Intel(R) Core(TM) i5-8600 CPU @ 3.10 GHz
MEM	32GB
DB	Maria DB
Language	Java
OS	Windows 10

표 2. 실험에 사용한 데이터 셋

	인구 조사 데이터	캘리포니아 주택 데이터
레코드 수	1080	20960
속성 수	4	9

최적 단변량 마이크로집계모델 기반으로 실험을 진행했으며 노이즈별 분배 방식만 다르게 하여 데이터 유틸리티를 구하였다. 노이즈 파라미터 ϵ은 0.1, 1, 5, 10의 총 4가지 값을 사용해 실험하였다. 다음은 실험 결과를 그래프로 나타낸 것이다.

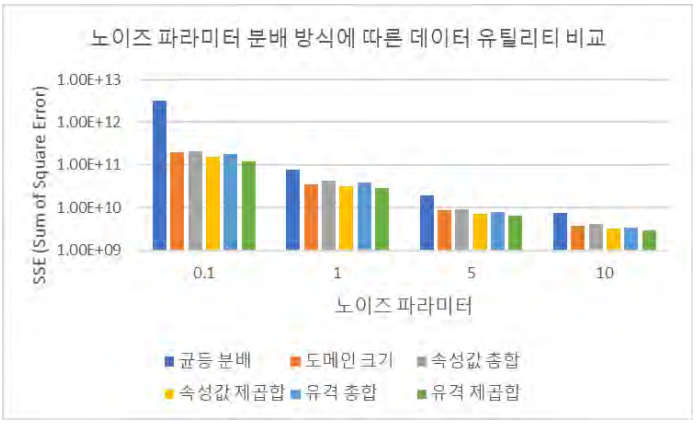


그림 1. 노이즈 파라미터 분배 방식에 따른 데이터 유틸리티 비교 (인구 조사 데이터)

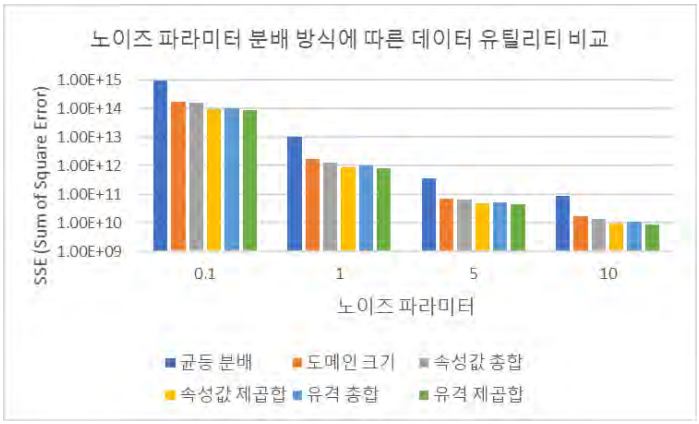


그림 2. 노이즈 파라미터 분배 방식에 따른 데이터 유틸리티 비교 (주택 데이터)

실험에서 지표로 사용한 SSE(Sum of Square Error)는 삽입되는 노이즈들의 제곱 합을 나타내는 것이다. 삽입된 노이즈가 클수록 원본 데이터 셋의 값과 큰 차이가 나는 것이기 때문에 데이터 유틸리티는 감소하게 된다. 따라서 SSE는 데이터 유틸리티에 반비례한다. SSE가 낮을수록 데이터 유틸리티가 높은 노이즈 파라미터 분배 방식으로 볼 수 있다. 기존의 균등 분배 방식과 도메인 크기 비례 분배 방식보다 본 논문에서 제안하는 네 가지 분배 방식에서 데이터 유틸리티가 높게 측정되었다. 그리고 속성값 총합과 유격 총합에 비례하여 노이즈 파라미터를 분배하는 방식보다 총합의 제곱에 비례하여 분배하는 방식에서 데이터 유틸리티가 약 5~10% 가량 더 높게 측정되었다.

5. 결 론

본 논문에서는 다양한 노이즈 파라미터 분배 방식을 사용하여 비상호적 차분 프라이버시 모델의 데이터 유틸리티를 향상시키는 방법에 대해 연구하였다. 속성에 속하는 모든 값들의 제곱의 합 또는 각 속성 값들의 유격을 구하여 유격의 제곱의 합에 비례하여 속성마다 노이즈 파라미터를 분배하는 방식이 기존 분배 방식보다 데이터 유틸리티를 향상시킴을 확인하였다.

6. 참고문헌

[1] C. Dwork, K. Kenthapadi, F. McSherry, I.Mironov, and M. Naor, “Our data, ourselves: privacy via distributed noise generation,” in Proceedings of the

24th Annual International Conference on The Theory and Applications of Cryptographic Techniques, pp.486-503, 2006.
[2] 정강수, 박석, “차분 프라이버시 기반 비식별화 기술에 대한 연구”, 정보보호학회지, 28(2), pp.61-77, 2018.
[3] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, Salil Pravin Vadhan, “On the complexity of differentially private data release: efficient algorithms and hardness results” in STOC '09: Proceedings of the forty-first annual ACM symposium on Theory of computing, pp.381-390, 2009.
[4] Jordi Soria-Comas, Josep Domingo-Ferrer, David Sánchez, Sergio Martínez, “Enhancing data utility in differential privacy via microaggregation-based k-anonymity” in The VLDB Journal volume 23, pp.771-794, 2014.
[5] David Sánchez, Josep Domingo-Ferrer, Sergio Martínez, Jordi Soria-Comas, “Utility-preserving differentially private data releases via individual ranking microaggregation” in Information Fusion Volume 30, pp.1-14, 2016.
[6] Jordi Soria-Comas, Josep Domingo-Ferrer, “Differentially private data publishing via optimal univariate microaggregation and record perturbation” in Knowledge-Based Systems Volume 153, pp.78-90, 2018.

온톨로지 기반 위험 구성 요소 분석을 통한 보안 요구사항 추천 방법 : 심층 보안 방어 관점

정지욱○ 이석원†
아주대학교 인공지능학과○† 소프트웨어학과†
jiwook@ajou.ac.kr, leesw@ajou.ac.kr

The Method of Security Requirements Recommendation for Analysis of Risk Components Based on Ontology : in Security Defense-in-depth Aspect

Ji-Wook Jung○ Seok-Won Lee†
Department of Artificial Intelligence○†, Software and Computer Engineering†, Ajou University

요 약

APT 공격은 장기적이고 특정 목표에 대한 정밀한 공격으로 공격 대상에 특화된 다양한 도구를 사용한다. 따라서 단순한 보안 대책이 아닌 체계적이고 종합적인 분석을 통한 보안 대책이 필요하다. 이를 위해 본 논문은 온톨로지 기반 위험 구성요소 분석을 통해 보안 요구사항과 보안 대책 추천 방법을 제안한다. 제안된 방법은 공격 요소와 공격 요소의 관계를 나타내는 온톨로지 구조, 보안 대책 제안을 위한 규칙 정의, 그리고 취약점이 포함하는 위험 산정 방법을 포함한다. 실제 APT 공격 사례를 기반으로 구성된 온톨로지를 통해 위험 구성요소를 분석하고 보안 요구사항과 보안 대책을 제안하는 과정을 통해 제안 방법의 기능적 테스트를 수행하였으며, 보안 대상의 취약점이 내재하고 있는 잠재 위험을 고려한 추가적인 보안 대책을 식별하였다.

1. 서론

APT(Advanced Persistent Threat) 공격은 지능형 지속 위협으로 단기적이고 정형화된 방식이 아닌 장기적이고 특정 목표에 대한 정밀한 공격을 가하는 사이버 공격이다. 공격자는 APT 공격을 통해 장기간에 걸쳐 공격 목적을 달성하고자 하며, 특히 대기업이나 국가의 주요 시설에 접근하기 위해 공격 대상의 공급망을 구성하는 상대적으로 보안에 취약한 협력 기업을 공략하고 있다[1]. 한편 스피어 피싱 등 사회공학적 기법을 통해 일반적 공격 대상인 기술적 요소 뿐만 아니라 관리적, 인적 요소를 대상으로 공격을 수행한다[2][3]. 이와 같이 단순하지 않은 공격에 대한 보안 대책을 마련하기 위해서는 다양한 관점과 요소를 고려한 분석 방법이 필요하다.

본 논문은 다양한 관점에서 위험 관련 공격 요소를 도출하고 요소 간 관계를 파악하여 APT 공격을 분석할 수 있는 온톨로지 기반 위험 구성 요소 분석을 통한 보안 요구사항과 보안 대책 추천 방법을 제안한다. 또한 추천된 보안 대책의 가중치를 통해 보안 대책 유무에 따른 잠재적 위험 수치를 나타내는 지표를 명세한다. 이를 위해 온톨로지 구조를 설계하고 인스턴스를 추가하여 온톨로지 기반 지식 베이스를 구축한다. 또한 SWRL 기반 추론 규칙을 생성하여 추론규칙에 따라 컨텍스트 정보를 생성한다. 제안된 방법에 대하여 사용자가 보유하고 있는 자산과 기존 보안 대책에 대한 추가적인 보안 요구사항 및 보안 대책을 추천하는 기능적 실험을 수행하고, 이를 통해 내재되어 있는 위험도를 표현한다. 2장은 본 논문에서 제안하는 방법에 대한 관련 연구로 APT 공격 분석과 보안 대책을, 3장은 제안하는 온톨로지 기반 위험 구성 요소 분석을 통한 보안요구사항 추천방법, 4장은 제안 방법에 대한 실험과 결과, 5장에서는 결론과 향후 과제를 서술한다.

이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF2020R1F1A1075605)

2. 관련 연구

APT 공격은 공격 사례와 목표에 따라서 생애 주기 모델이 제안되었다. [4]에서는 공격의 흐름과 목표를 분석하여 나타낼 수 있는 공격 단계를, [2]에서는 공격 사례를 분석하여 각 단계 별 공격 요소를 도출한 모델을 제안하였다.

APT 공격에 대한 보안 대책으로 다양한 기법을 사용한 방법들이 제안되었다. [5]에서는 머신러닝 기반 학습 모델을 이용하여 공격주기를 탐지하는 방법이 제안되었다. 이 방법은 APT 공격에 대한 조기 경보로 신속한 대응은 가능하지만 사전 예방이나 대책에 한계가 있다. 한편 요소들 간에 의미론적 관계를 정의하는 온톨로지를 이용한 방법도 있다. [6]은 온톨로지 기반 운영체제 이벤트와 네트워크 이벤트의 상관관계를 바탕으로 방어 대책을 제안하였고, [2][7]에서는 요구 공학적 측면에서 문제 도메인 온톨로지를 통해 상황에 따라 보안 요구사항을 추천하는 방법을 제안하였다.

3. 제안 방법

본 논문은 온톨로지를 기반으로 APT 공격과 공격 대상에 대한 위험 요소를 분석하여 위험을 평가하고 보안요구사항을 추천하는 방법을 제안한다. 이를 위해 공격 기술을 분석하고, 보안 요구사항과 위험 구성요소의 관계모델[7][9]을 재정의한 온톨로지와 추론규칙을 통해 공격 대상에 대한 위험을 평가하기 위한 지표를 제안한다.

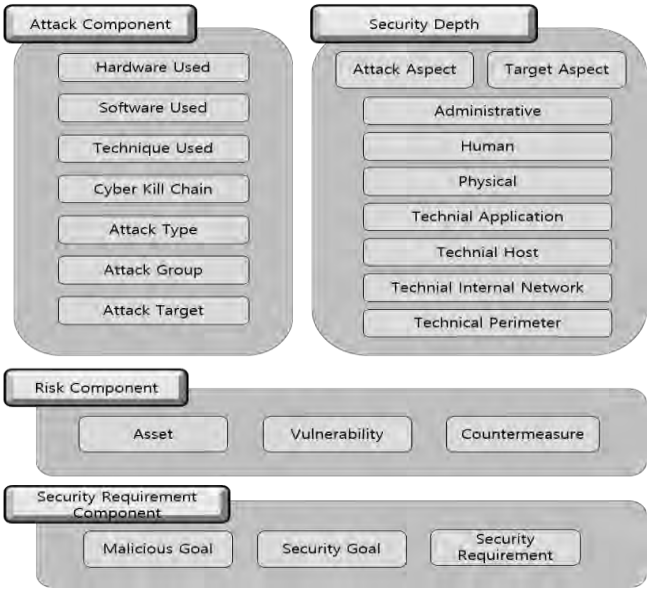


그림 1 온톨로지 구성을 위한 공격 요소 분석

3.1 공격 요소 분석

APT 공격 위험을 분석하기 위한 온톨로지는 그림 1과 같이 4가지 요소로 구성된다. 공격 컴포넌트(Attack Component) 에는 공격 관련 기술, APT 공격 주기, 공격 그룹 등 APT 공격에 관련된 요소를 포함하고 있다. 심층 보안(Security Depth) 컴포넌트는 공격과 공격 대상 관점의 공격 속성을 구분한다. 위험 컴포넌트(Risk Component)[7][9]에는 공격 대상의 자산, 자산에 대한 취약점 그리고 대응방안으로 구성된다. 보안 요구사항 컴포넌트(Security Requirement Component)는 공격 관점에서의 악의적인 목표를 정의하였고, 공격 대상 관점에서의 보안 목표와 보안 요구사항을 포함하고 있다.

3.2 온톨로지 구축

분석된 APT 공격 요소를 기반으로 그림 2와 같이 클래스를 정의하고, 각 클래스 간의 관계를 명세함으로써 온톨로지를 구축한다. 추가적으로 그림 2 하단의 Vulnerability 클래스는 가중치 클래스와 data property 관계로 연결하였다. 가중치 클래스는 실수 형태의 수치를 포함하고 있으며, 이를 추가함으로써 각 취약점에 대한 잠재 위험 수치를 도출할 수 있다.

구축된 온톨로지를 통해 공격과 공격 대상 사이의 관계를 식별할 수 있으며, 과거 사례를 구조에 맞게 정형화하여 축적할 수 있다. 축적된 사례는 지속적인 유지보수 과정을 거치면서 그 수가 증가됨으로써 이를 통해 더욱 정밀하고 다각도의 공격 분석이 가능하다.

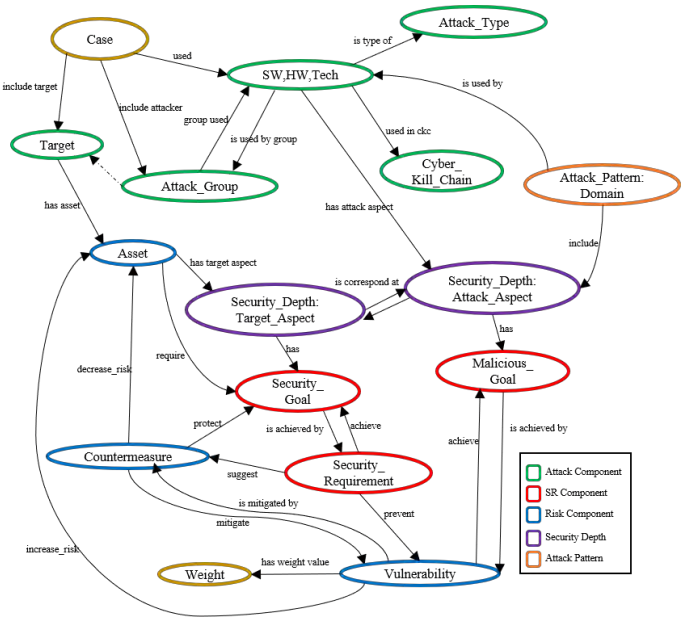


그림 2 온톨로지 구조 표현

3.3 추론 규칙 (Rule)

구축한 온톨로지를 기반으로 공격 대상의 취약점과 위험도, 대응방안을 도출해내기 위한 SWRL 기반 룰을 정의한다. SWRL은 기본적으로 if와 then 간의 관계를 표현하는 것을 전제로 한다. 추론 엔진을 통해 미처 관계가 정의되지 않은 인스턴스의 추가적 관계를 추론할 수 있으며, 기존 OWL 지식 베이스에서 새로운 지식을 추론할 수도 있다[8].

```
CBSAO:decrease_risk(?countermeasure, ?asset) ^ CBSAO:require(?asset, ?security_g...
CBSAO:SD-AT-Technical_Host(?a) ^ CBSAO:SD-TA-Technical_Host(?b) -> CBSAO:is_c...
CBSAO:prevent(?security_requirement, ?vul) ^ CBSAO:is_mitigated_by(?vul, ?counterm...
CBSAO:RC-Asset(?asset) ^ CBSAO:has_target_aspect(?asset, ?ta) ^ CBSAO:has_secu...
CBSAO:prevent(?security_requirement, ?vulnerability) ^ CBSAO:is_mitigated_by(?vulner...
CBSAO:suggest(?security_requirement, ?countermeasure) ^ CBSAO:mitigate(?counter...
CBSAO:SD-AT-Human(?a) ^ CBSAO:SD-TA-Human(?b) -> CBSAO:is_correspond_at_ta(...
CBSAO:SD-AT-Physical(?a) ^ CBSAO:SD-TA-Physical(?b) -> CBSAO:is_correspond_at_t...
CBSAO:SD-AT-Technical_Application(?a) ^ CBSAO:SD-TA-Technical_Application(?b) -> ...
CBSAO:is_mitigated_by(?vulnerability, ?countermeasure) ^ CBSAO:decrease_risk(?cou...
CBSAO:is_correspond_at_aa(?ta, ?aa) ^ CBSAO:is_achieved_by_vulnerability(?mg, ?vul...
CBSAO:RC-Vulnerability(?vul) ^ CBSAO:has_mal_goal(?aa, ?mal_goal) ^ CBSAO:has_a...
CBSAO:SD-AT-Administrative(?a) ^ CBSAO:SD-TA-Administrative(?b) -> CBSAO:is_corre...
CBSAO:SD-AT-Technical_Data(?a) ^ CBSAO:SD-TA-Technical_Data(?b) -> CBSAO:is_c...
CBSAO:SD-AT-Technical_Perimeter(?a) ^ CBSAO:SD-TA-Technical_Perimeter(?b) -> C...
CBSAO:has_target_aspect(?asset, ?target_aspect) ^ CBSAO:has_security_goal(?target...
CBSAO:suggest(?security_requirement, ?countermeasure) ^ CBSAO:protect(?counterm...
CBSAO:SD-AT-Technical_Internal_Network(?a) ^ CBSAO:SD-TA-Technical_Internal_Net...
```

그림 3 SWRL 추론 규칙의 예

추론규칙은 온톨로지 내부 명시적인 관계부터 정의하였으며, 관계 확장에 따라 깊이 있는 추론이 가능하다. 규칙 모음 일부는 그림 3과 같으며 자산의 공격 대상 관점과 보안 목표에 해당하는 보안 요구사항을 추천하기 위한 규칙, 공격 대상 관점에 대응하는 공격자 관점과 악의적 목표를 추론하는 규칙 등을 포함하고 있다. 이는 추후 규칙에 따라 지속적인 확장이 가능하다.

3.4 잠재 위험 산정 방법

공격 대상에 노출될 수 있는 취약점과 보안 대책 부재에 따른 잠재적인 위험 수치를 나타내는 지표를 제안한다. 위험 수치를 나타내기 위한 주요 클래스는 'Countermeasure'와 'Vulnerability'이며, 본 논문에서는 공격 대상이 사전 준비한 대응방안과 준비하지 못해 공격에 노출되어 있는 취약점을 트레이드 오프 (trade-off) 관계로 표현하였다. 도출된 취약점에 대해 제안된 대응방안을 수행한다면 잠재적인 위험도는 낮아지며, 수행하지 않는다면 노출되어 있는 취약점에 비례하여 위험도가 증가한다.

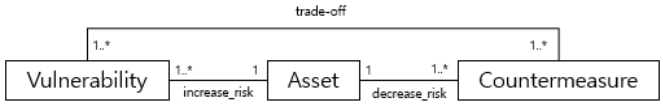


그림 4 자산(A)에 대한 취약점과 대응방안 관계

위험도를 평가하기 위한 지표로 지식 베이스에 구축된 취약점(V)과 대응방안(C)을 나타낸다. 그림 4와 같이 하나의 자산(A)은 여러 개의 취약점이 있을 수 있으므로 여러 개의 대응방안이 요구될 수 있다. 각 취약점에 따라 가중치(W)를 할당하였으며, 취약점과 가중치는 일대일 관계를 가진다. 이를 통해 취약점에 대한 잠재적인 위험을 수치로 표현할 수 있다.

표 1 위험도 지표 기호

기호	의미
A	Asset
V	Vulnerability $[v_1, \dots, v_i]$
C	Countermeasure(Ontology) $[c_1, \dots, c_j]$
C'	Countermeasure(New Case) $[c'_1, \dots, c'_k]$
W	Weight of Vulnerability $[w_1, \dots, w_i]$

각 C는 False 혹은 True의 논리값을 가진다. 대응방안이 사전에 준비되어 있다면 False, 그렇지 않으면 True를 가진다. 가중치는 취약점에 따른 가중치를 가지고 있으며 위험도를 표현해준다. 각 취약점은 하나의 가중치에 대응된다. 최초 가중치는 0에서 1사이의 실수 값을 가진다.

$$W = Softmax(W) \tag{1}$$

취약점 개수에 따라 벡터 값의 총합을 1로 맞추기 위해 (1)을 통해 정규화를 수행한다. 정규화를 거친 가중치와 C와 C'의 비교를 통해 해당 A가 취약점에 노출되었는지, 어느 정도의 위험을 가지고 있는지 나타낼 수 있다. C와 C'의 비교에 따른 결과는 (2)에서 확인할 수 있다. C와 C'의 요소가 다른 경우는 기존 지식베이스에 있는 대응방안보다 적게 구축되어 위험이 존재한다. 반대로 지식베이스의 사례보다 많거나 동일하게 구축되어 있다면, 해당 자산에 대한 위험이 존재하지 않는다.

$$A_{Risk} = \begin{cases} W & s.t.\ C \neq C' \\ 0 & s.t.\ C \leq C' \end{cases} \tag{2}$$

또한 하나의 자산이 아닌 사례에 대한 전체 위험도도 개별적인 요소부터 전반적인 사례에 대한 위험도로 계산할 수 있다.

4. 실험 및 결과

4.1 실험

제안한 방법은 가설을 설정하고 시뮬레이션을 통해 기능 시험을 수행하였다. 시험을 위한 인스턴스를 온톨로지 내부에 추가하였고, 주요 공격 요소들은 표 2와 같다. 사용자 A(공격 대상)가 자산 목록과 기존 대응방안을 입력 한다.

표 2 온톨로지 내부 가상 인스턴스

Asset	Target aspect	Security Goal	Security Requirement	Vulnerability	Weight	Softmax(W)	Countermeasure
asset1	administrative	sg1	sr1	vul1	0.4	1	countermeasure1 countermeasure2 countermeasure3
asset2	techinal_internal_network	sg2	sr2	vul2	0.52	0.3	countermeasure4
			sr3	vul3 vul4	0.56 0.74	0.31 0.37	countermeasure5 countermeasure6

본 논문에서는 정의한 세 가지 가정은 다음과 같다. 첫째, 가중치는 대응방안 개수에 맞추어 소프트맥스를 통해 정규화가 되어있다. 둘째, 인스턴스 명은 정의한 관계와 추론규칙 결과를 고찰하기 위해 추가하였으며, 각 클래스의 인스턴스는 임의 관계가 있다. 셋째, 사용자의 비 구조화된 형태의 쿼리에서 SQWRL 쿼리 형태의 변환 과정을 수행했다.

표 3 사용자 사례 가상 인스턴스

Asset	Countermeasure
asset1	countermeasure1
asset2	countermeasure4
	countermeasure6

표 3은 사용자가 입력한 정보로 사용자가 담당하는 기업 자산 목록과 적용된 보안대책이다. 정의된 가정하에 본 제안 방법으로 취약점 진단을 수행하고, 취약점에 따른 보안 요구사항과 보안대책을 추천한다. 그리고 잠재위험 지표를 통해 각 취약점의 위험도를 측정한다.

표 2와 표 3에서 사용자가 입력한 자산1에 대한 보안 대책은 대응방안1만 존재하므로 자산1에 대한 추가 방안이 요구된다. 자산2에도 1가지 보안 대책이 요구된다. 이에 따라 추가적인 보안 요구사항과 해당하는 대응방안이 제시되어야 한다.

```
CBSAO:RC-Asset(CBSAO:asset1) ^ CBSAO:decrease_risk(?countermeasure, CBSAO:asset1) ^
CBSAO:increase_risk(?vulnerability, CBSAO:asset1) ^ CBSAO:prevent(?security_requirement, ?vulnerability) ^
CBSAO:RC-Countermeasure(CBSAO:countermeasure1) . sqwrl:makeSet(?s1, ?countermeasure) ^
sqwrl:makeSet(?s2, CBSAO:countermeasure1) . sqwrl:difference(?s3, ?s1, ?s2) ^
sqwrl:element(?proposed_countermeasure, ?s3) ^ CBSAO:has_cou_weight(?proposed_countermeasure, ?wei) ->
sqwrl:select(?proposed_countermeasure, ?vulnerability, ?security_requirement, ?wei)

CBSAO:RC-Asset(CBSAO:asset2) ^ CBSAO:decrease_risk(?countermeasure, CBSAO:asset2) . sqwrl:makeSet(?s1,
?countermeasure) ^ sqwrl:makeSet(?s2, CBSAO:countermeasure4) ^ sqwrl:makeSet(?s3, CBSAO:countermeasure6)
sqwrl:union(?u, ?s2, ?s3) ^ sqwrl:difference(?d, ?s1, ?u) ^ sqwrl:element(?proposed_countermeasure, ?d) ^
CBSAO:decrease_risk(?proposed_countermeasure, CBSAO:asset2) ^ CBSAO:increase_risk(?vulnerability,
CBSAO:asset2) ^ CBSAO:prevent(?security_requirement, ?vulnerability) ^ CBSAO:suggest(?security_requirement,
?proposed_countermeasure) ^ CBSAO:has_cou_weight(?proposed_countermeasure, ?wei) ->
sqwrl:select(?proposed_countermeasure, ?vulnerability, ?security_requirement, ?wei)
```

그림 5 자산1, 2의 보안 대책 제안을 위한 쿼리 생성

사용자 자산의 보안 요구사항, 대응방안을 제안하기 위해 그림 5와 같이 SQWRL을 이용하여 사용자 요구를 온톨로지를 통해 결과를 얻기 위한 쿼리를 생성해낸다.

proposed_countermeasure	vulnerability	security_requirement	
CBSAO:countermeasure2	CBSAO:vulnerability1	CBSAO:security_requirement1	1
CBSAO:countermeasure3	CBSAO:vulnerability1	CBSAO:security_requirement1	1

proposed_countermeasure	vulnerability	security_requirement	
CBSAO:countermeasure5	CBSAO:vulnerability3	CBSAO:security_requirement3	0.31
CBSAO:countermeasure5	CBSAO:vulnerability4	CBSAO:security_requirement3	0.37

그림 6 쿼리 수행 결과

4.2 결과

그림 6은 SQWRL Tab에 출력된 실행 결과로 자산1과 2에 대한 결과를 보여준다. 자산1에 대한 결과로 사전에 대비된 대응방안1을 제외한 보안 대책과 그에 따른 취약 인스턴스, 보안 요구사항을 추천하고 가중치를 나타내어 보안 효과에 대한 지표를 표현한다. 자산2도 실험 예상 결과와 동일하게 수행된다. 대응방안에 대한 각 가중치는 정규화 이후의 수치로 나타났으며, 자산1에 대해서는 취약점1을 해결하기 위해 대응방안2와 3이 추가적으로 필요하기 때문에 결과 수치 1을 통해 노출되어 있음을 알 수 있으며, 자산2는 대응방안5의 부재로 보안 요구사항2에 충족되지 않아 취약점 3과 4이 나타났으며, 이들의 최초 위험에 대한 수치를 정규화를 통해 상대적인 수치로 보여준다. 위의 나타난 위험 수치는 대응방안이 모두 만족되면 0으로 나타날 것이며, 노출되어 있는 취약점에 대해 조회되는 결과는 없을 것이다.

5. 결론

본 논문은 APT 공격에 따른 공격 대상의 자산과 기존 대책 외에 추가적인 보안 대책을 제안하는 방법을 제안한다. 사용자의 쿼리로부터 자산과 기존 보안 대책을 식별하여 SQWRL 엔진을 통해 미처 마련하지 못한 보안 대책, 노출되어 있는 취약점, 그리고 잠재 위험 지표를 나타냈다. 온톨로지는 위험 분석 요소, 공격 기술과 공격 대상의 관점, 보안 요구사항 요소와 요소들 사이의 관계를 포함하고 있다. 기능 시험을 위해, 온톨로지 내부에 가상의 인스턴스와 사용자 입력에 대한 가상 인스턴스를 통해 결과를 출력하였다. 온톨로지는 지속적인 사례 축적을 통해 보다 적합한 보안 대책을 마련하는데 기여할 수 있다.

제안된 방법의 효과성과 신뢰성 향상을 위해 보다 다양한 사례들을 온톨로지 내부에 인스턴스화하여 온톨로지를 확장할 필요가 있으며, 기능 시험뿐만 아니라, 비기능적 시험을 통한 검증도 요구된다. 또한 각 자산이 가지고 있는 취약점에 대한 위험도 가중치를 해당 도메인 전문가를 통해 검증하고, 그 외 고려사항에 따라 지속적인 개선도 필요하다.

향후 보안대책 추천뿐만 아니라 공격자의 공격 패턴 분석을 통해 발생 가능한 공격을 예측하며, 지속적으로 노출되는 취약점을 분석하고 새로운 사례 축적을 통해 지식베이스를 개선해 나갈 수 있는 사례 기반 추론(CBR) 시스템으로의 확장 연구를 진행하고자 한다.

참고문헌

[1] kaspersky, "What Is an Advanced Persistent Threat (APT)?", http://www.kaspersky.com. 2019.

[2] Kim, MinJu, Sangeeta Dey, and Seok-Won Lee. "Ontology-Driven Security Requirements Recommendation for APT Attack." 2019 IEEE 27th International Requirements Engi neering Conference Workshops (REW). IEEE, 2019.

[3] 한국인터넷진흥원, “2020년 2분기 사이버 위협 동향 보고서”, 한국인터넷진흥원, 2020.

[4] Chen, Ping, Lieven Desmet, and Christophe Huygens. "A study on advanced persistent threats.", IFIP International Conference on Communications and Multimedia Security. Springer, Berlin, Heidelberg, 2014.

[5] LIU Hai-bo,WU Tian-bo,SHEN Jing,SHI Chang-ting. Advanced Persistent Threat Detection Based on Generative Adversarial Networks and Long Short-term Memory[J], Computer Science, 47(1): 281-286, 2020.

[6] Lajevardi, Amir Mohammadzade, and Morteza Amini. "A semantic-based correlation approach for detecting hybrid and low-level APTs.", Future Generation Computer Systems, 96, 64-88, 2019.

[7] Kim, Bong-Jae, and Seok-Won Lee. "Understanding and recommending security requirements from problem domain ontology: A cognitive three-layered approach.", Journal of Systems and Software, 169, 2020.

[8] Martin O'Connor, Holger Knublauch : Writing Rules for Semantic Web Using SWRL and Jess, Standford University School of Medicine 2004.

[9] 박신혜,이석원, "사례 기반 보안 공격 온톨로지를 사용한 공격 중심 위험 분석 방법 제안.", 한국정보과학회 학술발표논문집 807-809, 2020

임베디드 소프트웨어 도메인에서의 프로그램 합성 기법의 성능 확인을 위한 사례연구

김요엘○, 최윤자*
경북대학교 컴퓨터학부
kimyoel2305@gmail.com, yuchoi76@knu.ac.kr

A Case Study on the Performance of Program Synthesis in Embedded Software Domain

Yoel Kim○, Yunja Choi*
School of Computer Science and Engineering, Kyungpook National University

요 약

프로그램 합성 기법은 다양한 형태의 제약조건을 받아들여 프로그램 탐색 공간에서 제약조건을 만족하는 실행 가능한 프로그램을 찾아내는 기법이다. 임베디드 소프트웨어의 검증 시간과 복잡도를 단축하기 위해 높은 정확도를 가지고 더 단순한 형태로 합성할 수 있는 합성 기법을 찾아볼 가치가 높다고 판단하여 본 연구에서는 Duet과 DryadSynth 두 합성기를 통해 Object Follower의 함수들을 합성하고 합성 결과를 확인했다. Duet은 입/출력 예제 생성이 쉽고 자동화가 가능하며 프로그램 탐색 공간이 좁은 경우 성능이 매우 뛰어났지만 반대의 경우 합성 결과가 좋지 못했고, DryadSynth는 입/출력 사이의 관계를 완벽하게 표현하는 제약조건을 작성했을 때 성능이 매우 뛰어났지만 사용자가 직접 작성해야 하며 그 과정이 어렵고 시간이 오래 걸린다는 단점이 있음을 확인할 수 있었다. 이러한 장단점을 토대로 함수 유형별로 어떤 합성 기법을 선택해야 하는지에 대한 기준도 제시했다.

1. 서 론

프로그램 합성 기법은 다양한 형태의 제약조건(입/출력 예제, 입/출력 사이의 논리적인 관계, 자연어로 기술된 명세서, 프로그램의 일부분, 비효율적인 프로그램 등)을 받아들여 프로그램 탐색 공간에서 제약조건을 만족하는 실행 가능한 프로그램을 찾아내는 기법이다 [1].

[2]에서는 모바일 앱을 테스트하기 위해 안드로이드 프레임워크나 서드파티 라이브러리의 모의 객체(test mock)를 프로그램 합성 기법을 통해 생성하려는 시도가 있었다. 이와 비슷하게 프로그램 합성 기법을 통해 기존의 함수를 더 단순한 형태로 합성할 수 있을 경우 합성 결과를 소프트웨어 동적 검증에 사용하여 검증 시간과 복잡도를 단축시킬 수 있다. 특히 이러한 시도를 임베디드 소프트웨어 도메인에서 적용한다면 특정 하드웨어를 호출하는 함수를 플랫폼 독립적인 함수로 변환하여 검증에 사용될 수 있다. 따라서 높은 정확도를 가지면서 더 단순한 형태로 합성할 수 있는 프로그램 합성 기법을 연구해볼 가치가 높다고 판단한다.

플랫폼 독립적인 함수를 검증하기 앞서 본 연구에서는 임베디드 소프트웨어 도메인에서 자주 사용되는 함수별 합성 기법의 성능을 확인하기 위해 입/출력 예제를 제약조건으로 받아들이는 Duet [3]과, 입/출력 사이의 논리적인 관계에 대한 표현식을 제약조건으로 받아들이는 DryadSynth [4]를 소개하고, 간단한 예제를 통해 두 합성기를 비교한다. Object Follower [5]의 함수들을 대상으로 두 합성 기법을 적용하여 얼마나 잘 합성하는지를 비교하고 분석한다. 마지막으로 두 합성 기법의 장단점을 파악하고 함수 유형별로 어떤 합성 기법을 선택해야 하는지에 대한 기준을 제시한다.

2. 프로그램 합성 기법 소개

본 연구에서는 프로그램 합성 기법 중 SyGuS(Syntax-Guided Synthesis) [6] 방식을 채택하는 합성 기법을 사용한다. SyGuS는 사용자가 합성하고 싶은 프로그램의 스펙(Specification)을 작성할 때 기존의 제약조건에다 사용 가능한 자료형과 연산자, 상수 값들을 추가로 제한하여 프로그램 탐색 공간을 최적화하고 이를 표준화해서 작성하는 방법이다. 합성 결과 역시 동일한 방식으로 표현되기 때문에 여러 가지 합성 기법을 사용해도 동일한 방법으로 실행할 수 있고 서로의 결과를 비교하기 쉽다.

Duet은 가장 작은 표현식부터 점차 큰 표현식을 나열하는 방법을 사용하여 작은 프로그램 부품(컴포넌트)을 만들어내고, 큰 문제를 작은 문제들로 나눠 컴포넌트들이 작은 문제를 해결하고 서로 결합되어 최종적으로 입/출력 예제 제약조건을 만족하는 프로그램을 찾아내는 합성 기법이다. SyGuS-Comp 2019 [7]에서 우승을 차지했던 CVC4보다 SyGuS 벤치마크 문제들을 더 빠르고 더 많이 풀어내어 다른 유사 합성 기법들보다 우위에 있음이 입증되었다.

DryadSynth는 제약조건 표현식들을 연역적으로 풀어내어 합성하는 기법을 사용하고, 연역적으로 풀지 못할 때 해당 문제를 작은 문제들로 나눌 수 있으면 다시 연역적 기법을, 그렇지 않으면 표현식을 나열하는 기법을 사용하면서 서로 협력하여 제약조건을 만족하는 프로그램을 찾아내는 합성 기법이다. SyGuS-Comp 2019의 CLIA(Conditional Linear Integer Arithmetic) 트랙에서 1위를 차지하였다. CLIA는 문법엔 제약이 없지만 산술 연산과 조건 연산, 해당 연산으로 구성된 함수만 사용 가능하다는 의미이다.

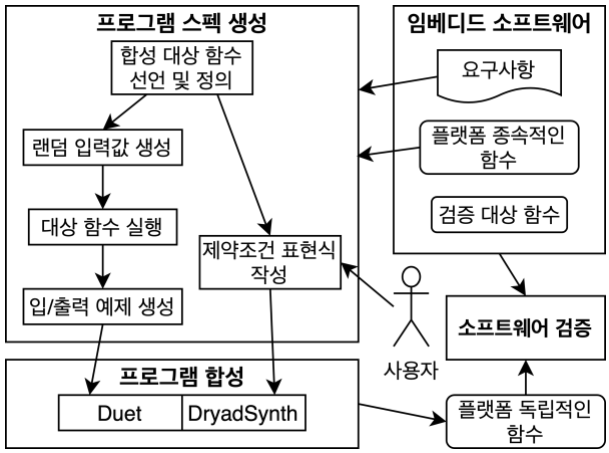


그림 1 프로그램 합성 기법의 활용 방안

3. 프로그램 합성 기법 비교

3.1 합성 과정 및 활용 방안

그림 1은 Duet과 DryadSynth의 구체적인 합성 과정을 비교해서 보여주고, 임베디드 소프트웨어 도메인에서의 프로그램 합성 기법 활용 방안을 보여준다. 합성 대상 함수와 요구사항이 입력으로 주어지면 프로그램 스펙 생성 단계에서 먼저 대상 함수의 정의 부분을 이용하여 함수의 이름, 매개변수, 반환형을 스펙에 작성한다. Duet의 경우 추가로 대상 함수의 소스 코드를 읽어서 사용할 연산자 및 상수 값을 스펙에 작성해야 한다. 반면 DryadSynth의 경우 CLIA 기반이므로 그 부분은 작성할 필요가 없다.

그다음, (1) 요구사항에 부합하는 입력값 범위 안의 랜덤 값을 생성하고, 생성된 입력값으로 대상 함수를 실행한 뒤에 제약조건에 해당하는 입/출력 예제를 생성하여 스펙에 추가하여 Duet을 통해 합성한다. (2) 사용자가 대상 함수의 소스 코드와 요구사항을 읽고, 이를 기반으로 어떠한 함수인지를 파악한 뒤 제약조건에 해당하는 표현식을 작성하고 스펙에 추가하여 DryadSynth를 통해 합성한다.

임베디드 소프트웨어 도메인에서 소프트웨어 검증 대상에 포함되기 힘들었던 특정 플랫폼에서만 호출 가능한 플랫폼 종속적인 함수를, 앞서 설명했던 프로그램 합성 기법을 통해 플랫폼 독립적인 함수로 변환함으로써 검증에 포함시킬 수 있다. 단, 검증에서 사용될 정도라면 합성 결과의 정확도(합성 결과의 출력값이 기존 함수의 출력값과 최대한 비슷한 값을 반환해야 함)가 매우 높으면서도 기존 함수보다 덜 복잡해야 하기 때문에 합성 기법의 성능이 매우 중요하다.

3.2 Duet 스펙 예제

Duet은 대상 함수의 실제 입/출력 예제를 기반으로 합성하기 때문에 대상 함수에 랜덤 입력값을 주고 실행하여 쉽게 입/출력 예제를 생성하여 스펙 생성을 자동화할 수 있다. 그렇게 생성된 스펙의 합성 결과는 표 1을 보면 알 수 있다. 맨 왼쪽 열부터 함수의 이름, 입/출력 예제의 수, 합성 시간, 합성 결과의 사이즈, 랜덤 입/출력 예제 1000개를 기준으로 한 정확도, 합성 시에 사용 가능한 연산자와 상수 값의 수, 그리고 출력값의 범위를 나타낸다. ‘int’는 정수형의 모든 값을, ‘0~’은 0을 포함한 자연수이다.

모든 함수는 매개변수와 반환형이 정수형이고 요구사항에 부합하는 입력값 범위 내에서 랜덤 입/출력값을 100개씩 추가하여 합성했으며, 정확도가 더 이상 증가하지 않을 때까지 합성했다. 대체적으로 대상 함수와 거의 정확하게 합성하면서도 합성 결과의 복잡도(합성 사이즈)가 낮았지만, 운행 거리와 운행 시간, 탑승 시간이 주어졌을 때 택시 요금을 반환하는 taxi 함수는 다른 함수에 비해 정확도가 매우 낮았고 합성 사이즈도 거대했다.

표 1 Duet의 합성 결과

함수 명	입/출력 예제 수	합성 시간	합성 사이즈	정확도	연산/상수 합계	출력 범위
stock	100개	0.3초	24	100%	10개	0~4
area	100개	0.1초	40	100%	5개	0~
median	600개	4.5초	461	100%	4개	int
bool	500개	9.1초	282	99.5%	7개	0~1
max	500개	6.1초	1821	52.2%	10개	0~7
taxi	200개	284.9초	3262	6.5%	15개	3300~

Duet이 taxi 함수를 제대로 합성하지 못했던 이유는 사용 가능한 연산자의 수와 상수 값이 15개로 표 1중에서 제일 많아서 원하는 합성 결과와 전혀 상관없는 작은 컴포넌트들을 많이 만들어내어 컴포넌트 사이즈가 더 이상 증가(복잡하고 큰 표현식을 생성)하지 않아도 쓸모없는 컴포넌트들이 서로 결합하여 합성 자체는 되었지만, 그 결과가 스펙에 작성된 입/출력 예제는 만족하나 사용자의 의도는 만족하지 못하는 과적합한 합성 결과를 도출하기 때문이었다.

3.3 DryadSynth 스펙 예제

DryadSynth는 대상 함수의 입/출력 사이의 논리적인 관계를 표현식으로 작성하여 프로그램을 합성한다. 실제 입/출력 예제를 받아들이지 않기 때문에 사용자가 대상 함수의 소스 코드나 대상 함수와 관련된 정보(요구사항, 입/출력 예제, 주석으로 된 설명 등)를 가지고 함수의 특징을 파악한 뒤에 직접 제약조건에 해당하는 표현식을 작성해야 한다. 표현식을 작성하는 과정에서 입/출력 사이의 관계를 완벽하게 표현할 수 있다면 대상 함수와 서로 같은 함수를 합성할 수 있다.

그림 2는 taxi 함수에 대한 DryadSynth 스펙의 제약조건 부분을 보여준다. 그림 2처럼 taxi 함수의 소스 코드를 보고 같은 연산을 수행하도록 스펙을 작성해서 Duet이 제대로 합성하지 못했던 taxi 함수를 0.2초 만에 합성 사이즈가 156, 정확도가 100%인 함수를 합성할 수 있었다. 또한, 대상 함수의 실제 구현과는 전혀 달라도 대상 함수의 요구사항을 만족하는 새로운 스펙을 작성해서 기존 함수와 비슷한 수준의, 혹은 더 낮은 복잡도를 가진 함수를 합성할 수 있다면 더 좋은 합성 결과를 얻을 수 있다. 하지만, DryadSynth는 스펙 생성을 자동화할 수 없고 직접 작성해야 하기 때문에 스펙 작성 시간이 길어질 수 있다.

```
(define-fun macro((meter Int) (second Int)) Int (
    + 3300 (+ (* 100 (/ meter 131)) (* 100 (/ second 31)))
))

(constraint (= (taxi meter second time)
    (ite (or (and (<= 0 time) (<= time 4)) (= time 24))
        (+ (macro meter second) (* 2 (/ (macro meter second) 10)))
        (macro meter second`)
    )
))
```

그림 2 taxi 함수의 제약조건이 담긴 DryadSynth의 스펙

4. 실험

4.1 실험 대상 함수

Object Follower [5]는 근처의 표적 물체를 따라 이동하면서도 항상 표적으로부터 일정 거리를 유지하는 로봇이다. Object Follower는 크게 카메라에서 측정된 센서 값을 해석하는 함수와 표적과의 거리를 계산하는 함수, 그리고 로봇의 움직임을 제어하는 함수로 이루어져 있다. 이 중에서 센서 값을 해석하는 함수를 두 합성 기법이 얼마나 잘 합성하는지를 실험해보고자 한다.

그림 3은 실험 대상 함수를 보여준다. 대상 함수들은 임베디드 소프트웨어의 전형적인 특징인 배열, 전역변수, 정수형의 사용과 넓은 입/출력의 범위 등이 두드러진다. 대상 함수들은 모두 nxtcamdata라는 전역변수를 사용하며, nxtcamdata은 최대 8개의 rectindex를 가지고 각 rectindex마다 nxtcamdata에 두 점의 x, y 좌표(총 4개의 값)를 저장하고 있어 직사각형의 넓이를 구할 수 있다. 각 좌표 값의 범위는 U8(unsigned char)로 정의했으므로 0~255 사이의 값이다.

```
typedef unsigned char U8;
static U8 nxtcamdata[41];
int getWidth(U8 rectindex);
int getHeight(U8 rectindex);
int getArea(U8 rectindex);
int getbiggestrect(U8 pcolorid, int minarea);

#define MYABS(x) (((x) >= 0)? (x):-(x));
int getWidth(U8 rectindex) {
    int xul = (int) nxtcamdata[5 * rectindex + 1 + 1];
    int xlr = (int) nxtcamdata[5 * rectindex + 1 + 3];
    return MYABS(xlr - xul);
}

int getHeight(U8 rectindex) {
    int yul = (int) nxtcamdata[5 * rectindex + 1 + 2];
    int ylr = (int) nxtcamdata[5 * rectindex + 1 + 4];
    return MYABS(ylr - yul);
}

int getArea(U8 rectindex) {
    return getWidth(rectindex) * getHeight(rectindex);
}

int getbiggestrect(U8 pcolorid, int pminarea) {
    int rectindex = -1;
    int maxarea = pminarea;
    for (int i = 0; i < nxtcamdata[0]; i++) {
        int colorid = (int) nxtcamdata[1 + 5 * i + 0];
        if (colorid == pcolorid) {
            int area = getArea(i);
            if (area >= maxarea) {
                maxarea = area;
                rectindex = i;
            }
        }
    }
    return rectindex;
}
```

그림 3 Object Follower의 실험 대상 함수

getWidth와 getHeight 함수는 rectindex에 해당하는 x/y 좌표 값 2개를 읽은 뒤에 두 수의 차의 절댓값을 반환한다. 출력값의 범위는 0~255 사이의 값이다. getArea 함수는 getWidth와 getHeight 함수의 결과를 서로 곱한 값을 반환한다. 출력값의 범위는 0~65025 사이의 값이다. getbiggestrect 함수는 getArea 함수를 통해 각 rectindex에 해당하는 직사각형의 넓이를 계산해 가장 큰 넓이를 가진 rectindex를 반환한다. 만약 rectindex에 있는 값이 모두 pcolorid와 일치하지 않거나, 가장 큰 넓이가 pminarea보다 작거나, nxtcamdata에 데이터가 없다면 -1을 반환한다. 그러므로 해당 함수의 출력값 범위는 -1~7 사이의 값이다.

4.2 실험 방법

- Duet과 DryadSynth는 실수형을 지원하지 않기 때문에 본 실험에서는 합성 기법이 정수, 논리 자료형 및 해당 자료형의 연산자만 사용 가능하도록 감안했다.
- nxtcamdata과 같이 전역변수를 사용할 경우 스펙에서 전역변수를 매개변수로 추가했다.
- Duet과 DryadSynth는 배열을 지원하지 않기 때문에 nxtcamdata과 같이 배열을 사용할 경우 각 요소들을 정수형 일반 변수로 변환하여 스펙에 추가했다.
- Duet을 통해 합성할 때는 랜덤 입/출력 예제를 100개씩 생성해 스펙에 추가하고 합성했을 때 가장 정확도가 높은 합성 결과를 사용했다. 시간 초과는 1시간으로 설정했으며, 실행 옵션은 lbu, max_size, init_comp_size, all을 사용했고, 적절히 값을 조절해서 사용했다.
- DryadSynth를 통해 합성할 때는 대상 함수의 소스 코드를 직접 확인하면서 스펙을 작성하였으며, getbiggestrect는 반복문이 포함되어 있어 소스 코드 그대로 스펙을 작성할 수 없었기 때문에 기존과는 다른 방식으로 구현되는 스펙을 만들어서 합성했다.
- 합성에 사용된 입/출력 예제를 제외한 1000개의 랜덤 입력값을 생성하여 실제 대상 함수의 출력값과 합성 결과의 출력값을 서로 비교하여 합성 결과의 정확도를 측정했다.

4.3 실험 결과

표 2는 실험 대상 함수에 대한 Duet과 DryadSynth의 합성 결과를 보여준다. 맨 왼쪽 열부터 함수의 이름을 나타내고, Duet에서는 입/출력 예제 수, 합성 시간, 합성 결과의 사이즈, 정확도, 컴포넌트 사이즈(표현식의 복잡도), 합성 시 사용 가능한 연산자와 상수 값의 수를 나타내고, DryadSynth에서는 합성 시간, 합성 결과의 사이즈, 정확도를 나타낸다.

먼저 Duet의 getHeight, getWidth, getArea 함수 합성 결과는 정확도가 전부 5% 미만이었다. 이렇게 합성 결과가 좋지 않았던 이유를 생각해 보면 매개변수가 42개라 입력값의 경우의 수가 매우 많았고, 매개변수 중에서 실제로 사용되는 값은 5개밖에 없어 사용되지 않는 나머지 37개의 매개변수가 합성을 방해했다.

표 2 Duet과 DryadSynth의 실험 결과

함수 명	Duet						DryadSynth		
	예제 수	합성 시간	합성 사이즈	정확도	컴포넌트	연산/상수 합계	합성 시간	합성 사이즈	정확도
getWidth	200개	177.7초	1341	2.7%	3	6개	0.3초	687	100%
getHeight	300개	458.1초	2006	2.2%	3	6개	0.3초	687	100%
getArea	300개	325.9초	3129	4.8%	3	6개	0.4초	2292	100%
getbiggestrect	100개	5.2초	416	37.9%	1	15개	3.0초	56411	100%

특히 출력값의 경우의 수가 각각 256, 256, 65026개로 매우 넓었다. 이러한 과적합 문제를 해결하기 위해 init_comp_size 옵션을 사용했지만 컴포넌트 사이즈를 5 이상으로 증가시키면 컴포넌트 생성 시간이 기하급수적으로 증가해 시간 초과였고, 시작 컴포넌트 사이즈를 4로 해도 3과 동일한 결과이면서 합성 시간만 증가했다.

Duet의 getbiggestrect 함수 합성 결과 약 38% 정도의 정확도를 보여주었다. 이전 함수들보다 정확도가 높은 이유는 출력값의 경우의 수가 8가지 밖에 되지 않았기 때문에 매개변수 간의 복잡한 계산 없이 조건 연산만으로도 그럴듯한 결과를 낼 수 있었기 때문이다. 그러나 랜덤 입/출력 예제 수가 증가한다고 해서 정확도의 유의미한 증가는 없었으며, 오히려 합성 시간과 사이즈만 증가할 뿐이었다.

마지막으로 DryadSynth의 합성 결과를 살펴보면 실험 대상 함수와 동일한 함수를 합성할 수 있었고, 수작업이 필요한 스펙 작성 시간을 제외한 합성 시간은 매우 빨랐다. 하지만 합성 사이즈가 기존 함수보다 너무 길어서 검증에 쓰이기엔 어려울 것으로 예상된다.

표 3 Duet과 DryadSynth의 장단점 요약

합성 기법	장단점
Duet	'간단한' 함수의 경우 성능이 뛰어남
	예제로 스펙을 생성하기 때문에 자동화가 쉬움
	'복잡한' 함수의 경우 성능이 낮음
DryadSynth	'복잡한' 함수를 합성할 수 있고, 성능이 뛰어남
	스펙 작성을 직접 해야 하기 때문에 자동화가 힘듦
	스펙 작성이 어렵고 시간이 오래 걸림

표 3은 3, 4장의 실험 결과를 토대로 Duet과 DryadSynth의 장단점을 요약한 것으로, '간단한' 함수의 기준은 프로그램 탐색 공간이 좁은 경우(컴포넌트 수가 적고 필요한 컴포넌트 사이즈가 5 미만으로 작은 경우), 출력값의 범위가 몇 가지 경우로 고정되는 경우를 말한다. '복잡한' 함수의 기준은 상수 값이 아닌 여러 매개변수 간의 복잡한 표현식을 찾아야 하는 경우(필요한 컴포넌트 사이즈 5 이상으로 큰 경우), 출력값의 범위가 넓은 경우(입력 값에 따라 출력값이 변동하는 경우), 사용 가능한 연산자 및 상수 값이 많은 경우(10개 이상), 사용되지 않는 입력값이 합성에 방해를 하는 경우를 말한다. 물론 절대적인 기준이 아니며, 정확도가 함수 유형마다 다른 것처럼 이러한 기준도 상대적으로 적용된다.

실험 결과로 보았을 때 '복잡한' 함수를 합성해야 할 경우 DryadSynth를 사용하여 사용자가 직접 스펙을 작성해서 합성하는 것이 더 나은 성능을 보인다. 다만 DryadSynth는 스펙 작성을 자동화하기 어렵고 시간이 오래 걸린다는 단점이 존재한다. 그에 비해 표 3에서 언급했듯이 스펙 생성 자동화가 가능하다는 점에서 Duet의 합성 가능성과 정확도를 높이는 방법을 조금 더 연구해볼 필요가 있다.

5. 결론 및 향후 연구

본 연구에서는 Duet과 DryadSynth를 사용하여 Object Follower의 함수들을 합성해 보았다. Duet은 스펙 생성이 쉽고 자동화가 가능하며 프로그램 탐색 공간이 좁은 경우 성능이 매우 뛰어났지만, 매개변수 간의 복잡한 표현식을 통해 출력값을 계산해야 하는 경우 합성 결과가 좋지 못했다. DryadSynth는 입/출력 사이의 논리적 관계를 완벽하게 표현했을 경우 성능이 매우 뛰어났지만, 사용자가 스펙을 작성하기 어렵고 스펙 작성 시간이 오래 걸린다는 단점이 있었다. 향후 연구에서는 본 연구에서 제시한 합성 기법이 동적 검증에 사용될 수 있는 수준까지 정확도를 끌어 올릴 수 있는 방법을 연구해볼 예정이며, 플랫폼 종속적인 함수를 플랫폼 독립적인 함수로 합성하여 실제 검증에 활용해볼 예정이다.

참고 문헌

[1]. S. Gulwani, "Dimensions in Program Synthesis," *Principles and Practice of Declarative Programming*, 2010

[2]. M. Fazzini, A. Gorla, A. Orso, "A Framework for Automated Test Mocking of Mobile Apps," *Automated Software Engineering*, 2020

[3]. W. Lee, "Combining the Top-down Propagation and Bottom-up Enumeration for Inductive Program Synthesis," *Principles of Programming Languages,* 2021 (accepted)

[4]. K. Huang, X. Qiu, P. Shen, et al., "Reconciling Enumerative and Deductive Synthesis," *Programming Language Design and Implementation*, 2020

[5]. Object Following Robot, https://devpost.com/software/object-follower

[6]. R. Alur, R. Bodik, G. Juniwal, et al., "Syntax-Guided Synthesis," *Formal Methods in Computer-Aided Design*, 2013

[7]. SyGuS-Competition 2019, https://sygus.org/comp/2019

입력 키워드 추출을 통한 뮤테이션 기반 Fuzzing의 성능 향상

조정인, 홍신

한동대학교 전산전자공학부
{21600689, hongshin}@handong.edu

Improving Mutation-based Fuzzing by Input Keyword Extraction

요 약

본 논문은 퍼징(fuzzing)을 통한 보다 효과적인 테스트 입력 생성을 위하여 퍼징에 사용되는 테스트 입력 값(seed input)에서 특정 분기 달성에 결정적인 역할을 하는 부분문자열을 입력 키워드로 자동 추출하고, 이를 향후 뮤테이션 과정에 사용하여 분기 커버리지를 향상하는 새로운 방법을 제안한다. 입력 키워드를 사용자가 제공하거나 혹은 테스트 대상 프로그램의 정적 정보를 활용하여 추출하는 이전 방식과 달리, 본 논문이 제안하는 방법은 동적 테인트 분석(dynamic taint analysis)을 통해 입력 텍스트로부터 입력 키워드를 자동 추출한다. 본 연구는 제안하는 기법을 Angora 퍼저에 구현하였고, djpeg을 대상으로 한 실험에서 17.6%의 문맥-분기 커버리지 향상을 확인할 수 있었다.

1. 서론

뮤테이션 기반 퍼징 기법(mutation-based fuzzing)은 주어진 프로그램 입력 값에 연속적으로 변형(뮤테이션 연산)을 적용시킴으로써, 다양한 프로그램 경로를 탐색하는 프로그램 입력 값을 생성하는 자동 테스팅 기법이다[1]. 프로그램의 동적 정보를 활용하여 전략적으로 뮤테이션을 적용하는 Greybox 퍼징 방식이 제안된 이후로 뮤테이션 기반 퍼징을 통한 오류 검출과 테스트 커버리지 달성 성능이 크게 향상되었으며[2], 최근에는 유닛 테스팅, 회기 테스팅 등 다양한 소프트웨어 테스팅 작업에 뮤테이션 기반 퍼징 기법이 널리 사용되기 시작하였다.

뮤테이션 통해 테스팅에 유용한 프로그램 입력을 도출하기 위해서는 (1) 다양한 방향으로 프로그램 입력 값을 변형하면서도, (2) 프로그램 실행경로 전환에 결정적인 값 변화를 발생시키는 효과적인 뮤테이션 연산자(mutator)가 사용되어야 한다. 이러한 두 가지 필요를 동시에 만족하기 위해서, 현재 널리 쓰이는 퍼징 도구의 경우(예: AFL, libFuzzer), 무작위적 뮤테이션 연산자와 입력 키워드(input keyword) 기반한 뮤테이션 연산자를 동시에 사용하는 방식으로 동작한다. 입력 키워드는 테스트 대상 프로그램의 특정 실행 조건에 연관성이 높은 프로그램 입력 값 요소를 별도로 입력 받거나, 테스트 대상 프로그램 코드 등의 정보로부터 수집하여 뮤테이션에 활용하는 방식이 제안되어 왔다.

본 논문은 테인트 분석(taint analysis)를 통해 프로그램 입력 값으로부터 입력 키워드를 자동으로 추출한 후, 이를 활용하여 뮤테이션을 수행하는 새로운 퍼징 기법을 제안한다. 제안하는 기법은 퍼징 과정에서 특정 분기 달성에 결정적인 것으로 추정되는 입력 키워드를 자동으로 추출함으로써, 테스트 입력 탐색 과정에서의 경험을 지식 형태로 표현하여 향후 퍼징 과정에 효과적으로 재활용할 수 있는 방법을 제공하며, 또한 입력 키워드를 사용자가 제공하거나 프로그램 코드로부터 추출하기 불가능한 상황에서도, 프로그램 입력으로부터 추출하여 사용하는 새로운 방법을 제공함으로써 기존의 입력 키워드 기반 뮤테이션 방식을 효과적으로 보완할 수 있다.

본 연구에서는 제안한 기법을 C/C++ 프로그램을 대상으로 하는 Angora 퍼징 도구[3]를 기반으로 활용하여 구현하였다. 특히, Angora 퍼저의 동적 테인트 분석 기능을 활용하여 특정 분기문 달성에 결정적인 역할을 하는 것으로 보이는 프로그램 입력 구간을 추론하여 입력 키워드로 추출하였다. 또한, 추출한 입력 키워드를 Angora 퍼저의 세 가지 뮤테이션 연산 방식에 각각 활용함으로써 프로그램 실행경로의 효율적인 탐색을 유도하였다. 제안한 기법을 Angora에 구현한 후 djpeg 프로그램을 대상으로 실험 평가를 수행한 결과, 제안한 기법의 사용이 17.6%의 문맥-분기 커버리지 성능 향상을 보임을 확인할 수 있었다.

본 논문의 나머지 부분에서는, 먼저 테인트 분석에 기반한 퍼징 도구인 Angora를 소개한 후(2.1절), 본 논문이 제시하는 입력 키워드 추출과 이를 이용한 뮤테이션 방법을 살펴본다(2.2절). 그 후 제안한 기법을 평가하기 위한 실험의 구성과 결과를 소개하고(3장), 향후 연구를 논의함으로써 논문을 마무리한다(4장).

본 연구는 소프트웨어중심대학 지원사업(2017-0-00130)과 한국연구재단의 지원(2020R1C1C1013512, 2017M3C4A7068179)을 받아 수행됨.

2. 입력 키워드 추출을 통한 뮤테이션

2.1. 배경: Angora 퍼저

Angora는 C/C++ 프로그램을 대상으로 한 뮤테이션 기반, Greybox 퍼징 기법이다. Angora는 효과적인 분기 커버리지 달성을 위한 여러 가지 기법이 적용되었는데, 그 중 세 가지 주요 기법의 원리는 다음과 같다:

(1) 세분화된 커버리지인 문맥-분기 커버리지 사용: Angora는 각 분기(branch) 커버리지를 이에 도달하는 당시의 함수 호출 스택(call stack) 경우에 따라 분화한 문맥-분기 커버리지를 달성 목표로 사용하여 퍼징을 수행한다. 문맥-분기 커버리지의 사용은, 실제 프로그램에서, 특정 분기의 달성 조건이 해당 분기를 실행하는 경로에 따라 상이한 경우가 많은 특징에 기인한다. Angora는 다양한 문맥-분기를 도달하도록 퍼징을 수행함으로써, 더 많은 실행 경로를 탐색하게 되고, 이를 통해 더 효과적인 오류 검출 및 커버리지 달성이 이루어지도록 유도한다.

(2) 동적 테인트 분석을 통한 목표 커버리지 대상 뮤테이션 수행: Angora는 데이터 흐름 분석의 일종인 동적 테인트 분석을 활용하여, 도달 목표 분기의 판별 조건 검사에 영향을 미치는 프로그램 입력 오프셋 집합을 파악한 후 해당 오프셋을 대상으로만 뮤테이션을 수행함으로써, 커버리지 목적 달성에 효율적인 테스트 입력 생성을 시도한다. Angora는 동적 테인트 분석의 런타임 오버헤드를 고려하여, 이전에 달성하지 못한 분기를 새로 커버하게 되는 실행에 대해서만 동적 테인트 분석을 수행한다.

(3) 다양한 뮤테이션 방법의 혼합적 사용: Angora는 커버리지 대상 뮤테이션 연산자와 무작위 연산자를 함께 사용함으로써 지향적 탐색과 탐사적 탐색을 복합적으로 수행한다. Angora는 내부적으로 다음 세 가지 종류의 뮤테이션 연산자를 사용한다:

- Exploit: 커버리지 지향적 뮤테이션 연산자로, 동적 테인트 분석을 통해 목표 분기에 연관된 것으로 추정되는 오프셋에 임의의 값 혹은 인근 값을 대입하는 변형을 발생시킨다.
- Explore: Exploit과 유사하게 동적 테인트 분석에서 지정한 오프셋을 임의의 값 혹은 인근 값으로 변형하는데, 이 때 분기 조건식과 적합도가 높은 방향으로 변형을 반복한다.
- AFL: 무작위적 뮤테이션 연산자로, 임의의 입력 오프셋을 지정하여 값을 여러 가지 무작위적 변형을 발생시킨다.

2.2. 제안하는 기법

Angora의 뮤테이션 연산자가 분기 달성에 결정적인 입력 키워드를 사용할 경우, 순차적 혹은 임의적 값 변형보다 효과적이며 효율적인 경로 탐색이 가능하다는 점에 착안하여, 입력 키워드 추출과 추출한 키워드를 활용한 뮤테이션 방법을 다음과 같이 제안한다:

(1) 동적 테인트 분석을 통한 입력 키워드 추출: Angora가 동적 테인트 분석을 통해 특정 분기에 연관된 것으로 추정한 입력 오프셋에 위치한 입력 값을 입력 키워드로 추출한다. 이 때 연관된 입력 오프셋이 불연속적으로 지정된 경우, 연속된 부분 구간별로 입력 키워드를 추출하고, 이에 더하여 가장 이른 오프셋으로부터 가장 늦은 오프셋까지의 전체 구간을 또다른 입력 키워드로 추가로 추출한다. 예를 들어, 프로그램 입력이 “ABCDEFGHIJK”이고 특정 분기에 연관된 오프셋이 {0, 1, 2, 4, 5, 7, 8]로 주어질 경우, “ABC”, “EF”, “HI”, 그리고 “ABCDEFGHI”의 4가지의 입력 키워드가 추출된다.

(2) 입력 키워드를 뮤테이션에 적용: 퍼징 과정에서 추출한 입력 키워드를 저장하여 커버리지 지향 및 무작위 뮤테이션에서 다음과 같이 활용한다:

- Exploit & Explore: 동적 테인트 분석을 통해 목표 분기에 연관된 것으로 추정한 오프셋에 무작위 혹은 인근 값으로 변형하는 뮤테이션에 앞서, 오프셋과 길이가 유사한 키워드를 찾아 대입한다. 오프셋이 연속적이지 않더라도 입력 키워드의 문자열을 순서를 맞추어 대입한다.
- AFL: 기존의 무작위적 뮤테이션 연산자에 추가하여 특정 위치에 추출한 입력 키워드를 삽입하거나, 혹은 특정 위치의 값을 추출한 입력 키워드와 치환하는 변경을 수행한다.

제안한 기법은 특정 초기 입력 값 (initial seed) 혹은 퍼징 과정 가운데 발견된 새로운 커버리지를 달성하는 특정 입력에서 존재하는 고유한 정보(특정 분기 달성에 결정적인 입력 조건)가 다른 여러 뮤테이션 대상 입력에 전파될 수 있는 채널을 제공하는데 목적이 있다. 이를 통해, 퍼징 과정의 특정 시점에 획득한 프로그램 입력 값에 대한 지식이 이후 퍼징 과정에 재활용됨으로써 퍼징의 효율성이 향상되는 효과를 의도하였다.

3. 실험

3.1. 실험 셋팅

본 연구에서는 제안한 기법을 Angora를 기반으로 구현한 후, 기존 Angora와의 성능을 비교하는 실험을 수행하였다. 실험을 위해 제안한 기법을 Angora version 1.2.2에 349 LoC의 Rust 코드를 추가함으로써 구현하였다. 실험대상으로는 퍼징 기법 실험 평가(예: [3][4])에 자주 사용되는 오픈소스 JPEG 압축해제 프로그램인 프로그램으로는 djpeg[1]을 사용하였다. djpeg은 229 LoC의

[1] Independent JPEG Groups. https://github.com/libjpeg-turbo/ijg

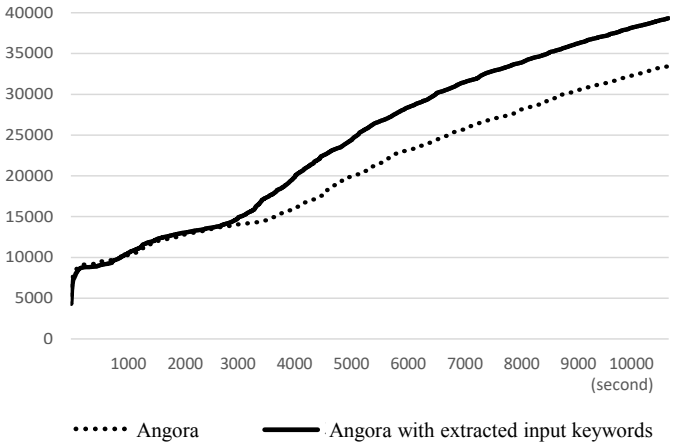


그림 1. 문맥-분기 커버리지 달성

C 코드로 구성되어 있으며, 총 153 개의 분기가 있다. 초기 테스트 입력(initial seed input)으로는 프로젝트 내에 있는 총 4개의 JPEG 이미지 예제 파일 중 2개를 임의로 선택하여 사용하였다.

실험은 기존 Angora와 본 논문이 제안한 기법을 구현한 Angora에 djpeg을 대상으로 각각 180분 동안 퍼징을 수행하는 방식으로 진행하였다. 실험 결과는 최종적으로 달성한 문맥-분기의 수를 측정하였는데, 이는 Angora가 더 높은 문맥-분기 커버리지 달성을 목표로 동작하는 특성을 반영한 것이다. 퍼징의 확률적 동작을 고려하여, 본 실험에서는 180분 간의 퍼징을 총 20회 반복한 후, 최종 커버리지 달성의 평균 값을 비교하였다.

3.2. 실험 결과

총 180분의 퍼징 테스팅 결과, 기존의 Angora는 평균 33440개의 문맥-분기 커버리지를 달성한데 반하여, 본 논문이 제안한 기법은 평균 39341개의 문맥-분기 커버리지를 달성하여 기존 Angora보다 약 17.6% 높은 커버리지 달성을 보였다.

그림 1은 시간에 따라 기존의 Angora (점선)와 본 논문에서 제안한 기법(실선)이 도달한 서로 다른 문맥-분기 커버리지의 총 개수의 증가 추세를 나타낸다. 이 결과를 통하여, 대부분의 시점에서 본 논문이 제안한 기법이 기존의 Angora보다 높은 커버리지 성능을 보임을 확인할 수 있다. 본 논문이 제안한 기법은 총 퍼징 시간동안 평균 1770개의 입력 키워드를 추출하였다. 추출한 입력 키워드의 평균 길이는 124바이트이며, 이 중 16바이트 미만의 입력 키워드의 비중은 평균 23.2%이며, 127바이트 미만의 입력 키워드의 비중은 평균 43.2%였다. 그림 2는 퍼징 과정에서 시간에 따라 추출한 입력 키워드의 평균 개수를 나타낸다. 그림 3은 각 퍼징에서 뮤테이션 대상 입력으로 사용하는 입력 값(favored seed input)의 개수, 즉 퍼징 과정에서 고유한 문맥-분기를 신규로 도달한 입력 개수가 시간에 따라 어떻게 증가하는 지를 나타낸다. 그림 3의 결과로 볼 때,

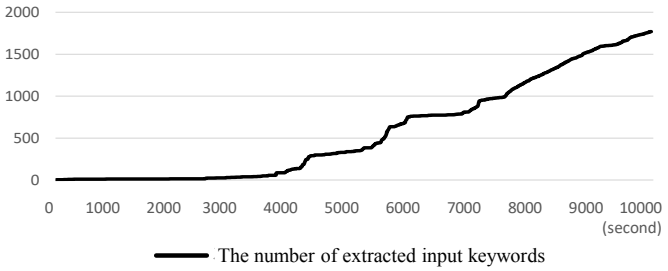


그림 2. 추출한 입력 키워드의 수

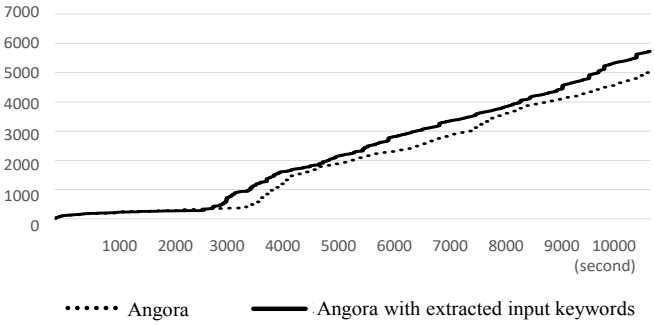


그림 3. 뮤테이션 대상 입력의 수

본 논문이 제안한 기법이 Angora보다 더 많은 문맥-분기에 도달하기 때문에, 퍼징 과정에서 더 많은 수의 뮤테이션 대상 입력을 활용하여 퍼징을 수행하였음을 확인할 수 있다.

4. 결론 및 향후연구

본 논문에서는 효과적인 뮤테이션 기반 퍼징을 위해 동적 테인트 분석을 통해 입력 키워드를 자동으로 추출하여 사용하는 방법을 제안하고, 이를 C/C++ 프로그램 대상 Fuzzing 도구인 Angora를 활용해 구현한 결과를 소개하였다. djpeg을 대상으로 한 실험 결과, 본 논문에서 제안한 방법은 Angora의 문맥-분기 커버리지 달성 성능을 17.4% 향상함을 확인할 수 있었다.

향후 연구에서는 다수의 입력 키워드가 추출되는 상황을 고려하여 입력 키워드를 선택적으로 저장하고 활용하는 방법을 개발하고자 한다. 특별히, 입력 키워드를 추출할 당시 연관된 분기문을 파악함으로써 뮤테이션 향상에 활용할 수 있을 것으로 기대한다. 또한, 동적 테인트 분석 이외의 방식을 통해[5], 보다 효율적으로 입력 키워드를 추출하는 방법에 대해서도 연구하고자 한다.

참조문헌

[1] P. Godefroid, Fuzzing: Hack, Art and Science, Communications of the ACM, 63 (2), Jan. 2020
[2] V. J. M. Manes et al., The Art, Science, and Engineering of Fuzzing: A Survey, IEEE Transactions on Software Engineering (early access)
[3] P. Chen and H. Chen, Angora: Efficient Fuzzing by Principled Search, IEEE Symposium on Security and Privacy (SP), 2018
[4] S. Rawat et al., VUzzer: Application-aware Evolutionary Fuzzing, NDSS Symposium, 2017
[5] S. Gan et al., GREYONE: Data Flow Sensitive Fuzzing, USENIX Security, 2020

패치 우선순위화를 위한 소스 코드 표현방식에 따른 유사도 계산 기법 별 영향도 분석

허진석[O1], 정호현[1], 이은석[2]
성균관대학교 전자전기컴퓨터공학과[1], 성균관대학교 소프트웨어대학[2]
{mrhjs225, jeonghh89, leees}@skku.edu

An Analysis of Similarity Techniques for Patch Prioritization : The Aspect of Source Code Expression Methodologies

Jinseok Heo[O1], Hohyeon Jeong[1], Eunseok Lee[2]
Department of Electrical and Computer Engineering, Sungkyunkwan University[1]
College of Software, Sungkyunkwan University[2]

요 약

자동프로그램수정(Automated Program Repair) 분야에서는 더욱 빠르게 옳은 패치를 선별하기 위해 패치 우선순위화를 활용한다. 이를 위해 기존 기법들은 의심스러운 스테이트먼트와 후보 패치가 적용된 스테이트먼트 간의 유사도 점수를 구하여 우선순위화를 한다. 하지만 유사도 계산에 사용되는 소스 코드들의 표현 방식에 따른 우선순위화 성능에 대한 분석 없이 사용되고 있다. 따라서 본 논문에서는 소스 코드 표현 방식에 따라 유사도 기법들이 패치 우선순위화에 얼마나 영향을 미치는지에 대한 분석을 진행한다. 분석 결과 소스 코드의 표현 방식을 추상 구문 트리(Abstract Syntax Tree), 문자열, 벡터로 나누었을 때 벡터로 표현하여 계산했을 때 유사도 기법의 패치 우선순위화 성능이 제일 높았다. 또한 옳은 패치가 새로운 코드를 삽입하는 형태의 패치일 경우 분석 대상의 모든 유사도 기법들의 우선순위화 성능이 낮았다.

1. 서 론

자동프로그램수정(APR, Automated Program Repair) 분야에서는 더욱 짧은 시간에 옳은 패치를 선별하기 위해 패치 우선순위화를 활용한다 [1]. APR 과정은 크게 버그 추적, 패치 생성, 패치 검증 단계로 나뉘게 된다. 패치 우선순위화는 이 중 패치 검증 단계에 적용되는 기법으로 후보 패치 중 사람에 의해 통과될 수 있는 패치인 옳은 패치를 더욱 높은 순위의 검증 대상으로 높이는 것에 목표가 있다.

기존의 패치 우선순위화 기법들은 의심스러운 스테이트먼트와 후보 패치가 적용된 스테이트먼트의 유사도를 계산하여 우선순위화를 진행한다 [2, 3, 4, 5]. Ripon K. Saha et al. [2]은 버기 코드가 가진 문맥을 반영하기 위해 의심스러운 스테이트먼트가 속한 라인과 후보 패치가 적용된 스테이트먼트의 라인 전, 후의 3줄을 각각 하나의 컨텍스트로 취급한다. 컨텍스트별로 소스 코드를 토큰화한 후 식별자(identifier) 만을 필터링하여 식별자 토큰 집합들을 구성한다. 그리고 식별자 토큰 집합 간의 Jaccard 계수를 구함으로써 유사도를 계산한다. Xuan-Bach D. Le et al. [3]은 소스 코드를 벡터화하여 벡터 간 코사인 유사도를 구한다. 벡터화에는 추상 구문 트리(AST, Abstract Syntax Tree)의 노드 유형을 기준으로 소스 코드를 벡터화 하였다.

소스 코드 간 유사도를 구할 때 사용되는 소스 코드의 표현 방식에는 크게 세 가지가 존재한다.

■ AST

AST는 소스 코드에서 발생하는 구조를 트리로 나타낸 것으로 AST의 특성인 노드의 개수, 노드 중복 여부, 노드의 유형 등을 통해 유사도를 구한다. Ming Wen et al. [4]은 의심스러운 스테이트먼트와 후보 패치가 적용된 스테이트먼트 각각이 속한 메소드들을 AST로 변환하여 노드 유형별 겹치는 수를 계산하여 유사도를 도출한다.

■ 문자열

소스 코드를 단순 문자열로 취급하게 되면 해당 문자열 간 유사도를 계산하여 이를 소스 코드 간 유사도로 활용한다. Zimin Chen and Martin Monperrus [5]는 유사한 소스 코드를 찾기 위해 소스 코드를 문자열 취급하여 공통된 가장 긴 subsequence인 LCS(Longest Common Subsequence)를 구하였다. 그리고 이 길이를 유사도 계산에 사용하였다.

■ 벡터

소스 코드를 벡터화하여 벡터 간의 유사도를 구하게 된다. 벡터 기반 유사도 계산 방식 또한 기존 연구들 [3, 5]에서도 사용되고 있다.

하지만 많은 패치 우선순위화 기법들이 유사도 계산에 사용되는 소스 코드의 표현 방식에 따른 패치 우선순위화 성능의 분석 없이 유사도를 측정하고 있다. 따라서 본 논문에서는 소스 코드의 표현 방식에 따른 유사도 계산 방식들이 패치 우선순위화에 어떠한 영향을 미치는지에 대해 분석한다. 분석을 위해 버그 추적 기법으로는 Ochiai, 패치 생성 및 검증에는

ConFix [6]를 사용하였다. 분석에 사용된 벤치마크는 Defects4j로 이 중 기법 ConFix가 옳은 패치를 생성한 버그들을 대상으로 분석을 진행했다.

본 논문의 기여점으로는

- ✓ 소스 코드의 표현 방식에 따른 유사도 기법 영향력 분석을 하였다.
- ✓ 사람이 작성한 패치 형태에 따른 유사도 기법들의 효과를 확인하였다.

이어지는 2장에서는 분석에 사용되는 유사도 기법에 관해 설명한다. 3장에서는 분석을 위해 설정된 실험 환경과 패치 우선순위화 과정에 대해 자세히 설명한다. 4장에서는 분석 결과에 대해 논의하고 5장에서는 향후 연구와 함께 본 논문을 마무리 짓는다.

2. 유사도 기법

본 장에서는 분석에 사용되는 소스 코드 유사도 계산 기법 6개를 계산의 대상이 되는 소스 코드의 표현 방식에 따라 분류하여 소개한다.

2.1 AST

2.1.1 중복 노드 개수

중복 노드 개수는 Tao Ji et al. [7]에서 사용된 기법으로 의심스러운 스테이트먼트와 후보 패치가 적용된 스테이트먼트를 AST로 나타냈을 때 겹치는 노드의 개수를 측정하여 유사도를 계산한다.

$$Sim_{overlap} = \frac{2 * O}{2 * O + S + C} \quad (1)$$

중복 노드 개수에 따른 유사도 계산은 위의 식과 같다. O는 겹치는 노드의 개수 S, C는 각각 의심스러운 스테이트먼트와 후보 패치가 적용된 스테이트먼트의 AST 노드들 중 겹치지 않은 노드의 개수를 의미한다.

2.1.2 Genealogy Context

Genealogy context는 Ming Wen et al. [4]에서 제안된 유사도 기법으로 후보 패치가 특정 유형의 노드와 같이 사용되는 특징을 고려하기 위해 제안되었다. AST 노드별 유형을 구분하여 유사도를 계산하며, 의심스러운 스테이트먼트와 후보 패치가 적용된 스테이트먼트 각각이 속한 컨텍스트를 대상으로 AST를 만들어 유사도를 계산하며 식은 아래와 같다.

$$Sim_{gen} = \frac{\sum_{t \in T} \min\left(AST_S(t), AST_C(t)\right)}{\sum_{t \in T} AST_C(t)} \quad (2)$$

T는 후보 패치가 적용된 스테이트먼트의 컨텍스트에 등장하는 모든 AST 노드의 유형을 뜻하며, $AST_S(t)$, $AST_C(t)$ 각각 의심스러운 스테이트먼트 와 후보 패치가 적용된 스테이트먼트의 컨텍스트 AST에 등장하는 특정 유형 t에 해당하는 노드의 개수이다.

2.2 문자열

2.2.1 LCS

LCS는 두 문자열 간 공통된 가장 긴 subsequence를 의미하며, 문자를 기반으로 유사도를 측정하기 때문에 가장 기본적인 syntactic 유사도 기법이다 [5]. 이를 기반으로 Zimin Chen and Martin Monperrus [5]은 소스 코드 간 유사도를 측정하였다.

$$Sim_{lcs} = \frac{len\left(LCS(string_S, string_C)\right)}{\max\left(len(string_S), len(string_C)\right)} \quad (3)$$

$string_S$와 $string_C$는 각각 의심스러운 스테이트먼트와 후보 패치가 적용된 스테이트먼트를 문자열로 변환한 것이다. len 은 해당 문자열의 길이를 의미한다.

2.2.2 Contextual Similarity

Contextual similarity는 의심스러운 스테이트먼트와 후보 패치가 적용된 스테이트먼트 간 유사도를 구할 때 근처 코드를 고려하여 유사도를 계산하기 위해 제안된 방식이며 Ripon K. Saha et al. [2] 에서 의심스러운 스테이트먼트와 후보 패치가 적용된 스테이트먼트간 유사도를 구하기 위해 사용되었다.

근처 코드를 고려하기 위해 의심스러운 스테이트먼트와 후보 패치가 적용된 스테이트먼트의 전, 후 3줄을 각각 하나의 컨텍스트로 취급하였고 해당 컨텍스트를 단어 단위로 토큰화하였다. 이후 토큰들 중 식별자만을 선별하여 토큰 집합인 S_S, S_C를 생성한 후 Jaccard 계수를 통해 유사도를 구한다.

$$Sim_{context} = \frac{|S_S \cap S_C|}{S_S \cup S_C} \quad (4)$$

2.3 벡터

2.3.1 코사인 유사도

코사인 유사도는 두 벡터가 가르키는 방향이 얼마나 유사한지를 의미한다. 코사인 유사도 계산 식은 아래와 같다.

$$Sim_{cos} = \frac{\vec{S} \cdot \vec{C}}{|\vec{S}| * |\vec{C}|} \quad (5)$$

S와 C는 각각 의심스러운 스테이트먼트와 후보 패치가 적용된 스테이트먼트의 벡터 형태를 의미한다.

2.3.2 유클리디안 유사도

유클리디안 거리는 n 차원의 공간에서 두 점 간의 거리를 계산하는 방식이다. 본 논문의 분석에서는 의심스러운 스테이트먼트와 후보 패치가 적용된 스테이트먼트가 n 차원 공간에서 얼마나 가까운지를 측정하는 방식으로 취급하였다. 의심스러운 스테이트먼트와 후보 패치가 적용된 스테이트먼트에 해당하는 벡터를 $S = (S_1, S_2, S_3 \dots\dots, S_n)$ 와 $C = (C_1, C_2, C_3 \dots\dots, C_n)$로 취급할 때 계산 식은 아래와 같다.

$$Ed = \sqrt{\sum_{i=1}^{n} (S_i - C_i)^2} \quad (6)$$

$$Sim_{ed} = \frac{1}{1 + Ed} \quad (7)$$

Ed는 유클리디안 거리를 뜻하며 0과 1 사이의 값으로 정규화하기 위해 두 번째 식을 통해 최종 유사도 점수를 계산한다.

표 1 데이터 셋

프로젝트	버그 ID	버그 수
Chart	1, 10, 11, 24	4
Closure	14, 38, 73, 92, 109	5
Lang	6, 24, 26, 51, 57	5
Math	5, 30, 33, 34, 70, 75	6
Time	19	1
합계	-	21

3. 분석 방법

3장에서는 분석에 사용된 도구들과 패치 우선순위화 과정, 분석에 사용된 데이터 셋과 실험 세팅에 대해 자세히 설명한다.

3.1 구현

APR 과정 구현을 위해 버그 추적 과정은 Ochiai 기법을 사용하였고 패치 생성 및 검증 과정은 도구 ConFix를 사용하여 구현하였다. AST 탐사 및 JDT에 내장되어있는 파서(parser)를 사용하였고 소스 코드 벡터화를 위해 도구 Deckard [8]의 알고리즘을 구현하여 벡터화를 진행하였다.

3.2 패치 우선순위화 과정

패치 우선순위화는 패치 생성 과정과 패치 검증 과정 사이에 구현하였다. 패치 생성을 통해 생성된 후보 패치를 적용한 스테이트먼트들을 대상으로 의심스러운 스테이트먼트와의 유사도 점수를 계산한다. 사용되는 유사도 점수가 여러 개 일 경우 각각의 점수에 weight를 주어 최종 유사도 점수 합을 계산한다. 계산된 유사도 점수가 높을수록 의심스러운 스테이트먼트에 적용하기 적합한 후보 패치로 취급하기 위해 오름차순으로 정렬하여 우선순위화 한다.

3.3 데이터 셋

평가를 위해 사용된 데이터 셋은 표 1 과 같다. Defects4j 벤치마크내에 ConFix가 옳은 패치를 생성할 수 있었던 버그 22개 중 deprecated된 Closure-93 버그를 제외한 21개의 버그를 대상으로 패치 생성 및 우선순위화를 진행하였다.

3.4 실험 환경

패치 우선순위화에 대한 분석을 진행하기 위해 기존 패치 검증 과정 및 종료 과정을 일부 변경하였다. 후보 패치를 제한 시간 없이 제한 수(20,000)까지 혹은 생성할 수 있는 패치 재료가 떨어질 때까지 생성한 후 우선순위화를 거쳐 우선순위화한 순서대로 패치 검증을 진행한다. 패치 검증 과정 또한 시간 제한 없이 생성한 모든 후보 패치에 대해 검증을 거친다.

4. 분석 결과

표2는 전체적인 분석 결과를 보여주고 있다. 총 개수는 해당 버그에 대하여 생성한 후보 패치의 총 개수이며 Original은 패치 우선순위화를 사용하지

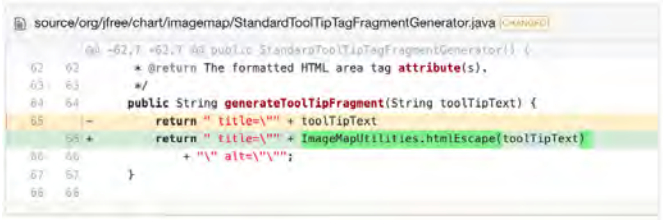


그림 1 Chart10 사람이 작성한 패치 예시

않았을 때 옳은 패치의 순위를 뜻한다. 각 표현 방식에 따른 우선순위화 결과는 각각 AST, 문자열, 벡터에 해당하는 열과 같으며 Average의 경우 각각 두 기법에 대한 유사도 점수의 평균을 우선순위화에 사용한 결과이다. 마지막으로 Total average의 경우 6개의 유사도 계산 기법의 점수 평균을 우선순위화에 사용한 결과이다.

패치 우선순위화가 옳은 패치를 더 높은 순위로 올릴 수 있었던 경우는 전체 21개 버그 중 14개였으며 나머지 7개에 대해서는 모든 유사도 기법들이 옳은 패치를 더 높은 순위로 끌어올릴 수 없었다.

표현 방식별 유사도 기법들의 평균 점수 사용 결과를 보았을 때 AST, 문자열, 벡터 각각 4개, 6개, 11개의 버그에 대해 옳은 패치의 순위를 높임에 따라 벡터가 가장 성능이 좋은 것을 확인하였다. 이는 기법 개별로 살펴보았을 때도 확인할 수 있는데, 코사인 유사도, 유클리디안 유사도를 사용했을 때 각각 10개, 12개의 버그에 대해 옳은 패치의 순위를 높일 수 있었다.

다만 Genealogy context 또한 21개의 버그 중 13개의 버그에 대해 옳은 패치의 순위를 높이는 데 성공했는데 이는 AST 노드 유형을 활용하여 패치 우선순위화를 했기 때문이다. 분석을 위해 구현에 사용된 벡터화 도구 Deckard의 알고리즘은 AST 노드 유형별 개수를 세어 벡터화를 진행한다. Genealogy context 또한 중복 노드 개수와는 다르게 유사도 계산에 AST 노드 유형 정보를 활용하였기 때문에 벡터 기반 유사도 기법들처럼 다수의 버그에 대해 옳은 패치를 높은 순위로 끌어올릴 수 있었다.

chart10를 포함한 7개의 버그에 대해서는 우선순위화를 통해 옳은 패치를 높은 순위로 올릴 수 없었다. 그중 5개의 버그들은 새로운 코드를 삽입하는 패치라는 공통점을 가지고 있었다. 그림 1은 Chart10의 사람이 작성한 패치의 예시를 보여주고 있다. APR 과정에서 생성된 옳은 패치 또한 그림 1과 동일한데 이 경우 htmlEscape이라는 메소드 호출을 새로 삽입하고 있다. 기존의 유사도 계산방식의 경우 의심스러운 스테이트먼트와 후보 패치를 적용한 스테이트먼트간의 유사도를 계산하기 위해 텍스트, 노드 유형 등의 일치하는 부분이 있어야 하지만 이 경우 새로운 코드가 삽입되었기 때문에 기존 유사도 기법으로는 옳은 패치를 높은 순위로 끌어올릴 수 없었다.

표 2 분석 결과

구분	Bug Id	총 개수	Original	AST			문자열			벡터			Total
				Overlap	Gen	Average	LCS	Context	Average	Cos	Eu	Average	Average
패치 우선순위화가 효과가 있는 그룹	Chart1	20,000	973	2,938	138	2433	248	18	150	589	85	85	264
	Chart11	8,002	3,069	4,286	770	4120	77	1,980	478	567	567	567	503
	Chart24	2,182	40	576	9	411	42	364	188	9	9	9	35
	Closure14	20,000	443	8,412	143	8243	4,194	8,629	6087	354	124	124	1471
	Closure38	20,000	15,127	6,415	3,091	5,846	829	810	471	6,581	2,397	6,581	892
	Closure73	20,000	17,175	2,936	5,844	2614	3,960	1,139	2253	4,852	15,550	14741	4369
	Closure92	20,000	683	8,851	137	8428	3,194	5,813	4070	3,124	95	3124	2325
	Closure109	20,000	969	14,431	202	14225	9,176	10,007	9891	884	153	153	7502
	Lang6	17,681	7,625	5,171	1,677	4685	1,385	3,734	2325	123	1,064	857	763
	Lang57	594	79	214	167	153	15	263	219	212	87	107	201
	Math5	3,830	304	3,104	27	2946	863	3,110	2854	105	22	22	2636
	Math33	20,000	19,620	6,992	5,170	6364	2,822	9,723	5649	3,686	3,792	3791	1514
	Math75	11,953	212	3,342	79	3007	664	2,760	1221	985	62	62	628
	Time19	20,000	25	476	4	256	3,248	2,704	2439	5,464	3,756	5363	2011
패치 우선순위화가 효과가 없는 그룹	Chart10	158	26	36	78	48	111	66	74	76	80	80	61
	Lang24	20,000	5,408	11,597	10,208	11991	13,247	10,045	11773	9,873	6,416	7946	11948
	Lang26	20,000	1,050	8,026	10,828	8760	3,942	2,838	3091	9,388	3,207	4058	5639
	Lang51	2,679	1,052	2,626	2,626	2626	2,626	2,626	2626	2,626	2,626	2626	2626
	Math30	20,000	545	11,225	10,289	11093	4,075	4,279	3994	10,178	4,889	8421	7529
	Math34	4,748	146	3,585	4,235	3754	3,887	3,555	3632	4,028	3,526	3979	3780
	Math70	5,623	173	1,404	2,514	1311	500	1,304	825	2,762	831	1140	1161

Overlap: 중복 노드 개수, **Gen**: Genealogy Context, **Context**: Contextual Similarity, **Cos**: 코사인 유사도, **Eu**: 유클리디안 유사도

5. 결론

본 논문에서는 유사도 계산에 사용되는 소스 코드의 표현방식에 따라 패치 우선순위화에 끼치는 영향을 확인하였다. 분석 결과 노드 유형 정보를 활용하는 유사도 계산 방식이 우선순위화에 효과가 있었고 그 중 소스 코드를 벡터화하는 방식이 효과가 있었다. 또한 새로운 코드를 추가하는 패치의 경우 기존의 패치 우선순위화 기법으로는 높은 순위로 끌어올리기 힘들었다. 향후 연구로 해당 표현 방식들에 대한 일반화와 코드 삽입 패치에 대해서 유사도를 측정할 수 있는 방식을 연구하고자 한다.

Acknowledge

이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업 및 한국연구재단의 지원을 받아 수행된 연구임(No. 2017M3C4A706817923, No. 2019R1A2C200641112).

참고 문헌

[1] Moumita Asad et al., “Impact Analysis of Syntactic and Semantic Similarities on Patch Prioritization in Automated Program Repair,” IEEE International Conference on Software Maintenance and Evolution, 2019.

[2] Ripon K. Saha et al., “Elixir: Effective Object-Oriented Program Repair,” IEEE/ACM International Conference on Automated Software Engineering, 2017.

[3] Xuan-Bach D. Le et al., “S3: Syntax-and Semantic-Guided Repair Synthesis via Programming by Examples,” Joint Meeting on Foundations of Software Engineering, 2017.

[4] Ming Wen et al., “Context-Aware Patch Generation for Better Automated Program Repair,” International Conference on Software Engineering, 2018.

[5] Zimin Chen et al., “The Remarkable Role of Similarity in Redundancy-based Program Repair,” arXiv:1811.05703, 2018.

[6] Jindae Kim et al., “Automatic Patch Generation with Context-based Change Application,” Empirical Software Engineering, 2019.

[7] Tao Ji et al., “Automated Program Repair by Using Similar Code containing fix ingredients,” Annual Computers, Software, and Applications Conference, 2020.

[8] Lingxiao Jiang et al. “Deckard: Scalable and accurate tree-based detection of code clones,” International Conference on Software Engineering, 2007.

TF-IDF 기반 유사도 측정 기법을 이용한 KAI 산업체 요구 사항 매핑 테스트케이스 추천 시스템

주가희*, 이효원*, 박지성*, 구나영*, 장혁***, 이선아*,**

경상대학교 항공우주 및 소프트웨어 공학 전공*, AI 융합공학과**, KAI***

E-mail : ghj1222@naver.com, soil0728@naver.com , bear0529@hanmail.net , faker1013@naver.com,ddack@gnu.ac.kr, saleese@gnu.ac.kr

Test Case Recommendation System for KAI Industry Requirements using TF-IDF based Similarity Measurement Techniques

Ga-hee Joo, Hyo-Won Lee, Ji-seong Park, Na-Young Goo, Hyuk Jang, Seonah Lee

Gyeongsang National University, KAI

요 약

KAI 산업체는 항공 전자 분야에서 항공기 설계를 위한 요구 사항과 테스트케이스의 추적성 매핑 관계를 요구 사항 추적성 관리 시스템을 사용하여 관리한다. 이 중에는 요구 사항 R_i에 대한 테스트케이스 T_i의 매핑도 존재한다. KAI 산업체의 테스터들은 이 매핑 관계를 통해 새로운 요구 사항이 들어왔을 때 이와 유사한 내용의 테스트케이스를 추천 받아 새로운 요구 사항에 대한 테스트케이스를 추가할 위치를 찾기를 원한다. 본 논문에서는 해당 KAI 산업체를 위해 요구 사항과 테스트케이스 간의 매핑 관계를 분석하고, TF-IDF를 통한 유사도 측정을 이용하여 테스트케이스를 추천하는 시스템을 제안한다. 테스트케이스 추천을 위한 유사도 측정 방안으로 TF-IDF 방법을 이용한 기본 유사도 측정 방법, 요구 사항과 입력된 요구 사항의 유사도 측정 방법, 테스트케이스와 입력된 요구 사항의 유사도 측정 방법을 점진적으로 제시하고 본 시스템과 가장 적합한 유사도 측정 방법을 결정한다.

키워드 : TF-IDF, 문장 유사도, 요구 사항, 테스트케이스, 주관적 가중치, 추천 시스템

Abstract

The department of avionics in KAI (Korea Aerospace Industries) manages the mapping relationships between requirements and test cases for aircraft design by using requirement traceability management system, where is a mapping relationship of test case Ti for requirements Ri. Through this mapping relationship, KAI testers want to add test cases for new requirements by getting the recommendation of test cases in similar contents to the new requirements. In this paper, we analyze the mapping relationships between requirements and test cases and also propose a system that recommends test cases by using similarity measurements through TF-IDF. As a similarity measurement method for recommendation of test cases, we gradually present a basic similarity measurement method using TF-IDF method, a weighting method, and a similarity measurement method by extracting keyword from test cases to determine the most suitable similarity measurement method for this system.

Key words : TF-IDF, sentence similarity, requirement, testcase, subjective weight

1. 서 론

KAI(한국항공우주산업주식회사)는 대한민국의 안보와 관련하여 항공기술을 개발하는 항공 방위산업체이다. KAI는 기술 개발을 진행하며 요구 사항과 테스트케이스를 통해 프로젝트를 관리한다. 요구 사항의 추적성은 어떤 것이 요청되었는지, 그것의 상태는 어떠한 지, 누가 어떠한 방법으로 관리하고 있는지에 대한 정보를 관리한다는 점에서 그 중요도가 높다 [1]. KAI(한국항공우주산업주식회사) 역시 요구 사항과 테스트케이스를 IBM DOORS [2]로 저장하고 관리한다.

KAI의 요구사항과 테스트케이스들은 양이 매우 많으며 데이터베이스인 IBM DOORS에서는 데이터의 내용에 따라 각 장들로 분류하여 관리한다. 그래서 요구사항과 테스트케이스들이 각각 매핑 되어있지 않고 절 단위로 구성되어 있다. IBM DOORS의 사용자로서의 테스터들은 새로운 요구 사항을 입력할 수 있다. 이 때 사용자는 새로운 요구 사항을 IBM DOORS에 저장하기 위해 비슷한 내용의 테스트케이스가 저장된 위치를 알기 원한다. 그와 가장 높은 유사도를 갖는 테스트케이스를 추천받아 테스트 케이스를 분류하기 위해 참조할 수 있기를 원한다.

테스트케이스 추천 시스템과 관련된 연구 사례로 Sudipto Nandan의 테스트케이스 추천 시스템 연구가 있다 [3]. 이는 유클리드, 코사인, 피어슨 유사도 기법을 사용하여 코드가 변경될 시 이 때마다 변경된 코드와 유사한 테스트케이스를 추천한다.

또한 본 연구에서 사용되는 유사도 비교 방법과 관련된 연구 사례로는 유은순, 최건희 등의 가중치를 부여하여 유사도를 비교하는 방법[4]과 박대서, 김화종의 키워드를 추출하여 유사도를 비교하는 방법[5]이 있다.

본 연구는 KAI 산업체를 위해 IBM DOORS에서 사용되는 테스트케이스의 데이터 형태를 고려하여 새로운 요구 사항을 추가할 시에 추가한 요구 사항과 유사한 내용의 테스트케이스를 추천해주는 메커니즘을 개발한다. 이를 위해서 앞서 설명한 기존 연구와 유사한 유사도 검사 방법을 이용하되 더 나아가 KAI산업체가 이용하는 데이터의 특수성을 고려하여 아래 4장에서 설명하는 유사도 비교 방법을 고안하였다.

본 연구는 유사도 검사 방법으로 TF-IDF를 이용하여 기본 유사도 측정 방법, 가중치를 이용한 측정 방법, 테스트케이스의 유사도와 가중치를 이용한 측정 방법을 제시한다. TF-IDF는 문서의 유사도를 측정하는 기법 중 하나로 문서 내에서 단어의 빈도수를 고려하여 계산하는 방법이다. 기본 유사도 측정 방법은 TF-IDF를 사용하여 요구 사항 간에 전체를 비교하여 유사도를 검사하는 방법이다. 가중치를 이용한 방법은 요구 사항 간에 항목별로 가중치를 달리 두어 TF-IDF 유사도를 검사하는 방법이다. 테스트케이스의 유사도와 가중치를 이용한 방법은 입력된 요구 사항과 기존의 테스트케이스로 TF-IDF를 실행하는 것이 특징이다. 해당 테스트케이스의 TF-IDF 결과를 가중치를 이용한 방법에 더하여 유사도를 측정하는 방법이다. 본 논문에서는 각 유사도 방법을 제시하고, 실험을 통해 세 방법을 비교하였다.

본 논문의 구성은 다음과 같다. 2장은 우리 방법의 대상 데이터와 이를 분석한 내용에 대해 설명한다. 3장은 TF-IDF에 대해 설명하고, 4장은 우리 방법에서 제시하는 세 가지 유사도 측정 방법을 설명한다. 5장은 구현한 데모 시스템에 대해서 보여준다. 6장은 실험 및 결과에 대해서 설명한다. 7장은 논문의 결론을 정리한다.

2. 데이터 분석 및 매핑

본 연구에 사용된 데이터는 요구 사항과 테스트 케이스이다. 요구 사항과 테스트케이스가 무엇이고 어떤 형식을 갖는지 설명한 후 KAI의 요구사항과 테스트케이스 매핑 관계를 만든 과정을 설명한다.

2.1. 요구 사항과 테스트케이스

프로젝트의 성공과 실패를 좌우하는 요소들의 40%가 요구 사항과 관련되어 있다. 성공을 위해서도 요구 사항 정의 및 관리를 잘해야 하며, 실패를 하지 않기 위해서도 요구 사항 정의 및 관리를 잘해야 한다. 일반적으로 부실하게 정의되고 관리된 요구 사항으로부터 출발한 프로젝트는 개발과 테스트 단계에서 많은 에러를 유발하고, 이는 일정 지연과 추가적인 비용을 발생하게 만든다. 이러한 맥락에서 보면, 요구 사항 관리는 프로젝트를 진행하는 데서 정말 중요한 요소임에 분명하다 [6].

테스트 케이스를 생성하기 위해서는 요구 사항으로부터 테스트 케이스 생성에 필요한 정보를 추출해야 한다. 이러한 정보는 요구 사항의 입/출력 정보 및 연산자 정보를 이용하여 추출된다. 요구 사항 규칙과 템플릿을 활용하여 추출된 데이터에서 입/출력 정보 파일과 연산자 정보 파일을 이용하여 해당하는 입/출력 변수 명과 연산자 기호로 변환하면 테스트 케이스 생성에 필요한 데이터를 추출한다 [7].

이렇게 생성된 테스트케이스는 특정 프로그램 경로를 실행해 보거나 특정 요구 사항의 준수여부를 확인하기 위해 개발된 입력 값, 실행 조건, 예상된 결과값의 집합이다. 테스트케이스는 SW의 개발, 운영, 유지보수에 따라 지속적으로 추가, 삭제, 변경된다 [8].

2.2. 요구 사항과 테스트케이스의 형식

KAI에서 사용하는 요구 사항은 Internal Input과 External Input으로 분류되어 있다.

표 1. Internal 요구 사항 형식

INPUT SOURCE	SIGNAL DESCRIPTION	TYPE	NUMBER

Internal 요구 사항의 구성요소에는 INPUT SOURCE, SIGNAL DESCRIPTION, TYPE, NUMBER가 있다. Internal 요구 사항의 INPUT SOURCE는 영어 대문자로만 이루어져 있으며 핵심 항공 용어 키워드로 사용된다. SIGNAL DESCRIPTION에서는 INPUT SOURCE에 대한 조건사항이나 하위 operation에 대한 설명을 나타낸다. TYPE은 SIGNAL DESCRIPTION 요구 사항의 입출력 값의 형태를 나타내고 NUMBER는 가장 중요한 요구 사항으로서 가장 상위에 분류되는 기준이자 핵심이 되는 정보를 담고 있다.

표 2. External 요구 사항 형식

INPUT SOURCE	SIGNAL DESCRIPTION	INTERFACE SPEC

External 요구 사항의 구성요소에는 INPUT SOURCE, SIGINAL DESCRIPTION, INTERFACE SPEC이 있으며 INPUT SOURCE와 SIGNAL DESCRIPTION에 대한 내용은 위의 Internal과 동일하다. INTERFACE SPEC은 Internal의 NUBMER 와 같은 역할을 하며 영어 대문자와 숫자로 이루어져 있고 마찬가지로 External 요구 사항 중에서 가장 중요한 요구 사항이다.

표 3. 테스트케이스 형식

Test Action	Expected Result	Pass/Fail

각 요구 사항에 대한 테스트 케이스는 한 가지 또는 여러 가지가 될 수 있다. 한 테스트 케이스 안에 수많은 명령 데이터와 설정 값, 연산자 정보들이 포함되어 있고 테스트케이스의 구성요소에는 Test Action, Expected Result, Pass/Fail이 있다. 우선 Test Action은 테스트케이스 실행 시 실행되는 명령어이며, Expected Result은 명령어 실행 시 테스트케이스에서 수행되는 결과이다. Pass/Fail은 IEEE 829에 의한 통과/실패 기준을 테스트 요소(기능) 또는 서비스가 통과했는지 실패했는지 여부를 결정하는 데 사용되는 지표이다.

2.3. 데이터 매핑

본 연구에 사용된 IBM DOORS에서 매핑된 요구 사항과 테스트케이스는 절 단위이다. 이에 상세한 요구사항과 테스트케이스 매핑을 위하여 우리는 다음 작업을 수행하였다. 매핑된 데이터는 구현 시험을 위해 사용했다.

2.3.1. 요구사항의 SIGNAL DESCRIPTION 내용과 INPUT SOURCE 키워드를 고려

테스트케이스에서 요구사항의 INPUT SOURCE의 키워드 존재 여부, SIGNAL DESCRIPTION의 행동 기술 내용과 항공 용어들을 먼저 파악한다. 그리고 테스트케이스와 요구 사항의 컴포넌트가 일치하고, 테스트케이스의 소제목과 내용이 요구 사항의 SIGNAL DESCRIPTION과 유사할 경우 해당 요구사항과 테스트케이스를 가장 먼저 매핑하였다.

2.3.2. NUMBER와 INTERFACE SPEC을 고려

2.3.1의 방법으로 먼저 매핑 시킨 후 추가로 테스트케이스 들에서 특정 요구 사항의 NUMBER나 INTERFACE SPEC이 존재하는지 확인하여 해당 요구사항과 테스트케이스의 매핑을 진행한다. NUMBER와 INTERFACE SPEC은 요구 사항 내에서는 가장 중요한 의미적 역할을 갖는다. 하지만 테스트케이스에 나타난 NUMBER와 INTERFACE SPEC은 해당 요구 사항과의 직접적인 의미관계를 나타내는 것은 아니다. 일반적으로 NUMBER나 INTERFACE SPEC에 해당하는 요구 사항의 내용이, 다른 컴포넌트의 테스트케이스에서 부가적인 의미관계를 가질 때 사용된다. 그렇기 때문에 NUMBER/INTERFACE SPEC으로의 매핑은 매핑 관계 내용상의 연관성을 보충해주는 목적이므로 2.3.1이 선행되고 시행하도록 한다.

3. TF-IDF(Term Frequency-Inverse Document Frequency)의 개념

문장 내의 연관성 없는 단어들에 제한을 두고, 문장 내 이슈 단어를 추출하여 분석하기 위해 TF-IDF를 사용한다. TF-IDF는 검색엔진에서 사용하는 텍스트 데이터 알고리즘으로 특정 문서나 문장이 있을 때 특정 단어가 해당 문서 내에서 어느 정도 중요한 의미를 갖는지 통계적 수치를 나타낸다. 또한 문서들 사이에 비슷한 정도인 유사도나 검색 결과 순위를 결정하는 검색엔진에 활용되기도 한다.

우선 TF 기법은 문서가 주어졌을 때 해당 단어가 몇 번 출현했는지를 나타내는 수치이며 식(1)과 같다.

$$TF = tf(t,d) \quad \text{--------} \text{ 식 (1)}$$

식 1의 tf(t, d)는 문장 d에서 단어 t가 몇 번 걸쳐서 나타났는지에 대한 빈도를 구한 값이다.

다음으로 IDF의 식은 식 (2)와 같다.

$$IDF = log\frac{D}{1+df(t)} \quad \text{----} \text{ 식 (2)}$$

식 2는 역문서의 빈도를 표현하기 위한 식이다. '역문서의 빈도'란 한 단어가 문서 전체에서 얼마나 공통적으로 나타나는지를 나타내는 값이다. D는 문서의 총 개수를 의미하고 df(t)는 특정 단어 t가 등장한 문서의 개수이다. 전체 문서의 수를 해당 단어를 포함한 문서의 수로 나눈 뒤, 로그를 취해 얻을 수 있다.

TF-IDF의 수학적 정의는 식 (3)을 통해 볼 수 있다.

$$TF-IDF = tf(t,d) * log\frac{D}{1+df(t)} \quad \text{--} \text{ 식 (3)}$$

식 3을 살펴보면 TF-IDF의 수학적 정의는 단어 빈도와 역문서 빈도의 곱이라 할 수 있다. 이는 여러 문서가 있을 때 특정 문서 안에서 특정 단어가 어느 정도 중요한 의미를 갖는지를 수치화해서 보여준다 [9].

4. 유사도 측정 기능 구현

시스템에 새로운 요구 사항이 입력되면 본문 2.3에서 매핑한 기존 데이터와의 유사도를 측정하는 기능이 필요하다. 요구 사항과 테스트케이스 데이터의 구성요소 형식을 고려하여 유사도 측정 기능을 구현하고자 하였다. 먼저 4.1장에서는 기본 유사도 측정 방법을 설명하고, 4.2장에서는 기본 유사도 측정 방법에 가중치를 이용한 방법을 설명한다. 4.3장에서는 가중치와 테스트케이스의 유사도를 이용한 방법을 설명한다.

4.1. 기본 유사도 측정

유사도 측정 기능을 구현하기 위해서 두 비교 문장이 있을 때 유사도를 계산하는 방법으로 TF-IDF를 사용하고자 한다. 2.2장에서 설명한 바와 같이 우리가 사용하는 데이터는 각각 다른 구성요소 형식을 갖추고 있다. 그래서 TF-IDF를 통해 우리가 사용하는 데이터의 구성요소 형식에 맞는 유사도 측정 방법을 설명한다.

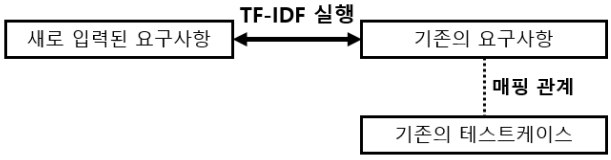


그림 1. 기본 유사도 측정 방법 개념도

기본 유사도 측정 방법에서는 그림 1의 개념도와 같이 새로 입력된 요구 사항과 기존의 요구 사항의 유사도를 TF-IDF로 계산한다. 요구 사항의 형식은 표 1과 같으며 각 구성요소마다 TF-IDF를 실행한 값을 더하여 최종 유사도 값을 구한다. 해당 과정은 그림 2로 나타나있다.

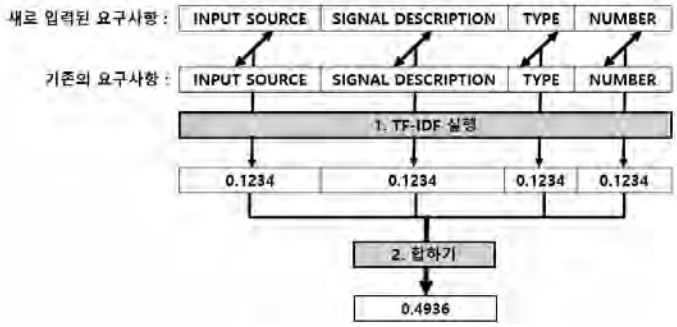


그림 2. 기본 유사도 측정 방법

4.2. 가중치를 이용한 유사도 측정 방법

사용자는 입력한 요구 사항에 대해 내용상 의미하는 바가 유사한 테스트케이스를 추천 받기를 원한다. 요구사항과 테스트케이스의 구성요소들은 각자가 갖는 의미가 다르기 때문에 사용자의 요구에 따라 각 요소의 의미적인 중요도가 다르게 매겨져야 한다.

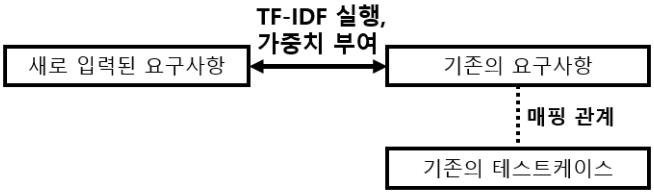


그림 3. 가중치를 이용한 유사도 측정 방법 개념도

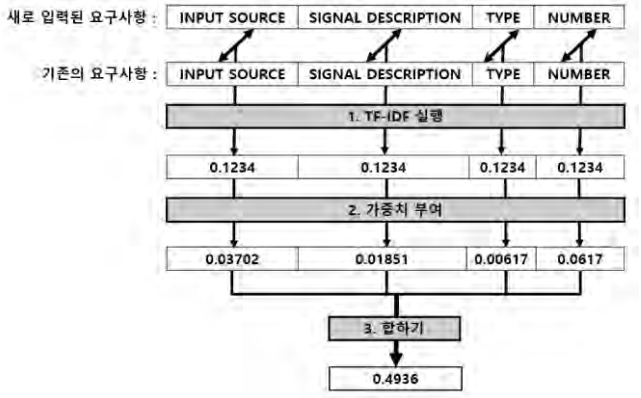


그림 4. 가중치를 이용한 유사도 측정 방법

일반적으로 특정 대상을 평가하기 위해서는 그 대상을 대표하는 다수의 지표를 사용하게 된다. 다수의 지표를 이용하여 종합적으로 대상을 평가할 경우, 각 지표의 중요성을 반영하기 위해 가중치를 부여하는 방안을 채택하고 있다 [10].

본 장에서는 요구 사항 구성요소들의 TF-IDF 실행 결과에 가중치를 부여하고 각 값들을 합하여 유사도를 측정한다. 그림 3은 개념도를 나타내며, 그림 4는 해당 과정을 나타낸다.

개별 구성요소의 가중치를 산정하는 방법은 주관적 방법, 통계적 방법, 사회적 판단 방법이 있다 [10]. 본 연구에서는 주관적 방법을 이용하여 가중치를 산정하고자 한다. 먼저 4.2.1장에서는 요구 사항 구성요소들의 중요도를 파악하여 순위를 매기고, 4.2.2장에서는 중요 순위에 따른 가중치 값의 산정 방법을 제시한다. 4.2.3장에서는 요구 사항 구성요소의 가중치를 산정한다.

4.2.1. 요구 사항 구성요소의 중요 순위 지정

주관적 방법으로 가중치를 산정할 때는 기관 혹은 연구자가 전문가 설문조사를 거쳐 자의적으로 산정한다 [10]. 그러므로 사용자가 판단하는 구성요소들의 의미 파악을 기본적인 바탕으로 한다. 하지만 해당 과정만으로 가중치를 산정하기에는 사용자나 전문가가 아닌 입장에서 각 요소들의 미세한 의미들을 정확히 파악하고 이해하기 어렵다고 판단하였다.

그래서 우선 KAI에서 IBM DOORS를 통해 요구사항과 테스트케이스를 관리하는 책임자와의 미팅에서 실시한 조사로부터 각 요소들의 중요 순위를 지정하였다. NUMBER가 같은 요구 사항들은 핵심이 되는 키워드가 동일하므로 가장 비슷하다. 따라서 NUMBER가 같은 두 요구 사항이 있다면 요구 사항의 다른 구성요소들이 달라도 항상 우선 순위에 위치해야 한다. INPUT SOURCE는 NUMBER에서 가리키는 핵심 항공 키워드를 나타내기 때문에 그 다음으로 중요 순위가 높다. SIGNAL DESCRIPTION은 INPUT SOURCE 키워드의 세부적인 하위 operation을 가리키기 때문에 세번째로 중요 순위가 높고 TYPE은 테스트케이스에서 사용하는 부가적인

형식이기 때문에 중요 순위가 가장 낮다. External 요구 사항의 구성요소에서는 INTERFACE SPEC이 NUMBER와 동일한 역할을 하고 나머지는 Internal 요구사항과 동일하다.

수치적인 가중치 값은 4.2.2장에서 제시하는 바와 같이 산정하도록 한다.

Internal 요구 사항의 구성요소에는 INPUT SOURCE, SIGNAL DESCRIPTION, TYPE, NUMBER가 있다. 조사를 바탕으로 중요 순위를 매기면 다음과 같다.

1. NUMBER
2. INPUT SOURCE
3. SIGNAL DESCRIPTION
4. TYPE

External 요구 사항의 구성요소에는 INPUT SOURCE, SIGNAL DESCRIPTION, INTERFACE SPEC이 있으며, 중요 순위를 매기면 다음과 같다.

1. INTERFACE SPEC
2. INPUT SOURCE
3. SIGNAL DESCRIPTION

4.2.2. 중요 순위에 대한 가중치 산정 방법

중요 순위를 선정하면 그에 따라 각 순위에 매겨지는 가중치의 값들은 다르게 산정되어야 한다. 가중치 값의 산정은 하나의 고정된 계산 모델을 만들어서 해당 모델을 기준으로 삼고자 하였다. 그 이유는 4.2.1장에서 언급했듯이 각 요소들의 미세한 의미 차이들을 정확히 파악할 수 없기 때문에 주관적인 판단으로는 중요 순위만 선정하였다. 그러므로 중요 순위에 대한 가중치 산정에 특정한 기준이 필요하다. 두 번째 이유는 본 연구에서 구현하고자 하는 시스템의 유사도 측정 기능에서는 가중치를 산정해야 하는 그룹이 2개 이상이기 때문에 공통적인 기준을 마련하기 위함이다. 본 장에서 제시하는 가중치 산정 방법은 4.3장에서 설명하는 테스트케이스의 유사도와 가중치를 이용한 유사도 측정 방법에서도 사용된다.

계산 모델은 특정 수열을 이용해 만들고자 하였다. 각 요소에 대한 가중치들은 값의 크기가 달라야 한다. 이런 크기가 다른 값들의 선정을 특정한 기준에 대해 수행하려면 규칙이 있는 값의 나열이어야 한다고 판단하였기 때문이다.

가중치 산정 모델을 만들기 위해 먼저 조건을 설정하였다. 첫 번째 조건은 수열 선정에 대한 조건으로 '중요 순위 1위 요소의 내용이 아예 다른 경우는 이하 순위 요소의 내용이 아무리 비슷하다고 해도, 중요 순위 1위 요소의 내용이 같은 경우보다 유사도가 높을 수 없다'이다. 두 번째 조건은 '가중치가 곱해진 구성 요소들의 유사도 측정 값의 합은 0 ~ 1 사이의 값으로 한다.'이다. 두 번째 조건을 선정한 이유는 4.3장에서 제시하는 가중치와 테스트케이스의 유사도를 이용한 유사도 측정 방법을 통해 알 수 있다. 해당 방법에서는 가중치가 곱해진 구성 요소들의 유사도 합이, 또 다시 하나의 구성 요소가 되어 가중치가 곱해지는 과정을 반복한다. 그래서 가중치가 곱해진 구성 요소들의 유사도 합은 TF-IDF 실행 결과와 동일하게 0 ~ 1 사이의 값을 가져야 한다.

먼저 첫 번째 조건을 충족시키기 위해 수열은 앞선 항들의 합보다 크거나 같아야 하는 식 (4)의 조건을 만족해야 한다.

$$A_n \geq A_1 + A_2 + \cdots + A_{n-1} \quad \text{------ 식 (4)}$$

수열의 첫 번째 값을 1로 두고, 식 (4)를 만족시키기 위해 식 (5)과 같은 수열을 지정하였다. 식 (5)의 수열을 나열하면 1, 2, 4, 8, … 과 같다.

$$A_n = A_0 + \sum_{k=0}^{n-1} A_k \quad (A_0 = 1 \text{이고} n = 1, 2, 3 \ldots \text{일 때}) \quad \text{-식(5)}$$

두 번째 조건인 '가중치가 곱해진 구성 요소들의 유사도 측정 값의 합은 0 ~ 1 사이의 값으로 한다.'를 만족시키기 위해 각 값은 수열 전체 값의 합으로 나눠져야 하며 이는 식 (6)과 같다. 식 (6)에서 n은 수열 A의 항의 개수를 의미하고, x는 구하고자 하는 항의 순서이다.

$$B_x = \frac{A_x}{\sum_{k=0}^{n} A_k} \quad \text{----------식 (6)}$$

중요 순위의 순서에 맞게 앞서 구한 값들의 큰 값부터 적용하여 각 구성 요소의 가중치 값을 산정하도록 한다.

4.2.3. 요구 사항 구성요소의 가중치 산정

표 4. Internal 요구 사항 구성요소 중요 순위 및 가중치

구성 요소	INPUT SOURCE	SIGNAL DESCRIPTION	TYPE	NUMBER
중요 순위	2	3	4	1
가중치	0.2667	0.1333	0.06667	0.5333

표 5. External 요구 사항 구성요소 중요 순위 및 가중치

구성 요소	INPUT SOURCE	SIGNAL DESCRIPTION	INTERFACE SPEC
중요 순위	2	3	1
가중치	0.2857	0.1429	0.5714

4.2.1장에서 Internal 요구 사항과 External 요구 사항의 중요 순위를 선정하였다. Internal 요구 사항은 구성요소가 4개이므로 식 (5)의 수열에서 1, 2, 4, 8의 값을 사용한다. 해당 수열 값에서 식 (6)을 통해 각 구성요소의 가중치 값을 산정하면 표 4와 같다. External 요구 사항은 구성요소가 3개이므로 수열의 1, 2, 4의 값을 사용하며, 마찬가지로 각 구성요소의 가중치 값을 산정하

면 표 5와 같다.

계산한 가중치는 그림 4에 나타난 과정과 같이 각 구성요소의 TF-IDF 값에 곱하고, 모두 합하여 최종 유사도 결과값을 계산할 수 있다.

4.3. 테스트케이스의 유사도와 가중치를 이용한 유사도 측정 방법

4.1장과 4.2장에서는 새로 입력된 요구 사항과 기존의 요구 사항의 유사도 측정만 실행되었다. 본 장에서는 4.2장에서 제시한 가중치를 이용한 방법에, 테스트케이스와의 유사도를 추가하는 방법을 제시한다. 개념도를 제시하면 그림 5와 같다.

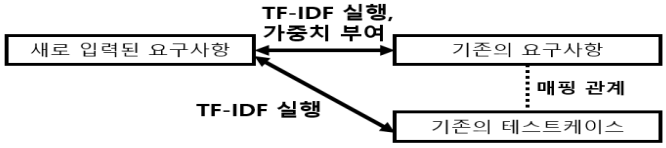


그림 5. 테스트케이스의 유사도와 가중치를 이용한 유사도 측정 방법 개념도

4.3.1장에서는 새로 입력된 요구 사항과 기존의 테스트케이스의 유사도를 측정하는 방법을 제시한다. 그리고 4.3.2장에서는 4.2장에서 수행된 가중치를 이용한 유사도 측정 방법과, 4.3.1장에서 제시한 테스트케이스 유사도 측정 방법을 합쳐서 결과를 도출하는 방법을 제시한다.

4.3.1. 테스트케이스의 유사도 측정 방법

요구 사항끼리의 유사도 측정 방법은 같은 의미를 갖는 구성요소끼리 TF-IDF를 실행하기 때문에 두 비교 대상의 선정이 정확하다. 그리고 사용자가 입력하는 데이터가 요구 사항이므로 요구 사항끼리의 유사도 측정은 본 시스템에서 바탕이 되는 값이다. 반면 요구 사항과 테스트케이스는 각자 기술된 내용과 그 의미가 다르고, 구성요소 또한 다르다. 그래서 요구 사항과 테스트케이스의 유사도를 측정하는 과정은 요구 사항끼리의 유사도 측정보다 중요하지는 않다.

하지만 새로 입력된 요구 사항과 테스트케이스의 유사도 측정 방법을 제시하는 이유는 두 가지가 있다. 첫 번째로 요구 사항끼리의 유사도보다는 의미 관계가 약하지만 관계성이 존재한다는 점이다. 2.3장에서 매핑 과정을 수행한 것처럼 의미 관계의 연결성이 존재하고, 요구 사항 구성 요소 중 SIGNAL DESCRIPTION의 경우에는 테스트케이스에 대한 정보를 포함하기도 한다. 두 번째는 같은 요구 사항에 매핑된 다수의 테스트케이스들 사이에 유사도 결과값의 차이를 두기 위함이다. 일반적으로 하나의 요구 사항에는 여러 개의 테스트케이스가 매핑되어 있다. 그래서 요구 사항끼리의 유사도 측정만 실행하면, 동일한 요구 사항을 가지는 테스트케이스들은 모두 같은 유사도 결과 값을 가진다.

테스트케이스는 소문자, 대문자 그리고 숫자가 포함되어 있고 비교적 내용이 길다. 테스트 케이스의 대문자로 표현된 단어들은 주로 요구사항에서의 INPUT SOURCE에 대한 작업 지시 코드들로 이루어져 있다. 요구사항의 INPUT SOURCE는 주로 항공에서 사용되는 용어들로 해당 용어의 알파벳 앞 글자들만 추출한 약어로 표현하여 주로 대문자로 이루어져 있다. 따라서 테스트케이스 내용 중에서 요구 사항의 내용과 비교할 때, 대문자로만 표현된 단어들과 숫자들이 의미적으로 중요한 요소라고 판단하였다.

그래서 해당 요소들을 추출하여 하나의 문자열로 만들고, 유사도 측정을 위해 요구 사항 또한 구성요소들로 하나의 문자열을 만들었다. 두 문자열로 TF-IDF를 실행하여 테스트케이스와 입력된 요구 사항 간의 유사도를 측정한다. 그림 6에서 해당 과정을 나타냈다.

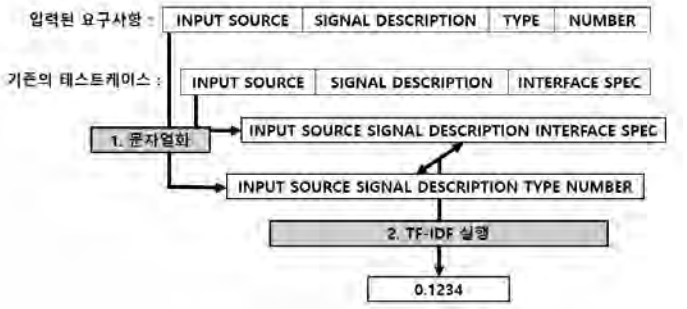


그림 6. 입력된 요구 사항과 기존의 테스트케이스의 유사도 측정 방법

4.3.2. 테스트케이스의 유사도와 가중치를 이용한 유사도 측정 방법

그림 7에서 보이듯이 4.2장의 가중치를 이용한 요구 사항끼리의 유사도와 4.3.1장의 요구 사항과 테스트케이스의 유사도를 합한다. 합할 때는 4.2.2장에서 제시한 가중치 산정 방법을 이용하여 두 값에 가중치를 곱하여 계산한다. 왜냐하면 4.3.1에 언급했듯이 요구 사항끼리의 유사도가 의미적으로 더 중요하기 때문에 중요도의 차이를 둬야한다. 따라서 계산한 가중치는 표 6과 같다.

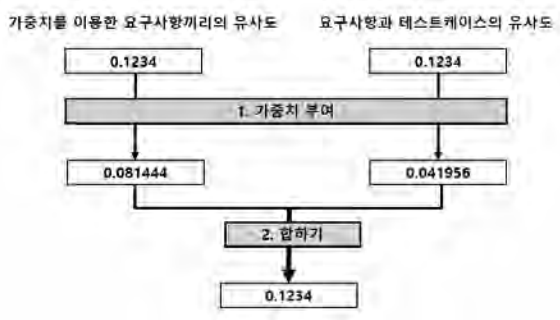


그림 7. 테스트케이스의 유사도와 가중치를 이용한 유사도 측정 방법

표 6. 테스트케이스의 유사도와 가중치를 이용한 유사도 측정 방법의 가중치 산정

구성 요소	가중치를 이용한 요구 사항끼리의 유사도	요구 사항과 테스트케이스의 유사도
중요 순위	1	2
가중치	0.6667	0.3333

5. 시스템 구성

본 시스템은 그림 8과 같이 요구 사항과 조건 설정, 유사도 측정, 추천된 테스트케이스 열람, 테스트케이스 추가 열람, 열람된 테스트케이스 파일화로 구성된다.

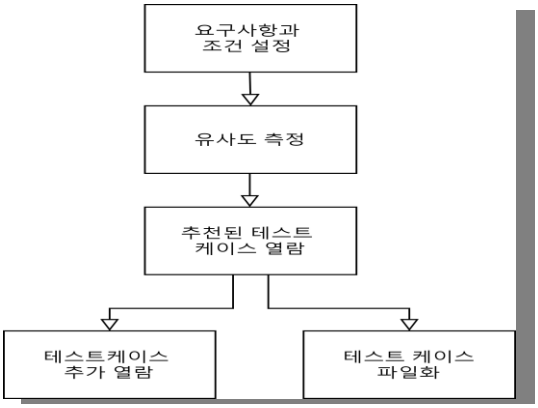


그림 8. 시스템 구성도

시스템의 전체 GUI는 그림 9와 같이 실행된다. 그림 9와 같이 전체 GUI는 세 부분으로 구성된다. 먼저 5.1장에서 설명하는 바와 같이 조건을 설정하는 부분이 있다. 다음으로 5.2장과 같이 새로운 요구 사항을 입력하는 부분이 있다. 마지막으로 5.3장과 같이 입력한 요구 사항에 대한 테스트케이스를 추천한 목록을 보여주는 부분이 있다. 각 부분에 대해서는 하위 장에서 설명한다.

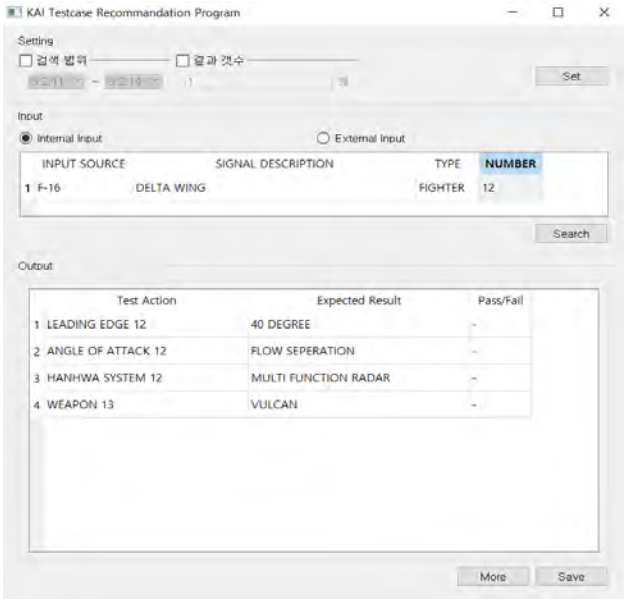


그림 9. 시스템 실행화면 GUI

5.1. 검색 범위 및 추천 개수 설정

사용자는 사용자가 원하는 요구 사항의 범위에서 유사한 테스트를 찾도록 범위를 지정할 수 있다. 또한 추천 받는 테스트케이스의 수를 조정할 수 있다. 이러한 조건 설정은 그림 10과 같은 조건 설정 GUI를 통해 가능하다. 우리는 검색할 테스트케이스의 범위를 리스트 박스로, 열람할 테스트 케이스의 개수를 텍스트 박스로 입력 받도록 구현하였다.

그림 10. 조건 설정 GUI

5.2. 새로운 요구 사항 입력

사용자는 새로운 요구 사항을 입력하여 참조할 수 있는 테스트케이스를 추천 받을 수 있다. 우리는 KAI의 요구 사항의 양식에 맞게 요구 사항 입력 GUI를 구현하였다. 그림 11은 그 결과의 요구 사항 입력 GUI를 보여준다. 그림 11에서 사용자는 먼저 INPUT 테이블 창에서 Internal/External을 라디오 박스로 체크한다. 다음 요구 사항의 각 구성요소를 입력한다. 이후 Search 버튼을 눌러 해당 요구 사항에서 참조할 수 있는 테스트케이스를 검색한다.

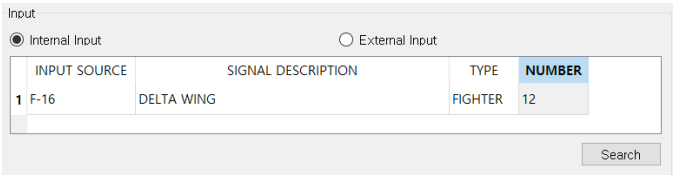


그림 11. 요구 사항 입력 GUI

검색 시 유사도 측정은 총 2단계로 구성되어 있다. 먼저 요구 사항 간의 측정에서는 TF-IDF를 이용하여 기존의 요구 사항과 입력 받은 요구 사항의 유사도를 도출한다. 다음으로 입력 받은 요구 사항과 테스트케이스 간의 유사도를 TF-IDF를 사용하여 측정한다. 도출해낸 두 가지 유사도에 각각의 가중치를 곱하여 더한 값을 최종 유사도 결과값으로 사용하고 수치가 높은 순으로 배열한다.

5.3. 추천 테스트케이스 열람

결과적으로 추천된 테스트케이스를 열람할 수 있다. 추천된 테스트케이스 열람에서는 5.1에서 입력 받은 요구 사항의 범위와 개수를 바탕으로 5.2에서 계산한 유사도 순위가 높은 순으로 OUTPUT 테이블 창에 나열하여 테스트케이스들을 보여준다. 유사도 결과값이 0이 나온 테스트케이스들은 신뢰성이 떨어진다고 판단하여 빨간색 글씨로 나타낸다. OUTPUT창을 아래 그림 12에서 나타내었다.

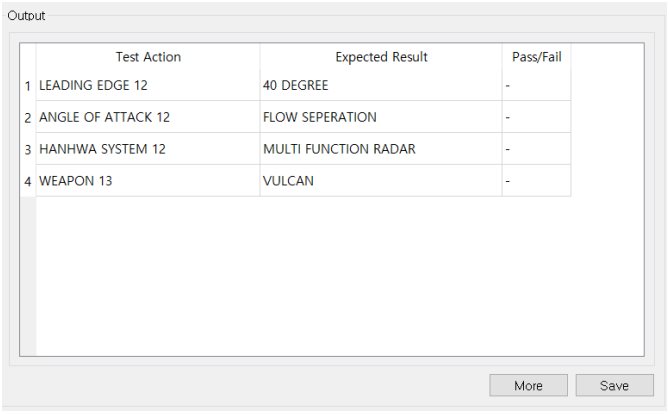


그림 12. output 창 열람 GUI

덧붙여 사용자의 편의를 위해 부가적인 기능으로 더 보기(More)기능과 저장(Save)기능을 추가하였다. 먼저 더 보기(More) 기능은 사용자가 현재 보여진 테스트케이스에 만족하지 못하고 테스트케이스를 하나 더 보려고 할 때, 'more' 버튼을 클릭하여 다음 순위에 해당하는 1개의 테스트케이스를 보여주는 기능이다. 저장(Save) 기능은 'save' 버튼을 눌렀을 때 현재 OUTPUT 창에 보인 테스트케이스 전부를 excel 파일화 시켜 원하는 경로에 저장하는 기능이다.

6. 실험 및 결과

6.1. 실험 설계

요구 사항 유형 (Internal/External)에 상관없이 진행되는 과정은 동일하기 때문에 구현된 전체 시스템에 대한 실험은 Internal 요구 사항이 새롭게 입력되는 경우로 가정하여 진행하였다.

해당 데이터들의 개념도는 그림 13과 같다. 그림 13에서 보여주는 바와 같이 기존 데이터셋으로 표 7의 Internal 요구 사항과 표 8의 테스트케이스가 존재한다. 이러한 상황에서 표 9의 새로운 요구 사항이 입력되면, 유사도 검사 진행을 통하여 추천을 진행한다. 이 때 어떤 테스트케이스를 추천할 지는 유사도 결과에 따른다.

KAI의 실제 데이터는 공개할 수 없으므로, 해당 실험을 위해 Internal 요구 사항 데이터 표 7과 Internal 요구 사항에 대한 테스트케이스 데이터 표 8을 실제 데이터의 형식과 유사하게 제작하였다. 일반적으로 하나의 요구 사항에 다수의 테스트케이스가 매핑 되지만 실험의 간소화를 위해 요구 사항 별로 1개 혹은 2개의 테스트케이스만을 매핑시켰다. 이후 입력할 Internal 요구 사항 표 9를 작성하였다.

그 다음, 4장에서 제시하였던 세 개의 유사도 측정 방법들을 단계적으로 적용시켜 결과의 차이를 확인하고자 3개의 유사도 기법으로 나누었다. 각 유사도 기법의 설명은 다음과 같다.

- 유사도 기법 1: 4.1장의 기본 유사도 측정 방법이다.
- 유사도 기법 2: 4.2장의 가중치를 이용한 유사도 측정 방법이다. Internal과 External 요구사항에 대해 각각 표 4와 표 5의 가중치를 적용했다.
- 유사도 기법 3: 4.3장의 테스트케이스의 유사도와 가중치를 이용하는 유사도 측정 방법이다. 요구사항과 테스트케이스에 대해 표 6의 가중치를 적용했다.

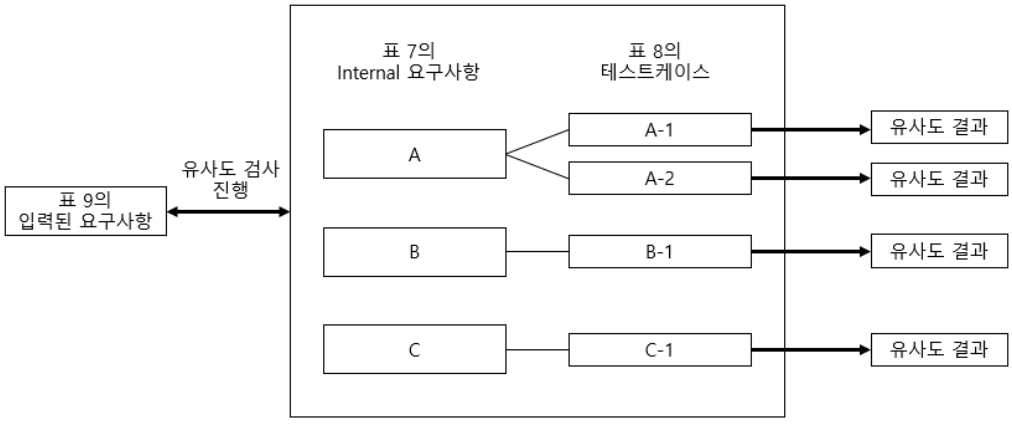


그림 13. 실험용 데이터 개념도

표 7. 기존 Internal 요구 사항 데이터 (테스트용)

Index	Input Source	Signal Description	Type	Number
A	F-16	CROPPEDDELTA WING	FIGHTER	12
B	F-16	BLENDED WING BODY	FIGHTER	13
C	KFX	VARIABLE CAMBER	FIGHTER	12

표 8. 기존 Internal 요구 사항에 대한 테스트케이스 데이터 (테스트용)

Index	Test Action	Expected Result	Pass/Fail
A-1	ANGLE OF ATTACK 12	FLOW SEPERATION	-
A-2	LEADING EDGE 12	40 DEGREE	-
B-1	WEAPON 13	VULCAN	-
C-1	HANHWA SYSTEM 12	MULTI FUNCTION RADAR	-

표 9. 새로 입력되는 Internal 요구 사항 (테스트용)

Input Source	Signal Description	Type	Number
F-16	DELTA WING	FIGHTER	12

표 10. 테스트케이스의 각 유사도 기법 별 결과값

Index	유사도 기법1	유사도 기법2	유사도 기법 3
A-1	3.709	0.9612	0.6747
A-2	3.709	0.9612	0.6783
B-1	2.261	0.3681	0.2454
C-1	2.000	0.6000	0.4339

표 11. 유사도 기법 3을 기준으로 정렬된 테스트케이스

Index	Test Action	Expected Result	Pass/Fail
A-2	LEADING EDGE 12	40 DEGREE	-
A-1	ANGLE OF ATTACK 12	FLOW SEPERATION	-
C-1	HANHWA SYSTEM 12	AESA RADAR	-
B-1	WEAPON 13	VULCAN	-

6.2. 실험 결과

표 9에서 설정한 입력된 요구 사항은 표 8에서 설정한 기존 테스트케이스 중에서 A-2와 가장 유사하게 매핑되는 것을 바람직하다고 본다. 그 이유는 표 7에서 A의 요구 사항과 표 9의 입력된 요구 사항을 비교했을 때 NUMBER, INPUT SOURCE, TYPE이 모두 같고, SIGNAL DESCRIPTION에서는 단어 'WING'이 같다. 그리고 표 7에서 B의 요구 사항과 표 9의 입력된 요구 사항을 비교했을 때는 NUMBER가 다르다. 그리고 INPUT SOURCE, TYPE이 같고, SIGNAL DESCRIPTION에서는 단어 'WING'이 같다. 그리고 C의 요구 사항을 비교했을 때 NUMBER가 같고 INPUT SOURCE와 SIGNAL DESCRIPTION이 다르며 TYPE은 같다. 요구 사항 B는 표 4에서 중요도 순위가 제일 높은 NUMBER가 다르기 때문에 유사도가 제일 낮아야 한다. 그리고 요구 사항 A와 C를 비교했을 때 중요도 순위가 첫 번째인 NUMBER는 둘 다 같다. 하지만 다음 중요도 순위인 INPUT SOURCE는 A가 같고 C는 다르기 때문에 C보다 A가 더 유사하다. 그래서 요구 사항 A는 요구 사항 B와 C보다 유사성이 높은 것이 바람직하다고 본다. 그리고 표 8에서 A-1과 A-2의 테스트케이스를 비교해 볼 때 글자수가 적은 A-2와 매핑되는 것을 바람직하다고 본다. 그 이유는 3장의 식 (3)를 보면 TF-IDF에서 특정 단어 t가 등장한 문서의 개수인 df(t)의 값이 커지면 log안의 값이 작아져 최종 값이 작아지는 것을 통해 알 수 있다.

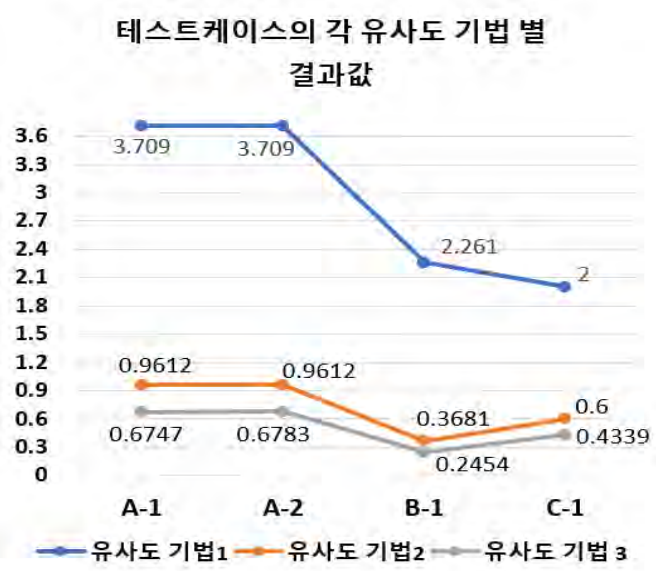


그림 14. 테스트케이스의 각 유사도 기법 별 결과값 그래프

표 10의 각 유사도 기법 별 결과값을 그래프로 나타내면 그림 14와 같다.

6.2.1. 유사도 기법1과 유사도 기법2의 비교

유사도 기법1은 구성요소에 가중치를 부여하지 않은 경우이고 유사도 기법2는 구성 요소에 가중치를 부여한 경우이며 해당 가중치는 표 4와 표 5에 표기되어 있다. 표 10의 유사도 기법1의 결과에서는 B-1의 결과값이 C-1의 결과값보다 더 크다. 그러나 표 7에서 B와 C의 NUMBER를 표 9에서 입력된 요구 사항의 NUMBER와 비교하면 C의 NUMBER는 입력된 값과 같지만 B는 다르다. 표 4에서 NUMBER는 중요도 순위가 1순위이므로 앞서 논의한 바에 의하면 B-1의 결과값이 C-1의 결과값보다 큰 것은 바람직하지 않은 결과이다. 표 10에서 가중치를 부여하는 유사도 기법2에서는 C-1의 결과값이 B-1의 결과값보다 더 큰 값을 보인다. 그래서 이는 기존의 중요도 순위를 충족하는 결과이므로 유사도 기법2와 같이 가중치의 부여가 필요함을 알 수 있다.

6.2.2. 유사도 기법2와 유사도 기법3의 비교

유사도 기법3은 유사도 기법 2에 추가로 테스트 케이스와 입력된 요구 사항의 유사도 측정을 진행한다. 각각의 결과값에 대해서도 가중치를 부여했으며 이는 표 6에 표기 되어있다. 표 10에서 유사도 기법3의 결과를 볼 시 기존의 유사도 기법2에서의 결과로 볼 수 있었던 C-1의 결과값이 B-1의 결과값보다 더 크다는 결과가 유지된 것을 볼 수 있다. 또한 결과 값이 같았던 A-1과 A-2의 결과값에 미세한 차이가 나 A-2의 결과값이 더 커진 것을 볼 수 있다. 이는 테스트 케이스의 우선순위를 매기는데 도움을 주며 결과적으로 표 11과 같이 동순위 없이 내림차순으로 정렬하여 테스트 케이스를 표기할 수 있게 된다.

테스트를 위해 표 7과 표 8을 임의로 만들었기에 실험에서 다루는 데이터는 적다. 하지만 실제 데이터는 내용이 길고 요구 사항에 대한 지시 사항들이 상세히 적혀 있기 때문에 사용자가 유사성을 판단하기 위해서는 테스트케이스와의 유사도를 측정할 필요성이 있다.

7. 결 론

본 논문에서는 해당 산업체를 위한 요구 사항과 테스트케이스의 새로운 매핑 방법과 TF-IDF 기법을 기반으로 하여 유사도 측정 기법을 세 가지로 나누어 무엇이 개발하고자 하는 시스템에 적합한 지 확인하였다. 비교할 두 대상이 명확하지 않다는 점에서 TF-IDF 기법만을 활용하기에 어려움이 있었다.

본 논문에서 제안하는 유사도 측정 기법은 새로운 요구 사항 추가 시 유사한 테스트케이스를 추천하고 이 요구 사항을 추가할 위치를 찾을 수 있다. 이는 해당 시스템 뿐만 아니라 요구 사항과 테스트케이스가 매핑된 데이터를 사용하는 다른 시스템에서도 활용될 수 있고 향상된 성능을 제공한다. 향후 연구로는 본 시스템에서 요구 사항과 테스트케이스의 데이터를 자동화하여 IBM DOORS와 연동시켜 관리하는 기능을 추가할 연구를 계획 중에 있다.

8. 참 고 문 헌

[1] Orlena Gotel, Jane Cleland-Huang, Jane Huffman Hayes, et al., “Traceability Fundamentals”, Software and Systems Traceability”, pp3-22.

[2] https://www.ibm.com/support/knowledgecenter/ko/SSYQBZ_9.6.0/com.ibm.doors.requirements.doc/topics/c_welcome.html

[3] Sudipto Nandan, “Test case recommendation system”, International Journal of Advance Research, Ideas and Innovations in Technology, Volume 5, Issue 2”, 2019.

[4] 유은순, 최건희, 김승훈 "TF-IDF와 소설 텍스트의 구조를 이용한 주제어 추출 연구" pp. 6-7, 2015

[5] 박대서, 김화종 "TF-IDF 기반 키워드 추출에서의 의미적 요소 반영을 위한 결합벡터 제안", pp. 5-6, 2018

[6] “The Standish Group Report”, pp. 2-16, 2014.

[7] 한혜진, 정기현, 최경희, “구조화된 자연어 요구사항으로부터 테스트 케이스 및 스크립트 생성”, pp. 5-6, 2019.

[8] IEEE Std 610.12-1990, “IEEE Standard Glossary of Software Engineering Terminology”, IEEE, 1990. 12.

[9] 이종화, 이문봉, 김종원 “TF-IDF를 활용한 한글 자연어 처리 연구”, pp. 107-109, 2019.

[10] 이정호, 류준호, 정태영, "국가과학기술혁신역량 평가지표의 가중치 산정방법에 관한 연구", 산업혁신연구, 제26권, 제3호, pp. 1-34, 2010.

결함 데이터 수집 통합 프레임워크

한주희[1], 윤수빈[1], 양수진[2], 남재창[1]
[1]한동대학교 전산전자공학부
[2]한동대학교 정보통신공학과
21600763@handong.edu, 21800491@handong.edu, sujinyang@handong.edu, jcnam@handong.edu

An Integrated Framework for Collecting Defect Data

Juhui Han[1], Subin Yun[1], Sujin Yang[2], Jaechang Nam[1]
[1]Handong Global University, Computer Science and Electrical Engineering
[2]Handong Global University, Dept. of Information and Communication Engineering

요 약

4차 산업 혁명 이후, 소프트웨어의 시장이 급성장 함에 따라 소프트웨어 품질 향상이 중요해 지고 있다. 이에 따라 결함 예측에 관한 연구도 활발히 진행되고 있다. 하지만 현재로서는 결함 예측 연구에 필요한 결함 데이터를 수집하는 공개된 프레임워크를 찾기 어렵다는 문제점이 있다. 따라서 본 논문에서는 연구자들이나 개발자들이 결함 관련 연구들을 위해 편리하게 활용할 수 있는 결함 데이터 수집 통합 프레임워크를 구현하였다. 본 프레임워크에서는 결함을 수정한 정보, 결함 정보, 메트릭 정보, 최종적으로 결함인지 아닌지 레이블링 된 파일과 머신러닝을 돌릴 수 있는 파일을 수집한다. 본 프레임워크 통해 결함 예측 연구와 실험에 필요한 결함 정보들을 누구나 손쉽게 얻을 수 있을 것으로 기대된다.

1. 서 론

2000년대 이후 버전 컨트롤 시스템이 대중화되면서 개발 과정을 리포지토리에 저장하여 관리함으로써 소스 코드에 대한 다양한 역사성 정보를 확보할 수 있다[1]. 이에 따라 소스 코드 및 파일의 변경사항을 추적하고 제어하는 시스템과 버그 추적 시스템의 데이터를 기반으로 소프트웨어 결함 예측 연구가 활발하게 진행되었다. 또한 머신러닝 기술을 접목해 결함 예측 모델을 생성하는 연구가 많이 진행되었다. 위와 같은 결함 예측 연구와 실험을 하기 위해서는 여러 종류의 결함 데이터들이 필요하다.

버전 컨트롤 시스템을 활용한 결함 예측 연구가 많이 진행되고는 있지만, 실제로 개발자가 버전 컨트롤 시스템 중 하나인 깃허브에서 결함 정보를 추출 할 수 있도록 지원하는 도구가 거의 만들어지지 않았다[2]. 현재로서는 결함 데이터를 수집하는 툴이 연구자와 개발자가 원하는 형식의 데이터를 제공하지 않아 원하는 데이터를 얻기 위해서는 사용자가 일부 기능을 구현하여 데이터를 얻어야 한다. 또한 결함 데이터보다는 주로 메트릭을 수집하는 도구들만 있기 때문에 연구자와 개발자는 원하는 결함 데이터를 편리하게 얻을 수 없는 상황이다.

따라서 본 논문은 지금 시중에 나와 있는 도구보다 다양한 결함 데이터와 메트릭을 함께 제공하는 결함 데이터 통합 수집 프레임워크를 제시한다. 해당 프레임워크를 오픈 소스로 공개하였다.[1]

[1] https://github.com/ISEL-HGU/DPMiner

2. 결함 예측 관련 연구 동향

2.1 결함 예측 연구

그림1은 소프트웨어 결함 예측의 일반적인 프로세스를 나타낸다. 결함을 예측하는 과정은 크게 3가지로 나뉜다. 1단계는 소프트웨어 아카이브에서 추출한 데이터를 이용해 인스턴스를 생성한다. 2단계는 1단계에서 생성한 인스턴스들을 대상으로 전처리과정을 거치고 마지막 단계인 3단계에서는 전처리 과정을 거친 인스턴스를 대상으로 prediction model을 훈련한다. 1단계에서 추출한 인스턴스는 크게 2가지의 종류가 있는데 소프트웨어 아티팩트로부터 추출한 metric이 있고 buggy 또는 clean, 버그의 수로 레이블이 지정된 데이터가 있다. 본 논문이 제시하는 프레임워크의 역할이 결함 예측 단계 중 데이터를 추출하여 인스턴스를 생성하는 1단계에 해당한다.

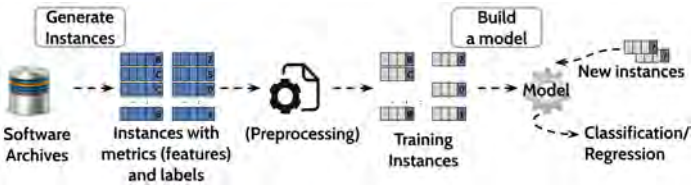


그림 1. 소프트웨어 결함 예측의 일반적인 프로세스 [3]

2.2 기존의 결함 데이터 수집 도구

결함 데이터를 수집하는 데 사용하는 도구 중 대표적으로 PyDriller[2]가 있다. Pydriller는 오픈소스로 공개되어있는 Python 기반의 소스 코드 리포지토리 마이닝 도구이다. 이는 Python library로서 간편하게 사용 할 수 있는 장점이 있고

commit hash, commit author name, date 그리고 Bug Introducing Commits 등의 git commit에 대한 모든 정보를 수집 할 수 있다. 또한 파일 단위로 metric을 추출하는 기능이 있다.

하지만 결함을 수정한 커밋(Bug Fix commits)을 스스로 찾지 못하는 한계점이 있어 자체적으로 결함 데이터에 레이블링하기가 어렵고, 다양한 메트릭을 다루지 않는다. 본 프레임워크는 이러한 한계점을 보완하여 BFC를 자동으로 수집하고, 다양한 메트릭을 만들 수 있도록 하였다.

3. 데이터 수집 절차

결함 예측 모델을 생성하는데 필요한 데이터를 추출하기 위해서는 여러 가지의 데이터와 단계가 필요하다. 제일 먼저 소프트웨어 아카이브인 깃허브에서 결함을 수정한 정보가 있는 커밋을 모은다. 그리고 결함을 수정한 정보가 있는 커밋에 대하여 SZZ알고리즘을 적용하여 결함이 유입된 시점의 커밋을 찾는다. 이 정보와 수집한 메트릭과 결합하여 결함 예측 모델을 만들 수 있는 최종 결과물을 추출할 수 있다.

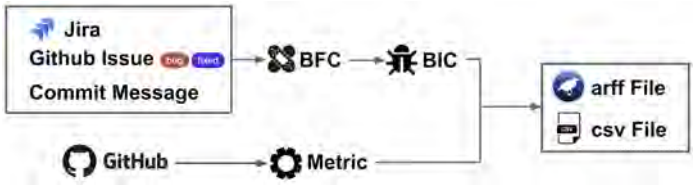


그림 2. 프레임워크의 결함 데이터 수집 절차

본 프레임워크는 5가지의 결함과 관련된 결과물을 추출할 수 있다. 첫 번째로는 리포지토리 URL 목록을 수집한다. 두 번째로는 결함을 수정한 정보를 수집한다. 세 번째로는 결함이 유입된 커밋 정보를 수집한다. 네 번째는 메트릭을 수집한다. 다섯 번째로는 실제 결함 예측 모델을 생성하는데 사용되는 Weka 도구에서 사용할 수 있는 arff파일과 csv파일을 선택적으로 생성한다.

그림2는 해당 논문이 제시하는 프레임워크의 전체적인 과정을 나타낸다. 결함을 수정한 정보(BFC)를 3가지의 방법을 통해 수집하고 이를 활용하여 결함이 유입된 커밋(BIC)을 수집한다. 그리고 Github로부터 추출한 Metric과 결함이 유입된 커밋정보를 이용하여 arff file과 csv file을 최종적으로 생성한다.

4. 프레임워크 설계

본 프레임워크의 목적은 결함 연구에 필요한 다양한 데이터들을 수집할 수 있는데 있다. 따라서 결함 연구에 많이 필요로 하는 Bug Fixing Change(BFC), Bug Introducing Change(BIC), Metric, 깃 레파지토리 URL 목록 등 다양한 데이터들을 수집할 수 있게 설계하였다.

4.1 데이터 수집 도구 설계

4.1.1 깃허브 레파지토리 URL 목록 수집

개발자가 자신의 연구 및 실험을 위해 특정 조건의 오픈소스 프로젝트를 필요로하는 경우가 존재한다. 이에 따라 버전 관리 시스템 및 오픈 소스 저장소인 깃허브에서 사용자가 원하는 조건에 맞는 레파지토리 URL 리스트를 추출한다.

URL 리스트를 추출하기 위해서 GitHub REST API 중 Search API를 사용한다. 깃허브에서 제공하는 Search API는 조건에 대한 정보를 쿼리 형식으로 송신을 하면 한 페이지에 100개의 레파지토리 URL 목록을 수신 받을 수 있다. 본 프레임워크는 여러 개의 쿼리를 통해 한 쿼리당 100개씩 레파지토리 URL 목록을 수집하여 조건에 해당하는 프로젝트 레파지토리 URL을 모두 수집 할 수 있다.

다음 그림3의 예제는 자바로 구성된 레파지토리를 요청하는 쿼리문이다. 해당 쿼리문에 대한 응답은 언어가 자바로 제한된 리포지토리를 한 페이지당 100개의 목록으로 반환이 된다. 언어 외에도 포크의 수, 마지막으로 프로젝트를 수정한 날짜, 프로젝트를 생성한 날짜를 조건으로 넣어 사용할 수 있다.

```
http://api.github.com/search/repositories?q=language:java
```

그림 3. 언어가 자바인 리포지토리 URL 목록을 수집하는 쿼리문 예제

4.1.2 결함 수정 정보 수집 도구 설계

결함 데이터 수집의 가장 기본이 되는 것은 결함이 수정된 정보가 있는 커밋인 BFC를 찾는 것이다. 결함이 수정된 정보가 있는 커밋으로 부터 결함 정보가 있는 커밋을 찾을 수 있기 때문이다. 결함이 수정된 정보가 있는 커밋을 수집하기 위해 3가지 방법을 사용하였다.

첫째, Jira 이슈에서 결함 정보를 수집하는 것이다[4]. Jira는 이슈를 관리하기 위한 저장소이다. Jira는 이슈의 본질을 의미하는 레이블과 이슈의 진행 과정인 상태 정보로 프로젝트를 관리한다. Jira는 버그 레이블이 붙은 이슈들만 모아서 수집 할 수 있다. 또한 Jira는 이슈를 고유한 키로 관리하므로, 이슈 키가 제목에 포함된다면 해당 커밋은 결함과 관련이 있다고 예측 할 수 있다. 이를 활용하여 Jira에서 결함 데이터 정보를 수집한다.

둘째, 깃허브 이슈에서 결함 정보를 수집하는 것이다. 깃허브 이슈는 프로젝트에서 개선되어야 할 것이나 해결되어야 하는 것을 의미한다. 깃허브도 각각의 이슈에

레이블이 주어져 있다[4]. 완전히 해결된 결함을 가지고 버그 유발 커밋을 예측해야 하므로, 레이블이 버그이고 상태가 닫힘일 경우에만 해당 결함 정보를 수집한다.

마지막으로, 커밋을 할 때 생기는 커밋 메시지를 활용해 결함 정보를 수집하고자 하였다. 개발자들은 커밋을 할 때마다 의미 있는 커밋 메시지를 함께 작성하는 것이 관례이다[4]. 이 점을 활용하여 커밋의 메시지에 기본적으로 'fix'나 'bug' 단어를 포함하는 경우나, 개발자가 임의로 키워드를 넣어서 결함 정보를 수집할 수 있도록 하였다.

4.1.3 결함 정보 수집 도구 설계

SZZ알고리즘[5]은 버전 관리 시스템을 기반으로한 프로젝트를 대상으로 결함 수정 정보를 사용하여 결함 정보를 자동으로 수집하는 알고리즘이다. 본 프레임워크에서는 내부적으로 세 가지 방법을 통해 결함 수정 정보와 관련있는 커밋을 수집한 뒤 SZZ알고리즘을 사용하여 결함 정보를 얻는다.

본 프레임워크에서는 내부적으로 세 가지 방법을 통해 결함 수정 정보와 관련있는 커밋을 수집한 뒤 SZZ알고리즘을 사용하여 결함 정보를 얻는다.

본 연구에서 제공하는 SZZ 알고리즘은 두 가지가 존재한다. 첫 번째 SZZ 알고리즘은 git blame 기반인 B-SZZ 알고리즘이다. B-SZZ 알고리즘은 버그를 수정한 커밋의 수정된 라인에 git blame을 실행하여 버그가 유입된 커밋을 찾는 알고리즘이다. 즉, 해당 알고리즘은 버그를 수정한 커밋에서 직전 커밋을 반환한다. 따라서 코스메틱 수정을 버그 유입이라고 잘못 판단하는 문제점이 있다.

두 번째 SZZ 알고리즘인 AG-SZZ 알고리즘은 Annotation Graph를 이용하여 빈 라인 수정, format 변화, 주석 수정 그리고 너무 많은 파일을 한번에 수정하는 이상치(outlier) BFC를 제거함으로써 B-SZZ의 문제점을 해결하였다. Annotation Graph는 결함이 수정된 정보를 포함하고 있는 커밋부터 처음 커밋에 포함되는 모든 라인을 노드로 가진다. 또한 이는 두 개의 커밋 사이에 새로운 라인이 추가된 경우에는 노드를 추가하고 수정된 경우 두 커밋들의 노드 간 엣지를 연결한 그래프이다. 제일 처음의 커밋부터 결함 수정 정보가 들어 있는 커밋까지 Annotation Graph를 만들고, 결함을 수정한 라인에 대하여 DFS알고리즘을 적용하여 결함 유발 라인을 찾아낸다[3].

4.1.4 Metric 수집 도구 설계

메트릭은 결함 예측을 위한 소스 코드의 정보이다. 본 프레임워크는 기존의 실시간 결함 예측 연구인 Tan et al.[6], Jiang et al.[7] 그리고 Kamei et al.[8]에서 사용된 것과 동일한 메트릭을 기반으로 하며, 개발 과정에서 일어나는 결함을 실시간으로 예측하기 위해 커밋단위로 메트릭 수집한다. 메트릭은 크게 3가지 카테고리로 나눌 수 있다. 1) Characteristic Vector, 2) Bag of Word Metric, 3) Meta data, Characteristic Vector와 Bag of Words는 Tan et al.[6], Jiang et al.[7]에서 사용되었고, Meta data는 Tan et al.[6], Jiang et al.[7], Kamei et al.[8] 세 연구에서 사용된 메트릭들을 통합한 것이다.

1) Characteristic Vector : Characteristic Vector는 소스 코드의 구조 변화를 나타내는 메트릭이다. 이는 commit에서 수정된 소스 코드와 수정되기 전 소스 코드를 Abstract Syntax Tree로 나타낸 후, 각 소스 코드의 tree를 비교하여 차이점을 고유벡터로 표현한다. 따라서 소스 코드의 변화를 보다 자세하게 알 수 있다. Characteristic Vector가 수집하는 벡터는 4가지로 추가, 삭제, 업데이트, 이동이 존재한다. 변경된 노드를 찾기 위해서 GumTree Diff 알고리즘[9]기반 도구인 gumtree를 사용한다.

2) Bag of Words : Bag of Words는 소스 코드와 커밋 메시지에서 문장을 단어 단위로 해체 한 후 단어의 출현 빈도수를 측정하는 메트릭이다. 이는 함수 이름을 잘못 사용하여 일어난 결함을 찾는데 유용하다. 단어 어간 추출을 위해 weka[10]의 SnowBall Stemmer를 사용한다.

3) Meta data : Meta data는 총 25가지의 결함 예측 메트릭이다. Kamei et al.[8] 메트릭 13개, Tan et al.[6] 메트릭 14개 그리고 Tan et al.[6] 메트릭의 subset인 Jiang et al.[7] 메트릭은 8개가 존재한다. 수집 가능한 Meta data는 다음과 같다 : 커밋에서 수정된 라인, 추가된 라인, 삭제된 라인, 수정된 chunk, 추가된 chunk, 삭제된 chunk, 수정된 라인의 분포, 소스 파일이 BIC였던 누적 횟수, 개발자 ID, 파일의 나이, 소스 파일 수정 누적 수, 소스 파일을 수정한 개발자 수, 소스 파일이 수정된 간격, 개발자의 경험, 최근 개발자 경험, subsystem에 대한 개발자 경험, 커밋한 시간(0, 1, 2, ..., 23), 요일(월, 화, …, 일), 커밋에서 서로 다른 subsystem, 디렉토리, 파일 경로 수, 소스 파일의 고유 커밋, 파일 이름, 파일 경로 이름, 소스 파일이 수정되기 전 라인 수, label(bug or clean).

4.2 프레임워크 확장성

그림4는 결함 정보 수집 통합 프레임 워크의 UML 일부분이다. 그림4를 보면 각 기능들마다 확장 가능성을 남겨두기 위해 기능 마다 인터페이스를 활용하여 설계하였다. 앞으로 새로운 기능이나 새로운 데이터들을 추가하고자 할 때

인터페이스를 활용해 손쉽게 프로그램에 기능을 추가 할 수 있다.

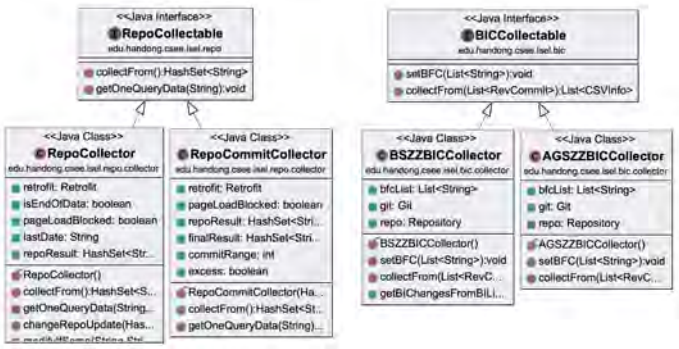


그림 4. 프레임워크 UML 일부

4.2.1 Repository URL List 수집

깃허브 서치 API에는 리포지토리 이외의 다른 정보도 수집할 수 있는 기능을 제공한다. 현재 본 프레임워크는 리포지토리 리스트와 커밋 바운더리에 따른 리포지토리 리스트를 반환한다. 사용자는 하나의 쿼리문에 대한 결과를 반환하는 getOneQueryData()와 전체 쿼리 결과에 대한 정보를 반환하는 collectFrom()기능만 추가하면 언제든지 필요한 데이터 수집 기능을 확장하여 사용할 수 있다.

4.2.2 Bug Fixing Commit 수집

본 프레임워크가 BFC를 수집하는 방법은 총 3가지 방법이 있다. Jira 키워드를 이용하는 방법, 깃허브 이슈를 이용한 방법, 그리고 깃 커밋 메시지를 활용한 방법이다. BFC를 수집하는 기능인 collectFrom()만 구현하면 전체적인 프로그램에 알맞게 실행시킬 수 있다.

4.2.3 Bug Introducing Commit - SZZ 알고리즘

Bug Introducing Commit을 수집하기 위해서 본 프레임워크는 git Blame을 기반으로 한 B-SZZ와 annotation graph를 적용한 AG SZZ 알고리즘을 제공한다. 이 외에도 다양한 SZZ 알고리즘이 존재한다. 사용자는 BIC을 찾기 위해 필요한 BFC를 설정해주는 setBFC() 기능과 실제로 SZZ 알고리즘을 적용하여 BIC을 모으는 기능인 collectFrom() 을 구현하면 제공하는 SZZ 알고리즘 외에도 다른 알고리즘을 추가하여 사용할 수 있다.

4.2.4 Metric

Metric을 수집하기 위해서 본 프레임워크는 인터페이스에 명시된 함수들인 BIC를 설정해주는 setBIC() 함수와 Metric을 생성하기 위해 필요한 정보를 모으는 기능인 collectFrom() 함수를 구현하면 새로운 메트릭을 수집할 수 있다.

5. 기존 도구와의 성능 비교

대표적인 데이터 마이닝 도구인 PyDriller와 본 논문에서 제안하는 결함 데이터 통합 수집 프레임워크를 비교하였다.

Pydriller가 BFC를 수집하지 못하여 해당 논문에서 제시한 프로그램을 통해 수집한 BFC를 인풋으로 하여 Pydriller가 수집한 BIC와 본 프레임워크가 수집한 BIC을 비교하였다. 첫 번째로 bval, aries, tiles 그리고 giraph 프로젝트에서 BFC를 25개씩 수집하여 총 100개의 BFC를 수집하였다. 두 번째로 결함 데이터 통합 수집 프레임워크의 B-SZZ 알고리즘을 사용하여 100개의 BFC에 대한 BIC을 수집하였다. 세 번째로는 Pydriller를 사용하여 100개의 BFC에 대한 BIC을 수집하였다.

그 결과 DPMiner는 총 106개의 BIC를 수집하였고, PyDriller는 총 218개의 BIC를 수집하였다. 그림 5를 참고해서 보면 교집합에 해당하는 BIC의 개수는 104이고, DPMiner만 찾은 BIC의 개수는 2개이고, PyDriller만 찾은 BIC의 개수는 218개이다.

DPMiner만 찾은 BIC를 분석한 결과, 2개의 BIC 모두 PyDriller는 못 찾았지만 DPMiner가 잘 찾은 경우인 것을 확인할 수 있었다. PyDriller만 찾은 BIC를 분석한 결과, DPMier가 찾지 못한 BIC를 찾은 경우가 8개, XML 파일 또는 테스트 파일의 내용을 추적하여 찾은 BIC가 103개 였다. 나머지 13개의 BIC는 주석 또는 Delete 된 코드가 그대로 insert 된 경우였다. 그림5는 delete 된 코드가 수정 없이 그대로 insert 된 예제이다. 그림6은 PyDriller만 찾은 BIC를 파이 그래프로 정리한 것이다.

결과를 분석하자면 PyDiller가 XML 파일 또는 테스트 파일의 내용을 추적하여 찾은 BIC는 PyDiller를 사용하는 사용자가 따로 필터 코드를 작성하면 제외될 부분이다. 일반적으로 결함 예측 분야에서 주석 소스 코드는 버그를 유발하는 코드가 아니라고 간주한다[3]. 따라서 주석과 관련해서 찾은 BIC는 결함 정보가 아니며, 이는 노이즈 정보가 된다. 또한 delete 된 코드가 그대로 insert 된 경우는 버그를 수정한 경우가 아니기 때문에 이를 추적한 결과는 노이즈 정보가 된다. 따라서 PyDriller가 찾은 13개의 BIC는 노이즈 정보가 된다. DPMiner는 주석과 같이 버그를 유발하는 코드가 아닌 것은 제외하고, delete 된 코드가 그대로 insert 된 경우 diff 알고리즘을 통해 제외하기 때문에 PyDriller에 비해 노이즈 BIC가 더 적은 것을 확인 할 수 있다. 따라서 본프레임워크가 정확한 결함 데이터를 찾는데 더 효율적이다. DPMiner가 찾지 못한 8개의 경우를 분석하여 추후에 프레임워크를 보강할 계획이다.

그림 5. 본프레임워크와 PyDriller가 수집한 BIC 결과 벤다이어그램

```
-                if(i == 0) {
-                        char temp = array[i].charAt(0);
-                        result += Character.toString(temp);
-                }
+        if(i == 0) {
+                char temp = array[i].charAt(0);
+                result += Character.toString(temp);
+        }
```

그림 6. delete된 코드가 그대로 insert된 diff 예제

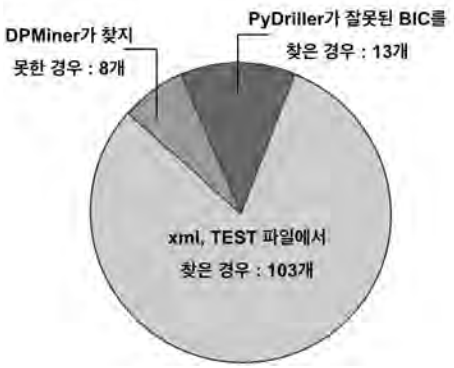


그림 7. PyDriller만 수집한 BIC 결과 파이 그래프

6. 결론 및 향후 계획

본 논문에서 제시하는 프레임워크를 활용하면 결함 예측 연구에 필요한 다양한 데이터를 수집할 수 있다. 해당 프로그램은 개발자의 편의를 위한 리포지토리 URL 리스트를 추출하는 기능이 있다. 또한 세 가지의 방법으로 BFC를 수집하고, 수집한 BFC에 두 가지의 SZZ 알고리즘을 적용하여 BIC을 수집한다. 그리고 총 5개의 metric을 추출할 수 있는 기능이 있다. 본 프레임워크는 Interface를 활용하여 확장성을 고려하여 설계되었기 때문에 사용자가 필요로 하는 기능을 추가해 다양한 기능을 제공하는 프레임워크로 성장할 가능성이 있다.

SZZ 알고리즘은 그 자체에 한계가 존재하여 추후에 더 개선된 SZZ를 개발하여 확장할 계획이다. 더 개선된 SZZ는 이전보다 노이즈가 없는 데이터를 수집할 수 있고 이로 인해 개발자들이 더 성능 좋은 예측 모델을 만들어 낼 것이라 기대할 수 있다. 또한 현재의 프레임워크는 Java 언어로만 구성된 프로젝트에 대한 데이터 수집 기능을 지원하고 있는데, 이후에는 여러 언어를 지원할 수 있도록 프로그램의 기능을 확장할 예정이다. 그리고 현재에는 이슈 트랙커 중 Jira에서만 결함 수정 정보를 수집해오는데 향후에는 Bugzilla 등 다양한 이슈 트랙커에서도 결함 수정 정보를 수집해 올 수 있는 기능을 추가할 예정이다.

※ 본 연구는 과학기술정보통신부와 정보통신기술 진흥센터의 2020년도 소프트웨어 중심대학 지원사업(2017-0-00130)과 2020년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2018R1C1B6001919).

7. 참고 문헌

[1] Jaechang Nam, (2014).Survey on Software Defect Prediction, HKUST PhD Qualifying Examination

[2] Spadini, D., Aniche, M., & Bacchelli, A. (2018). PyDriller: Python Framework for Mining Software Repositories. In ESEC/FSE 2018: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 908-911). Association for Computing Machinery (ACM). https://doi.org/10.1145/3236024.3264598

[3] S. Kim, T. Zimmermann, K. Pan and E. J. Jr. Whitehead, "Automatic Identification of Bug-Introducing Changes," 21st IEEE/ACM International Conference on Automated Software Engineering (ASE'06), Tokyo, 2006, pp. 81-90, doi: 10.1109/ASE.2006.23.

[4] 양수진, 임성빈(2018).소프트웨어 품질 관리를 위한 결함 데이터 수집 도구에 관한 연구 A Study on a Buggy Data Collection Tool For Software Quality Control. Proceedings of the 21th Korea Conference on Software Engineering (KCSE 2019) (pp.276-279)

[5] Śliwerski, J., Zimmermann, T., & Zeller, A. (2005). When do changes induce fixes?. ACM sigsoft software engineering notes, 30(4), 1-5.

[6] Tan, M., Tan, L., Dara, S., & Mayeux, C. (2015, May). Online defect prediction for imbalanced data. In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (Vol. 2, pp. 99-108). IEEE.

[7] Jiang, T., Tan, L., & Kim, S. (2013, November). Personalized defect prediction. In 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 279-289). Ieee.

[8] Kamei, Y., Shihab, E., Adams, B., Hassan, A. E., Mockus, A., Sinha, A., & Ubayashi, N. (2012). A large-scale empirical study of just-in-time quality assurance. IEEE Transactions on Software Engineering, 39(6), 757-773.

[9] Falleri, J. R., Morandat, F., Blanc, X., Martinez, M., & Monperrus, M. (2014, September). Fine-grained and accurate source code differencing. In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering (pp. 313-324).

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software: An update,” SIGKDD Explor. Newsl., vol. 11, no. 1, pp. 10-18, 2009.

단어 임베딩 유사도 측정을 통한 함수 구성요소에 사용된 단어 간 적합성 평가

남성국, 서영석*
영남대학교 컴퓨터공학과
sd05031@yu.ac.kr, ysseo@yu.ac.kr

Relevance evaluation of terms in functions through word embedding similarity

Seong-Guk Nam, Yeong-Seok Seo*
Department of Computer Engineering, Yeungnam University

요 약

개발자들은 자신의 코드를 쉽게 이해 할 수 있게 하기 위해, 주석을 주로 작성한다. 하지만 실제로 개발자들은 주로 바쁘며, 주석을 작성할 시간이 없을 만큼 시간이 부족하다. 이러한 경우 협업하는 다른 개발자가 코드를 읽고 이해하기 까지 많은 시간이 소요 될 것이다. 만약 작성한 코드의 가독성이 높아진다면 쉽게 내용을 유추하여 위와 같은 문제로 불편함을 겪는 일이 조금은 줄어들게 될 것이다. 코드의 가독성을 향상 시키기 위해 본 논문에서는 작성된 파이썬 함수에서 사용되는 식별자 이름들이 함수 이름과 얼마나 연관 관계가 있는지 평가하는 기법을 제안하고자 한다. 이를 위해 워드 임베딩을 이용하여 작성된 함수에 사용된 단어들의 벡터 값들을 통해 유사도를 계산하고 단어 간의 유사도가 높은 경우 가독성이 높은 코드라 평가할 수 있도록 하였다. 제안하는 기법을 검증하기 위해 GitHub와 Stack Overflow에 있는 코드 데이터 셋을 수집하여 본 기법을 적용 시켜보았다.

1. 서 론

현대의 소프트웨어 산업에서의 개발 규모의 증가에 따라 개인이 수행하는 프로젝트의 모든 코드를 작성 및 관리하는 것이 불가능에 가까워졌다. 개발 및 유지보수를 다른 개발자와 함께 협업을 통해 수행하여야 한다. 가독성이 높은 코드는 협업하는 다른 개발자가 빠르게 이해하는데 도움을 주며, 추후 유지보수시에도 다른 개발자가 발생한 문제를 빠르게 해결하는데 크게 영향을 미친다. 이러한 이유들로 인해 작성된 코드의 가독성을 향상 시키기 위해 Refactoring을 수행하기도 한다.

코드의 가독성을 향상 시킬 수 있는 요인들은 다양하게 존재하지만, 함수를 구성하는 다양한 식별자들과 함수들의 이름에 대해서만 이야기 한다. 작성된 소스코드의 정적 분석을 수행할 때는 자연어와 유사한 형태를 보인다. 작성된 문서를 통해 함수의 기능을 빠르게 이해 하는 방법도 있으나, 그러한 도움을 받지 못하는 경우 또한 존재 할 것이다. 만약 식별자 및 함수의 이름이 역할과 기능에 대해 직관적으로 설명 할 수 있다면, 소스 코드 이해에 큰 영향을 미칠 것이다. 실제로 식별자를 통해 식별자의 역할을 잘 설명 할 수 있는 이름으로 작성된 코드에서 이해에 소요되는 효율성 차이를 보인다는 결과가 있다[1]. 따라서 소스코드의 정적 분석을 수행할 때의 가독성을 향상 시키기 위해서는 적절한 함수의 이름 정의가 필요하며, 함수의 이름과 연관성이 높은 단어들을 사용하여 식별자를 정의 및 함수를 구성하여 소스코드를 보는 사람이 코드를 쉽고 직관적이게 이해 할 수 있도록 작성하는 것이 중요하다.

만약 모든 코드에 대해 빠른 이해를 할 수 있다면 좋겠지만 이는 쉽지 않다. 전체 코드에서의 식별자의 이름이 잘 정의되었는지 혹은 연관성 있는 식별자들로 구성 되었는지에 대해 여부를 검사하기 위해서는 전체적인 프로그램의 흐름을 상세히 이해하여야 할 것이며, 만약 프로그램이 다양한 기능들을 제공하고 있다면 자동화 과정을 통해 처리하기는 쉽지 않을 것이다.

따라서, 본 논문에서는 python 언어로 작성된 코드에 포함된 함수들을 대상으로, 함수 내 식별자들과 함수들의 이름이 적절하게 작성되었는지 검사를 수행하고자 한다. 함수를 대상으로 연구를 진행하는 이유로는 함수는 특정한 기능을 수행하는 뭉쳐진 코드들의 최소 단위로 볼 수 있기 때문이다. 하나의 함수가 한 두가지의 특정 기능을 수행한다면, 함수에 사용된 식별자들의 역할이 명확해 질 것이며, 수행하는 역할을 파악 할 수 있을 것으로 본다. 또한 함수의 이름은 함수의 흐름 또는 함수가 수행하는 기능을 이해하기 위한 수단으로 사용 할 것이다. 실험을 통해 함수의 이름과 함수의 식별자들에 사용된 단어들의 유사도 분석을 통해 함수 구성요소의 이름의 적합성을 평가하고자 한다. 함수의 식별자들이 적절한 단어로 구성되었는지 평가하기 위해, 함수의 이름에 사용된 단어와 연관성을 보일 것이라 가정한다. 실험을 통해 식별자 및 함수 구성 수준이 높다고 평가되는 코드가 구성 수준이 낮다고 평가되는 단어 셋 간 유사도가 더 높게 나타났다.

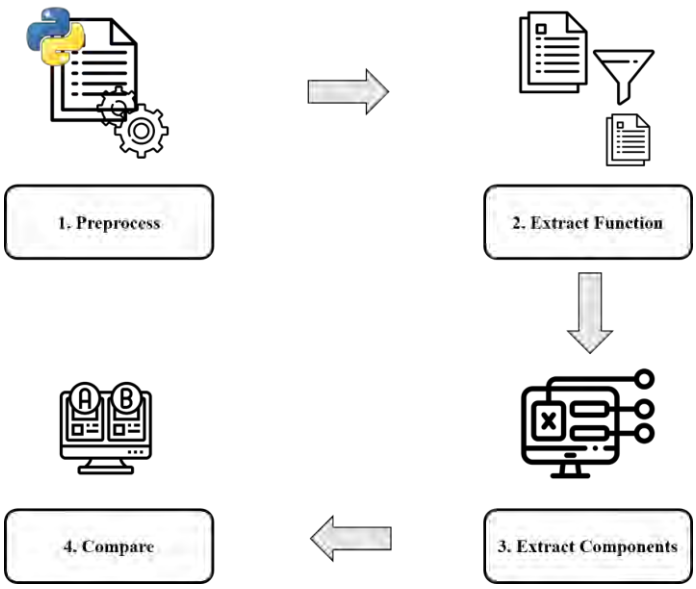


Fig. 1 Overall approach

본 논문의 2장에서는 연구 진행과정에 대해 소개 하며 3장에서는 실험 과정 및 결과 4장에서는 결론 및 향후 연구를 기술한다.

2. 접근법

단어 임베딩(Word Embedding[2]) 집합 간 유사도 측정을 통해 함수 이름에 사용된 단어들과 식별자에 사용된 단어들의 적합성 평가를 위해 Fig. 1와 같은 과정을 수행한다. 해당 파트에서는 접근법에서 제시하는 각각의 단계에서 수행하는 작업들에 대해 세부적인 설명을 진행한다.

2.1 전처리

프로그램 코드를 작성함에 있어 부가적인 내용들을 작성하게 되면, 코드를 읽어 처리하는 것이 어려워지게 된다. 따라서 부가적인 내용에 대해 제거하는 작업이 필요 하다.

첫번째로는 주석과 같이 부가적인 내용들을 제거한다. 파이썬에서 두가지 형태의 주석이 사용된다. 첫번째로는 단일 행 주석이다. #기호를 사용한 후 작성되는 내용들을 주석으로 만드는 방법이다. 해당 주석의 경우 개행이 일어나기 전까지의 내용만 주석으로 처리하게 된다. 두번째는 여러 행의 문자열을 표기하는 방법이지만, 여러 행의 주석으로도 사용되는 방법이다. 주석의 경우 작성 방법이 자유로워 직접 일반 문자열 처리를 통해 제거하기에는 어려움이 있어 정규 표현식을 사용하여 제거작업을 수행 하였다.

두번째로 파이썬에서 함수의 매개변수로 제공하는 내용들은 일반적으로 단일 객체로 전달된다. 하지만 특수한 처리가 두가지 있는데, * 키워드를 사용하여 가변 개수에 제한이 없는 리스트를 매개변수로 사용할 수 있다. 그리고 ** 키워드를 통해 식별자의 이름과 값 형태로 구성되는 딕셔너리 형태를 매개변수로 사용할 수 있다. 위의 경우 문자열의 replace함수를 통해 제거 작업을 수행하였다.

2.2 함수 추출

해당 단계는 2.1단계의 전처리를 통해 특수한 내용들이 제거가 완료된 소스코드들을 대상으로 함수를 추출하는 단계이다. 해당 연구에서는 함수를 추출 하되 클래스에 포함된 함수는 추출하지 않는다. 클래스의 경우 특수한 목적에 맞게 함수가 실행되므로, 이전에 실행된 함수나 정의된 값 들에 대해 알 수 없다. 이는 범용 목적으로 사용되는 전역 함수와는 다를 것으로 판단된다. 따라서 다른 변수들과 함수에 대해 고려해야 할 사항에 대한 처리가 어렵다. 따라서 일반 전역 함수들을 추출하여 실험을 진행한다.

2.3 함수 구성요소 추출

해당 단계는 추출된 함수들을 대상으로 함수의 이름, 매개변수, 식별자에 사용된 단어들을 추출하는 단계이다.

첫번째로 추출된 함수의 첫 줄을 통해 함수의 이름과 매개변수 부분을 추출한다. 함수 선언의 키워드인 'def'를 제거 한 후, 소괄호()을 기준으로 함수의 이름 영역과 매개변수 영역을 분리한다.

두번째로 실행되는 함수의 body부분에서 식별자를 추출한다. 라인 마다 split함수를 통해 '=' 문자를 기준으로 분할을 하여 분할 된 결과의 개수가 2개 이상의 경우, 식별자를 추출을 수행한다. 만약 분할된 결과의 개수가 1개 이하인 경우 단순 함수 호출인 경우이다.

세번째로 매개변수에 초기 설정 값을 정의한 경우 사용된 매개변수의 이름만 추출하는 것이 불가능하기 때문에, 두번째와 마찬가지로 split 함수를 통해 '=' 문자를 기준으로 분할 하여 첫번째로 나오는 문자를 추출한다.

네번째로 추출된 매개변수나 식별자에 대해 타입 지정, 인덱스 값, 클래스 변수에 대한 내용을 제거한다. 해당 내용들에 대해서는 함수 실행 이전에 선언 하였으므로 제거 작업을 수행한다.

다섯번째로 추출된 함수 이름이나, 매개변수, 식별자에 대해 스네이크 케이스, 카멜 케이스[3]로 작성된 내용들을 단어 단위로 분리 하는 작업을 수행한다. split함수를 통해 '_' 문자를 기준으로 스네이크 케이스로 작성된 단어의 분리를 수행하거나, 정규식을 통해 대문자로 시작하여 소문자로 구성된 단어들의 개수가 2개 이상의 경우 카멜 케이스로 작성된 단어의 분리를 수행한다. 분리된 단어들에 대해 strip함수를 통해 공백 제거, lower함수를 통해 소문자로 변경한다.

마지막으로 단어들을 set으로 구성하여 중복을 제거하였고, 학습된 모델에 포함여부 검사를 통해 학습되지 않은 단어의 경우 제거하는 작업을 수행한다.

2.4 비교

해당 단계는 함수에서 추출된 단어들의 집합을 단어 임베딩을 통해 집합 간 유사성 비교를 수행한다. 함수의 이름 단어 집합과, 매개 변수 및 식별자 단어 집합의 집합 간 유사성 비교를 통해 함수의 식별자가 함수의 이름과 유사성이 있는 단어들로 구성 되었는지에 대한 평가를 수행한다.

3. 평 가

해당 파트에서는 제시한 접근법의 평가 기준을 제시하고, 제시한 평가 기준의 결과에 대해 설명을 진행한다.

3.1 실험 설계

본 논문에서는 함수에 사용된 식별자들이 함수의 이름과 연관성 있는 단어로 구성 되었는지에 대해 분석을 수행하고자 한다. 실험에 사용할 첫번째 데이터셋 수집을 위해 깃허브[4]에 등록된 오픈소스들 중 star 개수가 20K가 이상이며, 주로 파이썬으로 작성된 프로젝트들을 수집 하였다.

데이터 셋으로는 The Algorithm[5], youtube-dl[6], tensorflow[7], django[8], flask[9], keras[10], scikit-learn[11], cpython[12], pandas[13], bert[14] 프로젝트를 수집하였다.

해당 데이터셋들의 소스코드들은 많은 사람들의 star을 받았음을 통해 코드 작성 규칙에 의해 잘 작성되었을 것이라 가정하였다.

단어 임베딩의 경우 사전에 학습된 모델을 (로드)하여 사용하였다. 모델의 경우 GoogleNews-vectors-negative300[15]을 사용하였다. Code2Vec[16]를 통해 수행할 수 있었지만, 본 논문에서 수행한 연구 결과를 참고하여 리팩토링을 수행한 소스코드를 처음 접하게 되는 다른 개발자가 최소한의 사전 설명으로도 빠른 이해를 통해 추가적인 작업을 원활하게 수행 할 수 있게 하고자 한다. 따라서 최대한 자연어와 가까운 형태로 함수의 이름이나 식별자를 구성하였음을 평가하고자 일반적으로 사용되는 단어로 학습이 된 GoogleNews-vectors-negative300 모델을 사용하였다.

각 프로젝트에서 추출된 함수를 대상으로 이름 단어 집합과 매개변수 및 식별자 단어 집합의 유사도 평가를 수행하였다. 만약 두 집합 중 평가가 불가능한 단어로만 구성되어 있거나 빈 집합으로 구성되어있는 경우 해당 함수의 단어 간 유사도를 0으로 평가 하였다.

3.2 실험 결과

Table. 1 Average of similarity

프로젝트 명	평균 값	프로젝트 명	평균 값
The Algorithm	0.233	keras	0.314
youtube-dl	0.323	scikit-learn	0.334
tensorflow	0.322	cpython	0.193
django	0.275	pandas	0.270
flask	0.304	bert	0.429

Table. 2 Average of similarity without Set Error

프로젝트 명	평균 값	프로젝트 명	평균 값
The Algorithm	0.264	keras	0.380
youtube-dl	0.355	scikit-learn	0.352
tensorflow	0.356	cpython	0.278
django	0.327	pandas	0.298
flask	0.329	bert	0.429

Table. 1은 첫번째 데이터셋에서 프로젝트별로 추출한 파이썬 함수에서 사용된 단어 집합 간 유사도를 계산하여 평균 값을 나타낸 것이다. Table. 2는 첫번째 데이터 셋에서 평가가 불가능한 항목에 대해 집계를 수행 하지 않은 결과의 평균 값이다.

첫번째 데이터 셋에 구성된 프로젝트들은 소스코드 규칙에 의해 잘 작성되었으며, 적절한 식별자를 사용하였을 것이라 가정하였으나 일부 항목에서 낮은 값을 보이기도 하여 도출한 값에 대해 세밀한 분석을 수행하였다.

전체 결과 중 가장 낮은 결과를 보인 cpython 프로젝트의 소스코드를 통해 세밀하게 분석하고자 한다. 해당 프로젝트에서게 유사도의 평균값이 낮아지게 만드는 다양한 요인을 발견하였다.

요인 중 첫번째로는, main 함수에 식별자가 등록된 경우에 발생하였다. 이 경우는 main이 아닌 프로그램의 이름을 통해 분석을 수행하여야 한다.

두번째는 특정 함수에 너무 많은 단어를 통해 식별자를 구성한 경우이다. 해당 케이스는 한가지 함수에 많은 기능을 포함하려고 하는 경우이다. 한가지 함수에 여러가지 기능을 구성할 경우, 코드를 처음 보는 개발자가 파악하는데 많은 비용을 요구하게 된다.

```
def _write_ushort(f, x):
    f.write(struct.pack(string, x))

function name : ['write'], fail ['', 'ushort']
function content : ['x', 'f'], fail []
similarity value: 0.17444735765457153
```

Fig. 2 Source Code with alphabet parameter

세번째는 Fig. 2. 와 같이 하나의 알파벳을 식별자로 사용하는 경우이다. 해당 케이스는 식별자가 어떤 역할을 하는지에 대해 개발자가 직접 함수를 분석하여야 하며, 함수 이름에 사용된 단어와의 연관성을 분석하기 어렵다.

```
== Item Error ==
name : [], fail: ['abstractmethod']
content : [], fail ['funcobj']
```

Fig. 3 Item Error Example

네번째로는 Fig. 3과 같이 임의로 축약한 단어를 사용한 경우이다. 해당 케이스는 개인의 기준으로 축약이 진행되어, 읽는 사람마다 다르게 해석할 여지가 있다.

다섯번째로는 여러 단어의 조합을 카멜케이스나 스네이크 케이스로 작성하지 않고 사용하는 경우이다. 식별자 작성에는 뛰어 쓰기의 사용이 불가능하다. 따라서 식별자를 모두 소문자로 구성하게 되면 해당 코드를 읽는 개발자들이 직관적인 해석을 하기 어렵게 한다. 또한 단어의 구성에 따라 다르게 해석이 이루어질 수도 있다.

반대로 높은 유사도를 획득한 케이스의 함수를 분석하고자 한다. 가장 높은 유사도 평균값을 나타낸 bert 프로젝트의 소스코드를 통해 분석하였다.

첫번째로 함수 이름에 사용되었던 단어가 그대로 식별자의 구성에도 사용된 경우이다. 해당 케이스는 같은 단어를 사용할 경우 높은 값이 나오는 것에 대해 연구로써 의문을 가질 수 있다. 하지만 함수의 이름을 객체와 동작에 대해 명확히 설명해 둔 것이라면, 함수의 동작 과정 중 해당 내용이 명확하게 나와야 한다.

두번째로는 같은 단어로 구성되지는 않았지만, 모두 특징이 유사한 단어로 구성된 케이스이다.

Table. 1과 Table. 2의 차이를 비교함으로써, 함수에 일반적으로 사용하지 않는 단어를 포함하여 구성하는 경우가 있음을 알 수 있게 되었다. 하지만 해당 케이스를 분석 해 본 결과 실제로 의미 없는 단어를 통해 구성하여 발생하는 경우도 있지만, 일부 개발자들은 이해 가능한 수준의 축약 단어를 사용하여 구성한 경우도 상당 수 검출되었다.

실험 결과 함수 이름에 사용된 단어 집합과 함수 식별자에 사용된 단어 집합의 집합 간 유사도가 특정 값 이하을 나타낼 때, 식별자가 함수의 이름과 관련성 없는 단어로 구성되어 개발자의 직관적인 이해가 어려운 상태임을 알 수 있었다. 따라서 본 논문에서는 함수의 이름에 사용된 단어 집합과 식별자에 사용된 단어 집합의 집합 간 유사도가 낮은 경우에 대해 리팩토링을 수행할 때 참고하여 사용할 수 있도록 제안하고자 한다.

4. 결 론

본 논문에서는 함수의 이름에 사용된 단어 집합과 함수의 내부의 식별자에 사용된 단어 집합들 간의 유사도 평가 방법을 연구하였다. 유사도 평가 결과는 소스 코드의 리팩토링 수행 시 참고 정보로 활용할 수 있을 것이라 생각된다. 본 연구의 실험에서는 총 10개의 프로젝트가 포함된 데이터셋을 사용하였고, 그 실험 결과 유사도의 평균값이 0.2~0.4정도를 보여주었고 범위를 벗어나는 케이스에서는 분석을 통해 유사도를 각각의 특징들을 발견할 수 있었다. 실험 결과의 대부분은 에서 큰 문제 없이 결과를 정상적으로 보여주었지만, 현재 개발자들이 주로 사용하는 약어들 혹은 용어들에 대한 벡터 값이 존재하지 않아 모든 케이스에 대해 유연한 처리가 불가능하였다.

향후 연구에서는 기존 학습된 모델에서 개발자들이 주로 많이 사용하는 단어들에 대해 추가적인 학습을 수행 한 후 실험을 진행하고자 한다. 또한 소스코드를 대상으로 하는 자연어 처리 프로세스를 하는 연구들의 원활한 진행을 위해 프로그래밍에 주로 작성되는 다양한 약어들에 대해 유연한 처리를 할 수 있는 기능 또한 연구하고자 한다.

Acknowledgement

이 성과는 2020년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2020R1I1A3073313) (corresponding author : Yeong-Seok Seo)

Reference

[1] Schankin Anrea, Berger Annika, Holt Daniel V., Hofmeister Johannes C., Riedel Till, Beigl Michael, Descriptive compound identifier names improve source code comprehension. In 2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC) (pp. 31-40), 2018

[2] Word Embedding [Internet], https://www.tensorflow.org/tutorials/text/word_embeddings

[3] Fakhoury, Sarah, Roy, Devjeet, Hassan, Adman, & Arnaoudova, Venera. Improving source code readability: theory and practice. In 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC) (pp. 2-12). IEEE, 2019

[4] GitHub [Internet]. https://www.github.com

[5] The Algorithm [Internet], https://github.com/TheAlgorithms/Python

[6] youtube-dl [Internet], https://github.com/ytdl-org/youtube-dl

[7] tensorflow [Internet], https://github.com/tensorflow/models

[8] django [Internet], https://github.com/django/django

[9] flask [Internet], https://github.com/pallets/flask

[10] keras [Internet], https://github.com/keras-team/keras

[11] sickit-learn [Internet], https://github.com/scikit-learn/scikit-learn

[12] cpython [Internet], https://github.com/python/cpython

[13] pandas [Internet], https://github.com/pandas-dev/pandas

[14] bert [Internet], https://github.com/google-research/bert

[15] GoogleNews-vectors-negative300 [Internet], https://code.google.com/archive/p/word2vec/

[16] Alon Uri. , Zilberstein Meital, Levy Omer & Yahav, Eran. code2vec: Learning distributed representations of code. Proceedings of the ACM on Programming Languages, 2019

비지도 학습 기반 결함 예측 방법들의 성능 비교 연구

권수진O, 남재창
한동대학교 전산전자공학부
21900044@handong.edu, jcnam@handong.edu

A Comparative Study on Unsupervised Learning Based Defect Prediction Approaches

Sujin KwonO, Jaechang Nam
School of Science and Electronic Engineering, Handong Global University

요 약

대부분의 소프트웨어 결함 예측은 지도 학습 기반의 모델로 만들어졌다. 그러나 과거 데이터 집합이 적은 프로젝트에 대한 결함 예측은 지도 학습 기반 예측 모델을 훈련시키기 어려운 문제가 있다. 그렇기 때문에 결함 데이터 없이 결함을 예측하는 비지도 학습 기반의 결함 예측에 대한 연구가 활발하다. 다른 프로젝트 데이터를 사용해 분류기를 만드는 프로젝트 간 결함 예측 (Cross-Project Defect Prediction, CPDP) 을 사용하는 것도 하나의 방법이지만, 이는 소스와 타깃 프로젝트 간의 분포가 다르다는 이질성 문제가 있다. 결함 예측 방법의 문제를 개선하기 위한 연구로 메트릭 값의 크기에 따라 자동으로 결함 데이터 정답을 찾는 새로운 접근법인 CLA와 CLAMI가 있다. 또한 CPDP의 한계를 해결하고자 연결 기반 비지도형 스펙트럼 분류기 (Connectivity-based Unsupervised Classifier, CUC)를 사용한 연구가 있는데 이는 비지도형 모델이기 때문에 훈련 데이터가 없어도 프로젝트에 적용할 수 있다. 이와 같은 비지도형 기반의 결함 예측 방법들에 대한 비교 연구가 없었기 때문에, 본 논문에서 CLA와 CLAMI 그리고 CUC의 성능을 비교한다. 그 결과 CLA는 정밀도가 높고, CUC는 재현율과 F1 값이 높은 것으로 나타났다. 본 연구의 결과를 통해 개발자의 개입이 필요 없는 비지도 학습 기반 두 기법의 제한점을 확인하고, 추후 새로운 비지도 학습 기법 연구를 진행할 계획이다.

1. 서 론

결함 예측은 소프트웨어 품질 개선 활동을 돕는 대표적인 기법 중 하나이다[3]. 결함 예측을 통해 개발자들에게 문제가 있는 부분을 찾아 줌으로써 결함이 밀집한 부분에 집중할 수 있도록 돕는다[4, 5]. 따라서 결함 예측은 개발자들의 효율성을 높이는 데 도움을 줄 수 있다는 점에서 필요한 연구 분야이다.

결함 예측을 위한 다양한 알고리즘과 메트릭(metric)에 대한 연구가 있다[3, 5, 6, 17, 18, 19]. 대부분이 지도 학습을 기반으로 만들어졌는데 지도 학습 기반의 결함 예측에는 한계가 있다[6]. 과거 데이터가 적은 프로젝트는 결함 정보가 부족하기 때문에 결함 예측 모델을 만들기가 어렵다는 것이다. 이 한계를 해결해 보고자 다른 프로젝트를 사용해 예측 모델을 만드는 프로젝트 간 결함 예측 (Cross-Project Defect Prediction, CPDP)에 대한 연구가 있다[7, 8, 9]. 하지만 이는 소스와 타깃 데이터의 분포가 다르다는 문제가 있다.

결함 데이터를 가지고 있지 않은 데이터 집합에 대하여 자동으로 입력 모듈 데이터에 결함이 있는지 답을 찾아 주고, CPDP의 문제를 해결하기 위한 방법으로 CLA와 CLAMI에 대한 연구와 연결 기반 비지도형 분류기 (Connectivity-based Unsupervised Classifier, CUC)에 대한 연구가 있다. CLA, CLAMI는 더 복잡할수록 결함이 있을 가능성이 높다는 가정으로부터 클러스터링, 라벨링 등의 과정을 통해 결함을 예측하고, CUC는 스펙트럼 분류기 (Spectral classifier)를 기반으로 defective entity 내 또는 clean entity 내에서의 연결이 defective와 clean 사이의 연결보다 강하다는 전제를 이용하여 결함을 예측한다[1, 2].

비지도 학습 기반의 결함 예측 방법이 제안되었지만, 이들에 대한 성능 비교 연구는 없다. 따라서 기존 결함 예측의 한계를 해결하고자 연구된 비지도 학습 기반의 결함 예측 방법 세 가지를 실험을 통해 비교 분석하고자 한다.

본 논문의 구성으로는 2장에서 관련 연구를 검토하고 3장에서 본 논문에서 사용하는 CLA, CLAMI 그리고 CUC의 접근법을 각각 설명한다. 4장에서는 실험 과정을 설명하고 5장에서는 실험 결과를 보여준다. 6장에서 본 연구의 의의를 논의하고 7장에서는 논문에 대한 결론을 내린다.

2. 관련 연구

전형적인 결함 예측은 지도 학습을 기반으로 이루어져왔다. 프로젝트 내 결함 예측 (Within-Project Defect Prediction, WPDP)의 과정은 다음과 같다. 먼저 Git과 같은 소프트웨어 아카이브로부터 데이터를 가져온다. 수집한 결함 정보로부터 메트릭과 결함 데이터를 이용해서 인스턴스를 만들어내고 정규화 또는 feature selection과 같은 전처리 과정을 거친다. 이와 같은 단계들을 거치고 나면 예측 모델을 만들기 위한 훈련된 인스턴스 데이터를 갖게 되고 예측 모델이 완성된다. 만들어진 예측 모델에 새로운 인스턴스가 들어오면 분류를 통해 결함이 있는지 없는지의 결과(defective, clean)를 알 수 있다.

그러나 이와 같은 결함 예측에는 한계가 있다[10, 11]. 새로운 프로젝트나 역사적 데이터가 적은 프로젝트는 결함 정보가 부족해서, 모델을 훈련시키기 위한 결함 데이터를 만들 수 없다는 것이다. 즉 결함 예측 모델을 만들기 어렵다. 이 한계점을 해결해 보고자 프로젝트 간 결함 예측 방법(CPDP), 비지도 학습 기반의 클러스터링 방식인 expert-based 방법[20], 그리고 threshold-based 방법[21]과 같은 선행연구가 있었다. 그러나 CPDP는 예측 성능이 좋지 않다는 한계가 있고[10], 그림 1을 통해 볼 수 있듯이 training project와 target project 간에 이질성 문제가 있다. 또한 expert-based 방법은 항상 전문가가 필요하다는 한계가 있고, threshold-based 방법은 메트릭 값을 구하기 위해 추가적인 노력이 필요하다는 한계가 있다[20, 21].

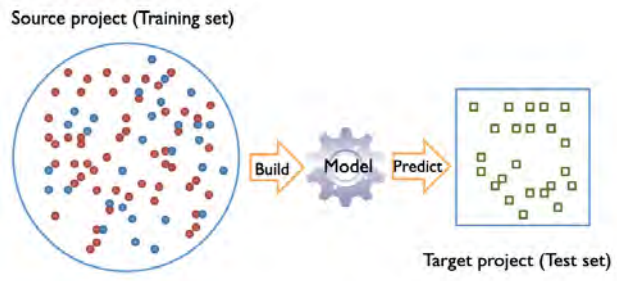


그림 1. Cross-Project Defect Prediction (CPDP) [1]

이러한 한계점을 해결하기 위한 접근법으로 CLA와 CLAMI[1] 그리고 CUC[2] 와 같은 연구가 있다. 두 연구 모두 비지도 학습 기반이라는 점에서 공통점을 갖고 있다. 즉, 결함 데이터를 알지 못하는 상황에도 예측을 할 수 있다. CLA와 CLAMI는 역사적 데이터가 적은 프로젝트의 결함 예측 문제를 해결했고, 결함 데이터가 없는 데이터 집합으로부터 예측 모델을 자동으로 만들 수 있다. CUC는 CPDP의 한계를 해결하고자 했다.

3. 배경지식

3.1. CLA/CLAMI 접근법

CLA와 CLAMI는 메트릭 값의 크기를 이용해서 결함 정보를 모르는 인스턴스에 결함 데이터를 찾아주는 방법이다[1]. CLA는 두 단계, CLAMI는 네 단계로 진행된다. 첫 번째로 클러스터링 (Clustering), 두 번째로 라벨링 (LAbeling) 그리고 CLAMI에서는 추가로 메트릭 선택 (Metric selection)과 인스턴스 선택 (Instance selection) 과정을 거친다. CLA는 모든 인스턴스의 결함 여부를 예측할 수는 있지만 여전히 메트릭과 인스턴스에 결함 경향을 따르지 않는 값이 있을 수 있다. 따라서 메트릭 선택과 인스턴스 선택 과정을 통해 잘못된 예측을 더 최소화하는 방법이 CLAMI이다.

CLA와 CLAMI의 접근법에서 첫 번째 단계인 클러스터링에서는 각 메트릭의 중앙값과 같은 특정 임계 값보다 메트릭 값이 높은 값을 찾는다. 그리고 각 인스턴스마다 메트릭 값이 높은 것의 개수를 세고 개수가 같은 인스턴스끼리 묶는다. 이와 같이 클러스터링을 거친 그룹들을 두 그룹으로 나누고 값이 큰 쪽이 defective, 값이 작은 쪽이 clean이라고 예측하는 것이다. 이 과정이 라벨링이다. 라벨링 과정을 마치면 모든 인스턴스에 대해 결함 예측 결과인 결함 데이터를 얻을 수 있다. 하지만 결함 예측 결과를 따르지 않는 메트릭과 인스턴스가 여전히 있을 수 있다. 따라서 CLAMI에서는 메트릭 선택, 인스턴스 선택을 한다. 메트릭 선택은 MVS (Metric Violation Score)를 기반으로 한다. MVS 값이 가장 작은 메트릭을 선택함으로써 메트릭 선택을 한다. 인스턴스 선택은 메트릭 선택을 하고 난 후에도 결함 예측 결과를 따르지 않는 메트릭 값을 가진 인스턴스를 제거한다. 이와 같은 방식으로 각 인스턴스에 대한 결함 예측이 가능하다.

3.2. CUC 접근법

CUC는 CPDP의 이질성 문제를 해결하기 위해 연구된 비지도형 분류기이다[2]. R로 구현된 CUC의 알고리즘은 입력으로 행이 데이터에 대한 각 인스턴스, 열이 메트릭으로 되어 있는 행렬을 받고, 결과로 모든 인스턴스의 결함 경향 데이터를 벡터로 나타낸다. 행렬이 입력으로 들어오면 먼저 z-score를 사용하여 메트릭을 정규화한다. 가중 인접 행렬(W)을 만들고, 라플라시안 행렬(Lsym)을 계산한다. Lsym에 고윳값 분해를 하고, 두 번째로 가장 작은 고유 벡터(v1)을 선택한다. 0을 사용해서 v1에 이중 분할을 하고 각 클러스터의 결함 데이터를 알아낸다. Defective와 clean 사이의 연결성은 defective entity 내에서 또는 clean entity 내에서 보다 약하기 때문에 같은 그룹에 속해 있으면 연결성이 더 강하다는 점을 이용하였다.

4. 연구 내용 및 방법

CLA, CLAMI, 그리고 CUC의 세 가지 방법에 대해 각각 실험을 하고 성능을 비교하였다.

4.1. CLA/CLAMI

GitHub에 오픈소스로 제공되어 있는 CLAMI 프로젝트를 복제해 와서 실험하였다. CLAMI의 입력 필수 옵션으로는 예측을 할 파일, 클래스 속성의 이름, 그리고 양성 라벨의 값을 받는다. 예측할 파일로 Arff 형식의 데이터 집합 파일을 주었다. 선택 옵션으로 있는 weka 분류기, 컷오프 백분위 수는 따로 지정해 주지 않고 기본 값인 각각 Logistic Regression과 중간 값으로 설정하였다[1].

4.2. CUC

논문에서 제공하는 R로 구현된 것을 이용해 실험한다. 실행을 하기에 앞서 CUC에 class attribute를 제거한 것, 즉 결함 정보가 없는 데이터 집합을 입력으로 주고 예측 결과를 얻어야 한다. CUC는 비지도형 기반인 결함 데이터가 없는 프로젝트를 위해 연구된 결함 예측 방법이기 때문이다. 실제 값과 예측 결과 값의 비교를 통해 성능을 계산할 수 있다. 원본 파일에서 클래스 속성들만 제거하여서 비지도형 기반의 결함 예측을 할 수 있도록 데이터를 준비한다. 즉, 원본 데이터로부터 클래스 속성이 없는 데이터를 생성한다.

R로 제공된 코드는 입력으로 행렬 형식을 받는다. 이때 행은 데이터에 대한 각 인스턴스이고 열은 메트릭으로 되어 있어야 한다. 따라서 CUC 함수에 입력을 주기 전에 R에서 arff 형식으로 되어 있는 데이터 파일을 읽어서 행렬 형식으로 바꾸어야 한다. 먼저 R에서 arff 파일을 읽기 위해 R에서 제공하는 RWeka 패키지를 설치한다. 그림 2의 ②번과 같이 RWeka 라이브러리에 있는 read.arff(…) 함수를 사용하여 R에서 arff 형식의 파일을 읽고, 행렬 형식으로 변환하기 위해 data.matrix(…) 함수를 사용한다. 그림 2의 ③번과 같이 행렬 형식으로 바꾼 데이터를 CUC 함수에 입력으로 넣어주면, 그림 2의 ④번처럼 인스턴스 각각에 대한 결함 정보를 True, False로 나타낸 행렬 형식의 예측 결과를 얻을 수 있다. 결과는 콘솔에 출력되는데 이를 파일에 저장하기 위해 R에서 제공하는 capture.output(…) 함수를 사용하여 결과가 저장된 텍스트 파일을 생성한다.

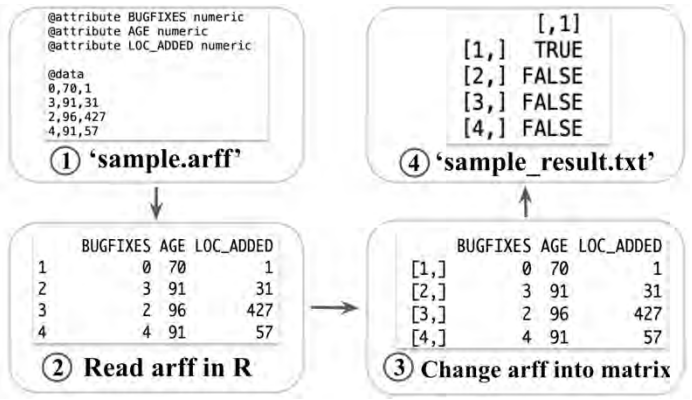


그림 2. CUC 실험 과정

이제 예측 결과와 원래 데이터 파일에 있던 실제 클래스 라벨 값을 비교하고 성능을 계산할 수 있다. 실제로 defective였는데 예측 결과가 True인 경우는 TP(True Positive), 실제로 defective였는데 예측 결과가 False인 경우는 FN(False Negative), 실제로 clean이었는데 예측 결과가 True인 경우는 FP(False Positive), 마지막으로 실제로 clean이었는데 예측 결과가 False인 경우는 TN(True Negative)으로 각각이 몇 개인지 계산해서 confusion matrix의 값을 구한 후 성능을 계산하였다.

4.3. 실험 데이터

세 가지의 비지도 학습 기반의 결함 예측 접근법을 비교하기 위해 표1과 같은 데이터들을 사용하여 실험하였다. AEEEM[13], JIT[14], NASA[15], PROMISE[15] 그리고 Relink[16]의 데이터를 사용하였다. 프로젝트별로 데이터 셋이 가진 인스턴스의 수는 다양하다. AEEEM은 EQ, JDT, ML, PDE와 같은 데이터 셋을 가지고 있고 JIT으로부터는 Bugzilla, columba, jdt, mozilla, platform, postgres를, NASA로부터는 cm1, jm1, kc1, kc2, pc1을 사용했다. PROMISE로부터 ant, camel, ivy, jedit, log4j, lucene, poi, synapse, xalan, xerces 와 같은 dataset으로 실험하였고, 마지막으로 Relink에서는 apache, safe, zxing을 사용했다.

4.4. 측정 방법

성능 측정 방법으로 정밀도(precision), 재현율(recall), F1 값(f-measure)을 사용했다. 정밀도는 defective로 예측된 모든 인스턴스 중에서 올바르게 예측된 것의 인스턴스의 비율을 나타내고, 재현율은 모든 실제 defective 인스턴스 중에서 올바르게 예측된 것의 인스턴스의 비율을 측정한다. F1 값은 정밀도와 재현율의 조화평균으로, 결함 예측 연구에서 성능 측정할 때 자주 사용된다. 또한 Wilcoxon signed rank test($p<0.05$)를 통해 통계적 유의성을 검증하였다.

표1. 실험에 사용된 데이터 집합

dataset	# of instance	dataset	# of instance
EQ	324	ivy v2.0	352
JDT	997	jedit v3.2	272
ML	1862	jedit v4.0	306
PDE	1497	jedit v4.1	312
bugzilla	4620	log4j v1.0	135
columba	4408	log4j v1.1	109
jdt	35339	lucene v2.0	195
mozilla	98228	lucene v2.2	247
platform	64203	lucene v2.4	340
postgres	20384	poi v1.5	237
cm1	498	poi v2.5	385
jm1	10885	poi v3.0	442
kc1	2109	synapse v1.0	157
kc2	522	synapse v1.1	222
pc1	1109	synapse v1.2	256
ant v1.5	293	xalan v2.4	722
ant v1.6	351	xalan v2.5	802
ant v1.7	745	xerces v1.2	440
camel v1.2	608	xerces v1.3	453
camel v1.4	872	apache	194
camel v1.6	965	safe	56
ivy v1.4	241	zxing	399

5. 실험 결과

CLA, CLAMI 그리고 CUC를 통해 실험한 결과 성능은 표2와 같다. CLA/CLAMI의 classifier로 Logistic Regression을 기본 값으로 사용하였다. 각 프로젝트의 성능 값은 프로젝트에 대한 모든 데이터 셋의 성능 결과의 평균을 계산한 것이다.

표2. 세 가지 비지도형 결함 예측 방법의 성능 결과

성능지표	CLA	CLAMI	CUC
precision	**0.438**	0.423	0.425
recall	0.621	0.661	**0.675**
f-measure	0.477	0.475	**0.489**

평균 정밀도는 CLA가 0.438, CLAMI가 0.423, CUC가 0.425로 CLA가 가장 높게 나타났으며, 재현율과 F1 값은 CUC가 각각 0.675와 0.489로 가장 높은 성능을 보였다. 재현율은 결함을 얼마나 빠뜨리지 않고 잘 찾았는지, 정밀도는 얼마나 정확하게 찾았는지 확인할 수 있는 지표이다. 그리고 Wilcoxon signed rank test($p>0.05$)를 통해 통계적 유의성을 검증하기 위해 CUC를 기준으로, CLA와 CLAMI 각각을 검증해보았다. 표 3과 같이 모든 p 값이 0.05 이상인 것으로 결과를 얻었고 따라서 성능 결과에 대해 통계적으로 유의미하지 않다.

표3. Wilcoxon signed rank test의 p-value

p-value	CUC & CLA	CUC & CLAMI
precision	0.081	0.644
recall	0.968	0.408
f-measure	0.679	0.847

CLA, CLAMI는 실험에 사용한 데이터 셋 모두에서 실험 결과를 얻을 수 있었지만, CUC는 규모가 큰 데이터에서 결과를 얻을 수 없었다. 서버의 사양이 물리 CPU가 2개, CPU 당 물리 코어의 수가 8개, 전체 물리 코어 수가 16개인 환경에서 실험을 진행하였는데, 크기가 2.0MB 이상인 데이터는 실험 결과를 얻지 못했다. 표 4에서 볼 수 있듯이 CUC는 실험을 한 총 44개의 데이터 셋 중에서 4개의 데이터 셋이 결과를 얻지 못하였다. 결과를 얻지 못한 데이터는 다른 데이터들과 동일한 환경에서 실험하였지만 메모리 문제 에러와 함께 실행이 강제 종료되었다.

표4. 실험에 성공한 데이터 셋 개수

	CLA	CLAMI	CUC
# of the succeed	44	44	40
# of the failed	0	0	4
Total # of dataset	44	44	44

6. 논의

성능을 비교한 결과, 정밀도는 CLA가 가장 높았다. CLA는 더 복잡할수록 더 결함이 있을 가능성이 높다는 사실을 전제로 만들어진 접근법이다. 따라서 CLA에서 메트릭의 값이 높은 것은 결함이 있는 클러스터로 묶인다. CLA의 정밀도가 높게 나온 것은 메트릭 값이 높을수록 결함이 있는 경향이 있다는 사실을 증명해 준다.

CLAMI의 성능이 다른 접근법에 비해 낮게 나온 이유로는 CLAMI가 메트릭 선택, 인스턴스 선택을 거치며 결함 데이터를 따르지 않는 인스턴스들이 제거되면서 데이터가 많이 남지 않기 때문인 것으로 보인다.

CLA와 CLAMI는 모든 데이터 집합에 대해 결과를 얻을 수 있었지만 CUC는 4개의 데이터에 대해 결과를 얻을 수 없었다. CUC는 대규모의 프로젝트에 대해서는

검증되지 않았었다[2]. CUC는 검증할 때 결함 예측 문헌에 일반적으로 사용된 26개의 프로젝트에 대해 실험이 진행되었는데, 그 데이터들은 본 논문에서 실험한 데이터보다 크기가 작고 대규모의 데이터에 대해서는 실험이 진행되지 않았었다[2]. 따라서 데이터 집합에 인스턴스가 많은 JIT의 프로젝트 중에서 결과를 얻을 수 없는 일이 발생했다. 이는 CUC의 알고리즘이 복잡하게 만들어졌기 때문인 것으로 생각된다. 즉, 시간 복잡도와 공간 복잡도가 커지면서 인스턴스가 많은 데이터가 들어오면 결과를 얻는 데 시간이 오래 걸리는 것이다. 따라서 이는 규모가 큰 프로젝트에 대해서 실제 개발자가 사용하기 어려울 것으로 예상된다.

그럼에도 불구하고 CUC는 재현율과 F1 값에서 가장 높은 값을 보였다. 이는 CUC를 결함 예측에 사용하게 된 가정과 연관이 있을 것으로 생각된다. 즉, CUC는 defective entity들이 서로 비슷한 영역에 있을 것이라 가정하고 있기 때문에, 해당 가정이 맞는 데이터의 경우 좋은 성능을 보이게 된다.

7. 결론

결함 예측 분야에서 비지도 학습 기반의 모델에 대한 연구는 적은 편이다. 그런데 훈련 데이터를 가지고 있지 않은 프로젝트에 대한 결함 예측의 필요성이 증가하면서 비지도형 모델의 결함 예측이 중요해지는 추세이다[12]. 본 논문에서는 CLA, CLAMI 그리고 CUC를 통한 비지도 학습 기반의 결함 예측 방법 세 가지의 성능을 비교해 보았다. 그 결과 정밀도는 CLA가 가장 높고, 재현율과 F1 값은 CUC가 가장 높다는 결과를 얻었으며, CLAMI는 다른 두 접근법보다 성능이 낮은 결과를 보였다. 그리고 훈련 데이터의 인스턴스 개수가 많을 경우에는 CUC의 한계로 인해 실험 결과를 얻지 못했고, 따라서 데이터의 규모가 큰 프로젝트에 대한 결함 예측은 CUC보다 CLA와 CLAMI가 더 적합하다는 결과를 얻었다.

본 논문에서는 CLA와 CLAMI의 cutoff 값을 중간 값으로 설정하고 실험하였는데 이후의 연구에서는 다양한 cutoff 값에 따른 성능 비교를 해볼 필요가 있다. 또한 데이터 집합의 defective와 clean의 분포에 따라 결함 데이터가 다르기 때문에 데이터 집합의 메트릭의 분포에 따른 비교 연구도 필요하다.

※ 본 연구는 과학기술정보통신부와 정보통신기술 진흥센터의 2020년도 소프트웨어 중심대학 지원사업(2017-0-00130)과 2020년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2018R1C1B6001919).

References

[1] 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, 2015, pp. 452-463, doi: 10.1109/ASE.2015.56.

[2] F. Zhang, Q. Zheng, Y. Zou and A. E. Hassan, "Cross-Project Defect Prediction Using a Connectivity-Based Unsupervised Classifier," 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), Austin, TX, 2016, pp. 309-320, doi: 10.1145/2884781.2884839.

[3] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," IEEE Trans. Softw. Eng., vol. 33, pp. 2-13, January 2007.

[4] E. Engstro¨m, P. Runeson, and G. Wikstrand, "An empirical evaluation of regression testing based on fix-cache recommendations," in *Software Testing, Verification and Validation (ICST), 2010 Third International Conference on*, April 2010, pp. 75-78.

[5] N. Ohlsson and H. Alberg, "Predicting fault-prone software modules in telephone switches," *Software Engineering, IEEE Transactions on*, vol. 22, no. 12, pp. 886-894, Dec 1996.

[6] 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, 2005, pp. 284-292.

[7] T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," in *Proceedings of the 30th interna- tional conference on Software engineering*, 2008, pp. 531-540.

[8] F. Rahman and P. Devanbu, "How, and why, process metrics are better," in *Proceedings of the 2013 International Conference on Software Engineering*. Piscataway, NJ, USA: IEEE Press, 2013, pp. 432-441.

[9] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," *Inf. Softw. Technol.*, vol. 54, no. 3, pp. 248-256, Mar. 2012.

[10] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, “Cross-project defect prediction: a large scale experiment on data vs. domain vs. process,” in *Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*. New York, NY, USA: ACM, 2009, pp. 91–100.

[11] J. Nam, S. J. Pan, and S. Kim, “Transfer defect learning,” in *Proceedings of the 2013 International Conference on Software Engineering*. Piscataway, NJ, USA: IEEE Press, 2013, pp. 382–391.

[12] Euy Seok Hong, Mi Kyeong Park. "Software Engineering: Unsupervised Learning Model for Fault Prediction Using Representative Clustering Algorithms". *KIPS Transactions on Software and Data Engineering*, vol. 3, pp.57-64, 2014.

[13] Marco D’Ambros, Michele Lanza, and Romain Robbes. 2012. Evaluating defect prediction approaches: a benchmark and an extensive comparison. Empirical Software Engineering 17, 4 (01 Aug 2012), 531–577. https://doi.org/10.1007/s10664-011-9173-9

[14] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi. 2013. A large-scale empirical study of just-in-time quality assurance. IEEE Transactions on Software Engineering 39, 6 (June 2013), 757–773. https://doi.org/10.1109/TSE.2012.70

[15] Baljinder Ghotra, Shane McIntosh, and Ahmed E. Hassan. 2017. A Large-Scale Study of the Impact of Feature Selection Techniques on Defect Classification Models. *In Proc. of the International Conference on Mining Software Repositories (MSR)*. 146–157.

[16] Rongxin Wu, Hongyu Zhang, Sunghun Kim, and Shing-Chi Cheung. 2011. ReLink: Recovering Links Between Bugs and Changes. *In Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering (ESEC/FSE ’11)*. ACM, New York, NY, USA, 15–25. https://doi.org/10.1145/2025113.2025120

[17] 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, 2010, pp. 59–73.

[18] V. R. Basili, L. C. Briand, and W. L. Melo, “A validation of object- oriented design metrics as quality indicators,” IEEE Trans. Softw. Eng., vol. 22, pp. 751–761, October 1996.

[19] 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, 2011, pp. 4–14. [Online]. Available: http://doi.acm.org/10.1145/2025113.2025119

[20] S. Zhong, T. Khoshgoftaar, and N. Seliya, “Unsupervised learning for expert-based software quality estimation,” in High Assurance Systems Engineering, 2004. Proceedings. Eighth IEEE International Symposium on, March 2004, pp. 149–155.

[21] C. Catal, U. Sevim, and B. Diri, “Clustering and metrics thresholds based software fault prediction of unlabeled program modules,” in Information Technology: New Generations, 2009. ITNG ’09. Sixth International Conference on, April 2009, pp. 199–204.

이상치탐지 모델 학습에서 학습 하이퍼파라미터가 모델의 성능에 미치는 영향 분석

고영진, 한성원
KAIST(한국과학기술원), KAIST(한국과학기술원)
rhdudwns1212@naver.com, lion4151@kaist.ac.kr

Analysis on the effect of hyper-parameters in anomaly detection training

Yeongjin Ko, Sungwon Han
KAIST(Korea Advanced Institute of Science and Technology)

요 약

4차산업혁명에 힘입어 최근 기계학습에 대한 논의가 활발히 진행되고 있으며, 이상치탐지 모델은 침입탐지 및 불량검출 등의 분야에서 활용성이 높을 것으로 예상되고 있다. 본 연구에서는 2020년 기준 SOTA성능을 보여주고 있는 이상치탐지 모델 GOAD를 바탕으로 하이퍼파라미터(배치사이즈, 에포크, 학습률, 데이터변환수, 신경망깊이)를 변화시키고 그에 따른 모델의 성능 변화추이를 관찰 및 t-검정 및 웰쉬 t-검정을 이용해 통계적 유의성을 검정하였다. 검정 결과, 유의 수준 0.05에서, 배치사이즈(0.00), 데이터변환수(0.00) 및 신경망깊이(0.00)은 모델의 성능에 영향을 미침을 확인할 수 있었지만, 에포크(0.15), 학습률(0.36)에 대해서는 모델의 성능에 큰 차이가 없음을 확인할 수 있었다.

1. 서 론

이상치탐지 모델은, 기계학습 모델, 그 중에서도 분류학습의 한 종류로서, 대량의 데이터를 이용해 정상적이지 않은 데이터를 추출하는 것을 목표로 하는 분야다. 분류학습 모델은 데이터셋, 신경망, 비용함수 등 다양한 구성요소로 이루어져 있는데, 일반적으로 모델의 신경망을 구성하고 있는 하이퍼파라미터들에 의해 모델의 성능이 큰 영향을 받는다고 알려져 있다[1][2][3]. 그러나, 이상치탐지 모델은 분류학습 모델과 그 목적과 기능이 다른 만큼, 본 연구에서는 2020년 기준 SOTA 성능을 보여주고 있는 이상치탐지 모델인 GOAD를 바탕으로 실제로 신경망을 구성하고 있는 하이퍼파라미터가 이상치탐지 모델에 미치는 영향을 t-검정을 이용해서 분석하고 그 결과에 대한 해석을 시도하였다.

2. 이상치탐지

이상치탐지(Anomaly Detection)란, 정상(normal) 샘플과 이상치(abnormal)샘플을 구별해내는 문제를 의미한다. 이상치탐지 모델은 일반적으로 샘플에 라벨(label)이 존재하는 지 여부[4]와, 이상치탐지 접근 방법[5]등에 따라 분류한다. 여기서는 간단히 라벨의 여부 및 이상치탐지 접근 방법에 따른 모델명을 소개한다.

2.1. 샘플의 라벨 유무에 따른 이상치탐지 모델의 분류

이상치탐지 모델은 샘플의 라벨 유무에 따라 '지도학습', '반지도학습', '비지도학습'으로 분류될 수 있다.

- 지도학습 이상치탐지(Supervised Anomaly Detection)
  주어진 학습 데이터 셋의 정상 샘플과 비정상 샘플에 라벨이 주어져 있는 경우이다. 지도학습 이상치 탐지 방법의 경우 비정상 샘플을 다양하게 보유하고 있을수록 성능이 더 높아지는 특징이 있다. 또한, 다른 방법들에 비해 보다 높은 정확도를 가지고 있다. 하지만 일반적으로 비정상샘플의 수가 현저하게 적으므로 클래스 불균형(Class-Imbalance) 문제를 자주 겪는 단점이 있다.
- 반지도학습 이상치탐지(Semi-supervised Anomaly Detection)
  주어진 학습데이터 셋 중 일부만 라벨이 주어진 상태에서 모델을 학습시키는 경우이다. 모든 데이터를 라벨링 하는 것은 비용이 많이 들기 때문에 약간의 라벨링을 통해 모델의 성능을 비지도학습보다 끌어올리려는 것이 이 방법의 목적이다. 지도학습과 마찬가지로 클래스 불균형 문제를 겪으며, 따라서 라벨링 된 비정상 샘플(이상치)의 수가 더 많을수록 성능이 높아지는 특징이 있다. 반지도학습 모델로는, Deep SAD[6] 등의 모델을 예로 들 수 있다.
- 비지도학습 이상치탐지(Unsupervised Anomaly Detection)
  지도학습 및 반지도학습 이상치탐지 방식은 학습에 앞서 라벨링 과정이 필요하다. 라벨링 과정은 비용이 들 뿐만 아니라, 한 번도 본적이 없는 아웃라이어에 대해서는 라벨링이 어렵다는 단점이 있다. 비지도학습 이상치탐지 방법은 대부분의 데

이터가 정상 샘플이라는 가정하에 라벨 취득 없이 학습을 시키는 모델로, 별도의 라벨링 없이 어느 정도 성능을 낼 수 있다는 점에서 장점이 있으나 지도학습 및 반지도학습 모델에 비해서 성능이 떨어진다는 단점이 있다. 가장 대표적인 반지도학습 이상치탐지 기법으로는 One-Class SVM[7]과 이를 확장한 Deep SVDD[8]가 있다.

2.2. 이상치탐지 접근 방법에 따른 이상치탐지 모델의 분류

다음은 이상치탐지 접근 방법에 따른 이상치탐지 모델 분류이다. '재구축 모델', '분포 모델', '분류 모델'를 기준으로 이상치탐지 모델의 유형을 구분하였다.

- 재구축 모델(Reconstruction Methods)
  보편적으로 사용되고 있는 이상치탐지 방법론 중의 하나이다. 이 방법은, 정상 샘플들을 낮은 오차율로 재구축할 수 있게 학습한 모델이 비정상 샘플(anomaly sample)을 만나게 될 경우 모델의 재구축 오차율이 클것이다는 발상에 기인한다. 현재까지 K-means, low-rank PCA, KNN(K-Nearest Neighbor)방법, 2018년에는 GAN[9]을 이용한 방법 등이 제안되었다.
- 분포 모델(Distributional Methods)
  이상치탐지를 위해서 자주 사용되는 방법 중에는 분포 모델을 이용한 방식도 있다. 분포 모델에서의 주요한 발상은, 비정상 샘플은 정상 샘플의 분포 상에서 나타날 확률이 굉장히 낮을 것이라는 데 있다. 가우시안 분포, 가우시안 혼합 모델 등이 초반에 제시되었으나, 이 모델들은 데이터가 가우시안 분포 및 가우시안 혼합 분포를 따를 때만 작동하는 단점이 있었다. 이외에 커널밀도추정(Kernel Density Estimate), DAGMM[10] 등이 제시 되었다.
- 분류 모델(Classification Methods)
  분류 모델은 정상 데이터가 분포하는 구역과 비정상 데이터가 분포하는 구역을 분리함으로써 이상치 데이터를 분류하는 모델이다. 예를 들어, One-Class SVM을 분류 기반모델로 분류할 수 있다. 분류 기반 모델에서는 특징 맵을 학습하기 위해 커널을 이용한 방식과 딥러닝모델 등의 방식을 사용하는 편이다.

2.3. 학습 하이퍼파라미터가 분류학습 모델의 성능에 끼치는 영향

이상치탐지 모델은 분류학습 모델(Classification Model), 더 크게 본다면 기계학습 모델(Machine Learning Model)의 일종이다. 기계학습 모델은 학습에 영향을 주는 다양한 하이퍼파라미터들을 지니고 있는데, 분류학습 모델의 경우에는 보통 배치사이즈(batch size), 에포크(epoch), 학습률(learning rate)등의 인자들을 하이퍼파라미터로 가진다. 분류학습 모델을 구성하는 각 인자들의 정의 및 인자들의 변화에 따른 분류학습 모델에 미치는 영향은 다음과 같다[1][2][3].

- 배치사이즈(Batch size)
  배치사이즈란 신경망의 파라미터를 업데이트하기 위해서 신경망에 입력하는 1회당 샘플의 수이다. 배치사이즈가 작으면, 파라미터를 업데이트하기 위한 이터레이션 당 계산비용이 낮아진다. 학습 중 안정도(stability)는 높아져 넓은 범위의 학습률로 학습이 가능하며, 경사(gradient)값에 잡음(noise)이 많이 들어가 모델의 일반화성능(generalization performance)이 좋아지는 경향 도 있다. 다만, 잡음이 너무 많이 섞이면 학습이 느려진다. 반대로, 배치사이즈가 크면, 파라미터를 업데이트하기 위한 이터레이션 당 계산비용이 높아진다. 작은 배치사이즈에 비해 경사에 영향을 주는 잡음은 서로 상쇄되어 모델의 일반화성능은 떨어지나, 경사값은 평행(parallel)하게 학습을 진행할 수 있으므로 다중그래픽카드(multi-GPU)를 이용한다면 학습시간을 단축시킬 수 있다. 일반적으로 배치사이즈를 결정하는 데에 정해진 규칙(rule of thumbs)은 없으나, 본 연구에서 채용하고 있는 GOAD 모델에서는 64사이즈의 배치를 사용한다.

- 에포크(Epoch)
  에포크란 인공신경망에서 전체 데이터를 모델에 넣고 순전파(forward pass) 및 역전파(backward pass)과정을 거친 횟수를 의미한다. 즉, 전체 데이터를 한 번 훑으면서 학습을 시킨다면 해당 모델의 에포크의 수는 1이라고 얘기한다. 또 다른 예로, 전체 데이터 단위로 50번을 학습시킨다면 에포크의 숫자는 50이다. 에포크의 숫자가 작은 경우에는 모델이 데이터를 충분히 학습하지 못해 과소적합(underfitting)의 위험이 있고, 반대로 에포크의 숫자가 너무 클 경우에는 모델이 데이터를 너무 많이 학습해 해당 트레이닝 세트에 과적합(overfitting)될 위험이 있다.

- 학습률(Learning rate)
  학습률이란 비용함수(cost function)을 최소화하기 위한 방법으로 경사하강법(gradient descent algorithm)을 이용할 때, 얼만큼 기울기가 줄어드는 지점으로 이동하는 지를 결정하는 수치이다. 즉, 파라미터의 증감크기(step size)를 결정하는 하이퍼파라미터로, 학습률이 너무 크면 오버슈팅(overshooting)하는 결과가 발생할 수 있다. 학습률이 너무 낮게 책정된 경우에는 모델이 극소값

(local minimum)에 수렴하는데 시간이 오래 걸리는 단점이 있다.

이상치탐지 모델은 분류학습 모델의 일종이지만 그 모델의 목적과 기능에 있어 분류학습 모델과 차이가 있다[11]. 분류학습모델은 그룹 간의 범주를 구분할 수 있는 경계면을 찾는 것을 목적으로 데이터를 분리하는 기능을 수행한다. 반면 이상치탐지 모델은 '정상 데이터'를 정의하고 이를 기반으로 이상 데이터를 찾아내는 것을 목적으로 '정상'데이터의 구역을 구분 짓는 기능을 수행한다(그림 1. 이항분류와 이상치 분류의 차이).
분류학습 모델의 영향을 끼치는 하이퍼파라미터들로는 위에서 살펴본 배치사이즈, 에포크, 학습률 등이 있지만, 이상치탐지 모델과 분류학습 모델간의 목적과 기능에 차이가 있는 만큼 이상치탐지 모델에서도 해당 하이퍼파라미터들이 실제로 모델의 성능에 영향을 미치는지를 확인해 볼 필요가 있다. 본 연구에서는 이상치탐지 모델에 대한 전반적인 이해를 넓히기 위한 방안으로 이상치탐지모델인 GOAD에서 신경망 하이퍼파라미터가 모델의 성능에 미치는 영향 분석을 시도하였다.

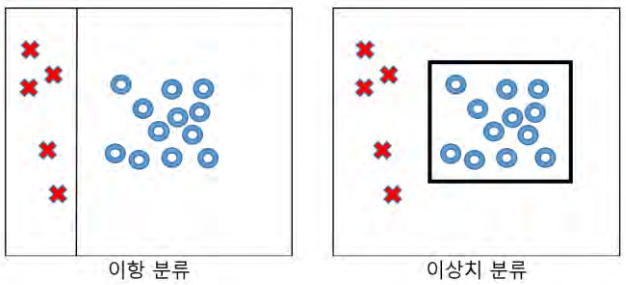


그림 1. 이항분류와 이상치분류의 차이

3. 하이퍼파라미터값 변화에 따른 GOAD 모델 성능 실험

GOAD는 2020년도 ICLR에 실린 "Classification Based Anomaly Detection for General Data"논문[5] 속 모델이다. 기존 분류기반 이상치 탐색 연구로는 Deep SVDD 방식과 GEOM(Geometric-transformation Classification)방식이 대표적인데 이 둘의 핵심 아이디어를 합쳐서 일반화 성능을 높인 모델이 바로 GOAD이다. 기존 Deep SVDD 방식과 GEOM방식은 트레이닝 세트에 포함된 샘플에는 잘 작동하였으나, 일반화 성능은 크게 떨어지는 문제를 가지고 있었다. GOAD는 이를 극복하기 위해 오픈셋 분류 (open-set classification)에서 활용되는 방법을 이상치탐지에 접목시키려는 노력을 하였고 결과적으로 Deep SVDD와 GEOM의 핵심 아이디어를 활용한 GOAD가 Liron Bergman과 Yedid Hoshen에 의해 제안되었다. 하이퍼파라미터의 변화에 따른 이상치탐지 모델의 성능 변화를 분석하기 위해 2020년 기준으로 이상치탐지 분야에서 SOTA(State Of The Art) 성능을 보여주는 GOAD 모델을 실험모델로 선택하였다. GOAD는 2.1절의 기준에 따라 분류하면, 반지도학습 모델로 분류해볼 수 있다. 또한, 2.2절의 기준에 따라 유형을 나누어 본다면, 분포 모델과 분류 모델의 혼합방식의 접근 방식을 지닌 모델이다. 본 연구의 실험에서는 학습 하이퍼파라미터의 변화에 따른 GOAD 모델의 성능을 검정하기 위해서 t-검정 및 웰쉬 t-검정 그리고 레빈 검정 및 바틀렛 검정 방식을 사용한다. t-검정 및 웰쉬 t-검정은 두 집단의 평균을 비교할 때 자주 쓰이며[12], 레빈 검정 및 바틀렛 검정은 동분산성을 검정하는 데 유용한 검정 방법이다[13].

3.1. t-검정 & 웰쉬 t-검정

t-검정(t-test / Two-Sample Test / Student`s T Test)이란 두 집단의 평균을 비교하는 통계적 검정 방법이다. 단순히 차이의 존재 여부를 떠나 두 집단의 비교가 통계적으로 유의미한가를 검정한다. 즉, 이 두 집단의 모집단의 차이가 우연에 의해서 인지 아닌지를 (= 같은 모집단에서 나왔는지 아닌지를) 검정하는 테스트이다[14]. t-검정을 하기 위해서는 먼저 두 가지 가정을 체크해야 한다. 이 두 가지 가정에 따라 그에 맞는 t-검정을 해야만 검정결과에 따른 통계분석결과에 대한 타당성이 있다고 얘기할 수 있기 때문이다.

우선, t-검정을 하기 위해서는 두 집단의 분포가 정규분포를 따르는 지를 점검해야 한다. 이를 정규성 가정이라 부르며, 본 연구에서는 중심극한정리를 이용하여 정규성 가정을 만족시키기 위해 표본의 수(=모델의 학습 횟수)를 30으로 설정하고 실험을 진행하였다. 다음으로는 등분산성 가정이다. t-검정은 등분산성 여부에 따라 기본 t-검정 방식과 웰쉬 t-검정 방식으로 나누어 수행해야 한다. 본 연구에서는 등분산성 가정을 테스트 하기 위해 4.2. 절에서 소개할 레빈 검정(Levene test) 및 바틀렛 테스트(Bartlett test)를 이용하였다.

정리하자면, t-검정의 종류는 2가지로, 정규성 과정을 만족한 상태에서 두 집단의 등분산성 여부에 따라 기본 t-검정 방식 및 웰쉬 t-검정 방식으로 나누어 두 집단의 평균의 차이에 통계적 유의성이 있는 지를 알아볼 수 있다. 기본 t-검정은 두 집단이 모두 정규분포를 따르고 분산이 같을 때 두 집단의 모분포가 같은 지를 판단하는 테스트이다. 반면, 웰쉬 t-검정은 두 집단이 모두 정규분포를 따르지만 분산이 다를 때 두 집단의 모분포가 같은 지를 판단하는 테스트라고 볼 수 있다.

3.2. 레빈 검정 & 바틀렛 검정

레빈 검정은 동질성 검정 방법 중 하나로, 두 개 이상의 그룹에 대해서 분산이 동일한 가를 검정하는 방법이다. 분산의 동질성 테스트, 짧게는, 동분산성(homoskedasticity)테스트 라고도 불린다. 일부 일반적인 통계 절차에서는 표본을 추출한 모집단의 분산이 같다고 가정하는데, 레빈 테스트는 이러한 가정을 평가하는 테스트

이다. 즉, 모집단의 분산이 같다는 귀무가설을 평가하는 테스트라고 이해할 수 있다. 레빈 검정의 결과 p값(p-value)이 일부 유의수준(일반적으로 0.05)보다 작은 경우 표본 분산에서 얻은 차이는 분산이 동일한 모집단의 무작위 표본 추출에서 발생했을 가능성이 낮다고 보고 모집단의 분산이 같다는 귀무가설을 기각하여 모집단의 분산간에 차이가 있다는 결론을 내린다 [15].
바틀렛 검정은 동질성 검정 방법 중 하나로, 모집단으로부터 추출한 표본의 상관계수 행렬의 행렬식 값을 계산하여 상관계수 행렬이 단위행렬인지 아닌지 카이제곱분포를 이용해서 검정하는 방법이다. 바틀렛 검정 또한 레빈 검정과 마찬가지로 비교하려는 집단 간의 모집단의 분산이 같은 지를 귀무가설로 책정하고 이를 검정하는 테스트이다. 바틀렛 검정은 자료가 정규 분포를 이룬다는 확신이 있을 때 사용하면 매우 정확한 결과를 도출하지만 자료가 정규분포를 이루지 않을 경우 정확도가 낮다는 단점이 있다. 즉, 바틀렛 검정은 자료의 정규성에 따른 검정력이 레빈 검정보다 더 민감하다[16].

3.3. GOAD 모델 속 변경할 하이퍼파라미터 선정

GOAD 모델은 총 10개의 파일(표 1. 모델 구현 파일목록)로 구성되어 있다. 이 중 모델의 학습을 조절하는 하이퍼파라미터는 7번 train_ad.py, 8번 train_ad_tabular.py에 위치해있다. 7번 train_ad.py는 이미지데이터를 학습하기 위한 모델이고, 8번 train_ad_tabular.py는 도표자료(tabular data)를 학습하기 위한 모델이다.

표 1. GOAD 모델 구현 파일목록

GOAD 구현 파일 목록	
1. README.md	2. requirements.txt
3. data_loader.py	4. transformations.py
5. opt_tc.py	6. opt_tc_tabluar.py
7. train_ad.py	8. train_ad_tabular.py
9. wideresnet.py	10. fcnet.py

실험에서는, 도표자료에 대한 학습을 진행하는 8번 파일을 실험모델로 정하고, 8번 파일 속 하이퍼파라미터(그림 2. 하이퍼파라미터 목록)들과 8번 파일이 모델 학습을 위해 사용하는 신경망(그림 3. fcnet 신경망)인 10번 fcnet.py 파일 속 신경망 깊이를 변경해가며 도표자료(arrhythmia)에 대한 GOAD모델의 이상치탐지 성능 변화를 분석하고 성능이 통계적으로 유의미한 차이가 있는 지를 t-검정과 웰쉬 t-검정을 통해 판단해볼 것이다. t-검정과 웰쉬 t-검정 중 어떤 테스트를 사용하여 판단할 지는 레빈 검정과 바틀렛 검정을 통해 결정한다.

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--lr', default=0.001, type=float)
    parser.add_argument('--n_rots', default=32, type=int)
    parser.add_argument('--batch_size', default=64, type=int)
    parser.add_argument('--n_epoch', default=25, type=int)
    parser.add_argument('--d_out', default=4, type=int)
    parser.add_argument('--dataset', default='thyroid', type=str)
    parser.add_argument('--exp', default='affine', type=str)
    parser.add_argument('--c_pr', default=0, type=int)
    parser.add_argument('--true_label', default=1, type=int)
    parser.add_argument('--ndf', default=8, type=int)
    parser.add_argument('--m', default=1, type=float)
    parser.add_argument('--lmbda', default=0.1, type=float)
    parser.add_argument('--eps', default=0, type=float)
    parser.add_argument('--n_iters', default=500, type=int)
```

그림 2. 하이퍼파라미터 목록

```
class netC1(nn.Module):
    def __init__(self, d, ndf, nc):
        super(netC1, self).__init__()
        self.trunk = nn.Sequential(
        nn.Conv1d(d, ndf, kernel_size=1, bias=False),
        )
        self.head = nn.Sequential(
        nn.LeakyReLU(0.2, inplace=True),
        nn.Conv1d(ndf, nc, kernel_size=1, bias=True),
        )
```

그림 3. fcnet(netC1) 신경망

하이퍼파라미터의 변경에 따른 GOAD의 성능변화를 관찰하기 위해 변경해볼 하이퍼파라미터는 배치사이즈, 에포크, 학습률, 데이터변환수, 그리고 신경망의 깊이로 총 5개이다. 배치사이즈, 에포크, 학습률은 이상치탐지 모델 외에 일반적인 신경망에도 존재하는 하이퍼파라미터로서, 2.3.절에서 살펴봤듯이, 신경망의 성능에 큰 영향을 미친다고 알려져 있는 요소들이다. 본 연구에서는 이 3개의 하이퍼파라미터에 변화를 주어 일반적인 신경망이 아닌, 이상치탐지 모델의 경우에서도 해당 하이퍼파라미터가 모델의 성능에 영향을 미치는 지 검정해볼 것이다. 또한, 데이터변환수(ndf)는 GOAD 모델 속에서 데이터증강을 위해서 사용되는 하이퍼파라미터로서, 일반적으로 데이터증강은 트레이닝 세트에 대한 과적합을 막고 모델의 성능을 끌어올리기 위해 이용되는 만큼, 데이터변환도 변경해볼 하이퍼파라미터로 선정하였다. 마지막으로, 신경망의 깊이 또한 모델의 성능에 큰 영향을 끼치는 요소로 알려져 있는 바, 신경망의 깊이까지 총 5개의 하이퍼파라미터에 따른 모델의 성능변화를 t-검증을 통해 검증해볼 것이다.

3.4. t-검정 및 웰쉬 t- 검정 실험

하이퍼파라미터(= 배치사이즈, 에포크, 학습률, 데이터

변환수, 신경망의 깊이) 변화에 따른 모델의 성능변화를 t-검증을 통해 분석하기 위해 하나의 하이퍼파라미터만을 변화시킨 후, 다른 파라미터는 모두 동일한 값을 유지한 채, 30회의 반복시행을 하여 모델의 성능 지표인 f1-score 30개를 구하였다. 아래의 결과값들은 이렇게 해서 구한 모델의 f1-score 30개와 기존에 논문에서 사용한 하이퍼파라미터값(batch size: 64, num of epoch: 1, learning rate: 0.001, ndf: 8, depth: 1)들을 이용하여 30회의 반복시행을 통해 얻은 30개의 f1-score를 비교하여 얻어진 값들이다. t-test를 통해 검증하고자 하는 귀무가설 및 대립가설은 다음과 같다. (t-test가 μ1 = μ2 인지를 검증하는 방법인 만큼, 원래는 통념적으로 옳다고 여겨지는 μ1 != μ2를 귀무가설로 설정해야 하나, 여기에서는 반대로 대립가설 로 설정하였다.)

- H0(귀무가설): μ1 = μ2

이상치탐지 모델의 신경망을 학습시킬 때, batch size(or epoch, learning rate, ndf, depth)값은 모델의 성능(f1-score 분포)을 바꿀 만큼 모델에 영향을 미치지 못한다.

- H1(대립가설): μ1 != μ2

이상치탐지 모델의 신경망을 학습시킬 때, batch size(or epoch, learning rate, ndf, depth)값은 모델의 성능(f1-score 분포)을 바꿀 만큼 모델에 영향을 미친다.

(μ1, = 기본 하이퍼파라미터 값(batch size, epoch, learning rate, ndf, depth)을 가지고 학습한 모델의 평균 f1-score)

(μ2, = 기본 하이퍼파라미터 값(batch size, epoch, learning rate, ndf, depth)중 하나를 변경시킨 후 학습한 모델의 평균 f1-score)

t-검증 적용을 앞서서 살펴보아야 할 것은 두 집단 간 등분산성의 여부이다. 유의수준 0.05를 따르는 레빈 검정과 바틀렛 검정을 이용해서 하이퍼파라미터 변화에 따른 집단 간 등분산성을 검정한 결과 그림 4의 결과를 얻을 수 있었다. 배치사이즈와 신경망깊이를 변화시킨 경우에 대해서는 p-value가 유의수준(0.05)보 다 낮아, 두 집단 간 분산이 다르다는 이분산성(Heteroskedasticity) 결과를 얻을 수 있었고, 에포크, 학습률, 데이터변환수를 변화시킨 경우에 대해서는 p-value가 유의수준(0.05)보다 높아, 두 집단 간 분산이 동분산성을 따르는 모습을 보였다.

등분산성 검정 결과 배치사이즈와 신경망깊이를 변화시킨 모델에 대해서는 이분산성을 띄는 것이 확인되었으므로 모델의 성능 차이를 검정하기 위해 웰쉬 t-검정 기법을 사용하였다. 그리고 등분산성 검정 결과 에포크, 학습률, 데이터변환수를 변화시킨 모델은 동분산성을 띄는 모습이 확인 되었으므로 해당 모델과 기본 모델과의 성능 차이를 검정하기 위해서 기본 t-검정을 시도하여 하이퍼파라미터 변화에 따른 모델의 성능차이를 검정하였다. 기본 t-검정 결과 및 웰쉬 t-검정 결과는 표 2.에 정리하였다.

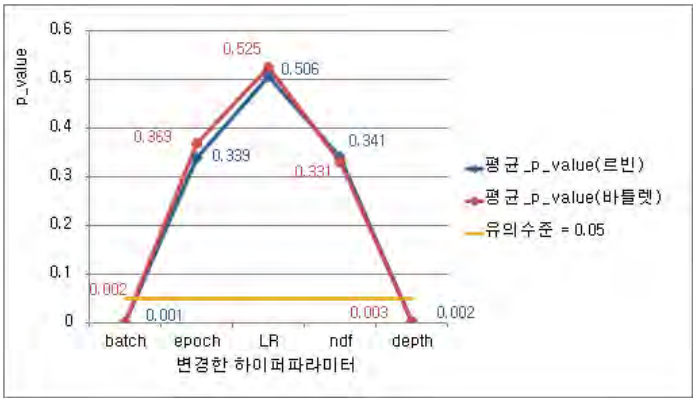


그림 4. 집단 간 분산성 검증 결과(변경한 하이퍼파라미터에 따른 테스트별 평균 p-value, 유의수준 = 0.05)

표 2.의 결과에서 주목할 만한 점은 우선 에포크 및 학습률의 변화에 따른 p-value이다. 유의수준 0.05 수준에서 t-검정결과 p-value가 0.05보다 큰 에포크 및 학습률의 변화에 대해서는 귀무가설(하이퍼파라미터의 변화가 모델의 성능에 영향을 미치지 못한다.)을 기각하지 못했다는 점이다. 분류학습에 있어서 에포크 횟수와 학습률의 변화가 결과에 일정 부분 영향을 미친다는 일반적인 인식과 대조되는 결과이다. 배치사이즈, 데이터변환수, 신경망깊이의 변화에 대해서는 유의수준 0.05 수준에서 t-검정결과 대부분의 하이퍼파라미터 설정값에서 귀무가설을 기각하고 대립가설(하이퍼파라미터의 변화가 모델의 성능에 영향을 미친다.)을 지지하는 결과를 보였다[부록 ㄱ.].

기본 t-검정 결과 및 웰쉬 t-검정 결과의 p-value뿐만 아니라 t-value도 주목할 만하다. 배치사이즈를 기존 64에서 4배인 256으로 높였을 때의 t-value 값은 24.378이고, 데이터변환수를 기존 8에서 4배인 32로 늘렸을 때의 t-value 값은 19.192지만, 신경망깊이를 기존 1에서 4배인 4로 늘렸을 때의 t-value값은 82.344로, t-value값이 훨씬 큰 것을 볼 수 있다. 신경망깊이를 변화시켰을 때의 t-value값이 약 4배 크니까 신경망의 깊이가 배치사이즈나 데이터변환수보다 4배 중요하다고 단순하게 얘기할 수는 없지만, 최소한 모델의 성능이 배치사이즈나 데이터변환수보다 신경망의 깊이에 더 민감하다는 것은 명확하다.

다음으로는 모델의 학습 안정성의 측면이다. 신경망 학습에 쓰이는 하이퍼파라미터를 같은 비율로 변경시켰음에도, 배치사이즈와 신경망의 깊이를 변경시킨 경우에만 집단의 분산이 이분산성를 띄었고, 에포크, 학습률, 데이터변환수를 변경시킨 경우에는 집단의 분산이 동분산성를 띄는 모습을 레빈 검정 및 바틀렛 검정 결과 볼 수 있었다.

표 2. 기본 t-검정 결과 및 웰쉬 t-검정 결과

변경한 하이퍼파라미터	기본 t-검정 (동분산성) (p-value / t-value)	웰쉬 t-검정 (이분산성) (p-value / t-value)
배치사이즈 (batch size) (기본값 64)	등분산성 검증 결과: 이분산성	
16		0.000 / 7.070
32		0.124 / 1.568
128		0.000 / 14.515
256		0.000 / 24.378
에포크(epoch) (기본값 1)	등분산성 검증 결과: 동분산성	
2	0.673 / -0.425	
5	0.229 / 1.217	
10	0.638 / 0.472	
20	0.151 / 1.459	
학습률 (learning rate) (기본값 0.001)	등분산성 검증 결과: 동분산성	
0.1	0.670 / -0.429	
0.01	0.451 / -0.759	
0.0001	0.796 / -0.259	
0.00001	0.368 / -0.907	
데이터변환수(ndf) (기본값 8)	등분산성 검증 결과: 동분산성	
2	0.108 / 1.633	
4	0.300 / -1.047	
16	0.000 / 8.516	
32	0.000 / 19.192	
신경망깊이(depth) (기본값 1)	등분산성 검증 결과: 이분산성	
2		1.000 / 0.000
4		0.000 / 82.344
8		0.000 / 290.15
16		0.000 / 403.90

즉, 모델의 학습 안정성의 측면에서는 배치사이즈와 신경망의 깊이가 다른 파라미터보다 큰 영향을 미치는 것을 알 수 있었다. 이상치탐지 모델이 학습 때 마다 성능 편차가 심하다면 배치사이즈와 신경망 깊이를 조정해보는 것이 좋은 선택이 될 수 있을 것이다.

마지막으로는 기본 t-검정 결과와 웰쉬 t-검정 결과와의 비교이다. 본 연구에서는 실험 집단(하이퍼파라미터 값을 변화시켜 학습한 모델)이 대조 집단(기본 하이퍼파라미터값을 통해 학습한 모델)과 비교하여 두 집단의 분산이 동분산성을 띌 때와 이분산성을 띌 때를 각각 나누어 각각 기본 t-검정과 웰쉬 t-검정을 수행하였다[부록 ㄴ.]. 그러나 실제로 두 개의 검정방법을 통한 p-value 및 t-value 값을 비교한 결과, 아래의 표3.의 모습과 같이 거의 동일한 것을 알 수 있었다. 이와 같은 모습은 분산의 크기가 평균보다 비율적으로 상당히 작았기 때문으로 보인다.

예를 들어, 배치사이즈 32에 대한 기본 t-검정과 웰쉬 t-검정의 경우, 실험집단은 평균 51.98, 분산 0.06의 통계값을 가지는 반면 대조 집단은 평균 52.02, 분산 0.12의 통계값을 가지는 데이터셋이었다. 이 경우 분산은 무려 2배가 차이나서 두 데이터셋은 이분산성을 띄었지만 평균값인 약 52에 비하면 실험집단과 대조집단의 분산 값은 0.06 및 0.12로 모두 상대적으로 그 크기가 작기 때문에 동분산가정하에서의 기본 t-검정과 웰쉬 t-검정 사이의 p-value 및 t-value 차이가 거의 나지 않았다. 따라서 t-검정을 적용하려는 두 집단이 평균에 비해서 분산의 값이 상대적으로 많이 작을 경우, 등분산성 테스트는 생략하고 기본 t-검정을 할지라도, 거의 오차가 없는 t-검정 값을 얻을 수 있을 것이다.

표 3. 배치사이즈 변경에 따른 p-value/t-value값

배치사이즈	기본 t-검정 (p-val / t-val)	웰쉬 t-검정 (p-val / t-val)
16	0.000 / 7.070	0.000 / 7.070
32	0.122 / 1.568	0.124 / 1.568
128	0.000 / 14.515	0.000 / 14.515
256	0.000 / 24.378	0.000 / 24.378

4. 결 론

이상치탐지 모델의 하이퍼파라미터 변화에 따른 성능 변화를 분석하기 위해 배치사이즈, 에포크, 학습률, 데이터변환수, 신경망깊이를 다르게 지정한 후, 그에 따른 GOAD모델의 성능 변화를 t-검정 및 웰쉬 t-검정을 통해 검정하였다. t-검정과 웰쉬 t-검정은 실험집단의 등분산성 여부에 따라 경우에 맞게 적용하여 분석하였다. t-검정 및 웰쉬 t-검정 결과 배치사이즈, 데이터변환수, 신경망의 깊이는 모델의 성능에 영향이 있는 것으로 파악 되었고, 그 중에서도 신경망의 깊이가 모델의 성능에 제일 큰 영향을 미치는 모습을 볼 수 있었다. 한편, 데

이터변환수와 배치사이즈는 신경망의 깊이보다는 작지만 서로 비슷한 정도로 모델의 성능에 영향을 미쳤다.

이상치탐지 모델은 일반데이터의 특징을 최대한 파악하는 것이 목적으로, 즉 일반 데이터에 과적합한 모델을 만드는 것이라고도 생각할 수 있다. 따라서 신경망의 깊이가 너무 얕거나, 데이터 변환 수가 적거나, 배치 사이즈가 작을 경우, 일반 데이터의 특징을 제대로 파악하지 못해 이상치 데이터를 일반데이터로 판단하는 오류가 모델 성능에 영향을 미쳤을 것이라 생각해볼 수 있다. 반대로, 신경망 깊이가 너무 깊거나, 데이터 변환 수가 많거나, 배치 사이즈가 클 경우에는 모델이 트레이닝 세트에 너무 과적합되어 테스트데이터 속 일반데이터도 이상치로 구분해서 생기는 영향을 생각해보아야 할 것이다. 또한, 신경망의 깊이가 너무 깊어지면 학습을 진행하여도 모델의 성능 이 변화하지 않는 모습을 볼 수 있었는데, 배치정규화(batch normalization)과정을 거친 후에는 학습이 다소 원활히 진행된 것을 보아[부록 ㄷ.] 모델의 파라미터를 학습시키기 위한 경사가 소멸했던 것이 원인으로 파악된다.

에포크와 학습률 변화에 대해서는 모델의 성능이 큰 영향을 받지 않은 모습을 보였는데, 이는 GOAD 모델 속 옵티마이저(Optimizer)인 adam이 잘 작동하였기 때문으로 생각된다. 이를 직접적으로 확인하기 위해서는 다른 옵티마이저를 사용하여 에포크 및 학습률 변경에 따른 t-검정을 재수행한 후 여전히 “이상치탐지 모델을 학습시킬 때, 에포크 및 학습률은 모델의 성능에 영향을 끼치지 못한다.”는 귀무가설을 기각할 수 있는 지를 확인해보아야 할 것이다.

이번 연구에서는 수많은 이상치탐지 모델 중 2020년 기준으로 SOTA성능을 보여주는 GOAD 모델 하나에 대해서만 하이퍼파라미터를 변화시키고 모델의 성능 변화 추이를 관찰 및 검정하였다. 따라서 “이상치탐지 모델은 배치사이즈 및 신경망의 깊이 변화에는 성능에 영향을 많이 받고, 에포크 및 학습률에 대해서는 성능에 영향을 크게 받지 않는다.”라고 일반화시키기에는 많은 무리가 있다. 다만, 성능 변화에 큰 변화를 주고 싶은 이상치탐지 모델이 있을 경우, 최우선적으로 신경망의 깊이를 바꾸고 이후 배치사이즈 및 데이터변환수(데이터증강을 하는 모델에 한 함)을 바꾸고 마지막에 에포크 및 학습률을 변동시키는 순으로 모델의 성능 변화에 있어 하이퍼파라미터의 변경 순서를 정하는 데에는 도움이 될 수 있다고 생각한다.

5. 참고문헌

[1] D. Masters and C. Luschi, *Revisiting Small Batch Training for Deep Neural Networks*, arXiv, 2018

[2] N. S. Keskar et al., *On Large-Batch training for Deep Learning: Generalization Gap and Sharp Minima*, ICLR, 2017

[3] Dmytro Mishkin, Nikolay Sergievskiy, Jiri Matas, *Systematic evaluation of CNN advances on the ImageNet*, arXiv, 2016

[4] Chandrahas Mishra, D. L. Gupta, *Deep Machine Learning and Neural Networks: An Overview*, IAES, Vol. 6, No. 2, pp. 66~73, 2017,

[5] Liron Bergman and Yedid Hoshen, *Classification-Based Anomaly Detection for General Data*, ICLR, 2020.

[6] Lukas Ruff, Robert A. Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, Marius Kloft, *Deep Semi-Supervised Anomaly Detection,* ICLR,2020

[7] Erfani, Sarah M., et al. *High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning,* Volume 58, pp. 121-134, 2016.

[8] Lukas Ruff, Robert A. Vandermeulen, Nico Gornitz, Lucas Deecke, Shoaib A, Siddiqui, Alexander Binder, Emmanuel Muller, Marius Kloft, *Deep One-Class Classification*, PMLR 80:4393-4402, 2018.

[9] Huan-gang Wang, Xin Li & Tao Zhang, *Generative adversarial network based novelty detection using minimized reconstruction errors*, Frontiers Inf Technol Electronic Eng 19, 116-125, 2018

[10] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, Haifeng Chen, *Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection*, ICLR, 2018

[11] 오미애, 박아연, 김용대, 진재현, 기계학습(Machine Learning)기반 이상 탐지(Anomaly Detection)기법 연구 - 보건사회분야를 중심으로, 한국보건사회연구원 연구보고서, pp. 68-74, 2018

[12] Yim KH, Nahm FS, Han KA, Park SY. *Analysis of statistical methods and errors in the articles published in the korean journal of pain*, Korean J Pain. 2010

[13] B. Odoi, S. Samita, S. Al-Hassan, S. Twumasi-Ankrah, *Efficiency of Bartlett and Levenes Tests for Testing Homogeneity of Variance Under Varying Number of Replicates and Groups in One - Way ANOVA,* IJITEE, 2019

[14] Tae Kyun Kim, *T test as a parametric statistic*, Korean Journal of Anesthesiology, 2015

[15] LEVENE, H. (1960). *Robust tests for equality of variances. In Contributions to Probability and Statistics (I. Olkin, ed.),* 278 - 292. Stanford Univ. Press, Palo Alto, CA.

[16] Dr. Hossein Arsham, Miodrag Lovric, *Bartlett`s Test*, International Encyclopedia of Statistical Science, 2011

6. 부록

ㄱ. 배치사이즈(batch size), 에포크(epoch), 학습률 (learning rate), 데이터변환수(ndf), 신경망깊이 (depth) 변경에 따른 Levene 및 Bartlett 테스트 결과값 및 결과값에 따른 분산성 유형 판별

1. 배치사이즈(기본값 64):

batch size	Levene Result (t-val/p-val)	Bartlett Result (t-val/p-val)	Variance type
16	12.495/0.000	13.698/0.000	이분산성

32	9.918/0.002	7.850/0.005	이분산성
128	9.890/0.002	8.318/0.003	이분산성
256	13.046/0.000	13.480/0.000	이분산성

2. 에포크(epoch, 기본값 64):

epoch	Levene Result (t-val/p-val)	Bartlett Result (t-val/p-val)	Variance type
2	0.132/0.716	0.312/0.576	동분산성
5	1.720/0.194	1.776/0.182	동분산성
10	1.446/0.233	0.626/0.428	동분산성
20	1.571/0.216	1.118/0.290	동분산성

3. 학습률(learning rate, 기본값 0.001):

learning rate	Levene Result (t-val/p-val)	Bartlett Result (t-val/p-val)	Variance type
0.1	0.240/0.473	0.473/0.491	동분산성
0.01	1.201/0.277	0.850/0.356	동분산성
0.0001	1.019/0.316	0.615/0.432	동분산성
0.00001	0.002/0.959	0.049/0.823	동분산성

4. 데이터변환수(ndf, 기본값 0.001):

ndf	Levene Result (t-val/p-val)	Bartlett Result (t-val/p-val)	Variance type
2	2.382/0.128	1.784/0.181	동분산성
4	2.132/0.149	1.537/0.214	동분산성
16	0.032/0.857	0.040/0.841	동분산성
32	1.447/0.233	2.857/0.090	동분산성

5. 신경망깊이(depth, 기본값 1)

depth	Levene Result (t-val/p-val)	Bartlett Result (t-val/p-val)	Variance type
2	7.792/0.007	6.919/0.008	이분산성
4	14.486/0.000	17.845/0.000	이분산성
8	24.000/0.000	32.772/0.000	이분산성
16	0.521/0.473	0.205/0.649	동분산성

ㄴ. 배치사이즈(batch size), 에포크(epoch), 학습률 (learning rate), 데이터변환수(ndf), 신경망깊이 (depth) 변경에 따른 t-검정(동분산성의 경우), 웰쉬 t-검정(이분산성의 경우) 결과값

1. 배치사이즈(웰쉬 t-검정 사용):

batch size (기본값: 64)	standard t-test (t-val / p-val)	Welch t-test (t-val / p-val)
16	7.070 / 0.000	7.070 / 0.000
32	1.568 / 0.122	1.568 / 0.124
128	14.515 / 0.000	14.515/ 0.000
256	24.378 / 0.000	24.378/ 0.000

2. 에포크(t-검정 사용):

epoch (기본값: 1)	standard T-test (t-val / p-val)	Welch T-test (t-val / p-val)
2	-0.425/ 0.673	-0.425/ 0.673
5	1.217/ 0.229	1.217/ 0.229
10	0.472/ 0.638	0.472/ 0.638
20	1.459/ 0.151	1.459/ 0.151

3. 학습률(t-검정 사용):

learning rate (기본값: 0.01)	standard T-test (t-val / p-val)	Welch T-test (t-val / p-val)
0.1	-0.429 / 0.670	-0.429/ 0.670
0.01	-0.759 / 0.451	-0.759/ 0.451
0.0001	-0.259 / 0.796	-0.259/ 0.796
0.00001	-0.907 / 0.368	-0.907/ 0.368

4. 데이터변환수(t-검정 사용):

ndf (기본값: 8)	Standard T-test (t-val / p-val)	Welch T-test (t-val / p-val)
2	1.633 / 0.108	1.633 / 0.108
4	-1.047 / 0.300	-1.047 / 0.300
16	8.516 / 0.000	8.516 / 0.000
32	19.192 / 0.000	19.192 / 0.000

5. 신경망깊이(웰쉬 t-검정 사용):

depth (기본값: 1)	standard T-test (t-val / p-val)	Welch T-test (t-val / p-val)
2	1.969 / 0.054	0.000 / 1.000
4	82.344 / 0.000	82.344/ 0.000
8	290.159/ 0.000	290.159/0.000
16	403.908/ 0.000	403.908/0.000

ㄷ. batch normalization 적용에 따른 Gradient vanishing 현상 비교

depth	Gradient Vanishing before BN	Gradient Vanishing after BN
16	468	5
32	500	291
Total test case = 500 Counted gradient vanishing cases are the ones which shows 40.6153 f1-score performance, because 40.6153 is the score from the initialization.		

협동 로봇 고장 진단을 위한 요구사항 분석 및 블랙박스 설계

김양곤[1(O)], 유동연[2], 박예슬[2], 이정원[1,2]
아주대학교 전자공학과[1], 아주대학교 AI융합네트워크학과[2]
e-mail : djwoajs96@ajou.ac.kr, dong0125@ajou.ac.kr,
yeseuly@gmail.com, jungwony@ajou.ac.kr

Analysis of Requirements and Design of Blackbox for Fault Diagnosis of a Collaborative Robot

Yang-Gon Kim[1(O)], Dong-Yeon Yoo[2], Ye-Seul Park[2], and Jung-Won Lee[1,2]
Department of Electrical and Computer Engineering, Ajou University[1],
Department of AI Convergence Network, Ajou University[2]

요 약

스마트팩토리는 제품의 불량이나 생산의 중단이 없이 최대 효율로 제품을 생산하는 것을 목표로 한다. 이를 위해 설비 기기에 대한 결함, 비정상, 고장에 대해 정확히 예측하는 것이 필수적으로 요구되지만 데이터를 통해 기기의 상태를 파악할 수 있지만 협동 로봇의 경우 수행된 작업에 따라 센싱되는 데이터의 패턴이 매우 상이하다. 따라서 다양한 센싱된 데이터에 대한 의미(예: 작업 종류, 상태, 통신 등)적인 분석을 위해서 체계적인 데이터 생성과 저장 체계가 필요하다. 따라서 본 논문에서는 이러한 데이터 블랙박스를 제안하는 것으로 협동 로봇의 고장 진단을 수행하기 위한 제조 시나리오 및 데이터 생명주기에 따른 데이터 요구사항을 도출하고, 이를 기반으로 데이터 블랙박스를 구현한다. 본 연구에서는 블랙박스 개발을 위해 요구사항 기반의 5가지 세부 모듈과 2가지 데이터 생성 체계를 설계하였으며, 기존의 진단 방식에서 수행하던 1레벨 수준(센싱 정보) 정보를 제공할 뿐만 아니라, 2레벨 수준(작업 정보) 및 3레벨 수준(고장 정보)의 정보를 제공함으로써 효과적인 데이터 기반의 고장 진단을 보조함을 검증하였다.

1. 서 론[i]

스마트팩토리에서는 다양한 설비들과 작업 공간을 공유하고 작업자와 유연한 협동 과정을 추구한다. 이 중 협동 로봇은 스마트팩토리에서 인간과의 협업 운용 조건으로 인간과의 상호 작용을 위한 안전성과 프로그래밍을 통한 작업의 설계로 유연성, 신뢰성, 자율성을 충족하는 산업용 로봇이다[1].

스마트팩토리는 제품의 불량이나 생산의 중단이 없이 최대 효율로 제품을 생산하는 것을 목표로 한다. 하지만 산업 현장의 설비 기기들은 갑작스러운 고장으로 인해 가동이 중단되는 현상이 발생할 수 있으며, 고장이 사전 예방되지 못한다면 제품의 불량이나 생산에 치명적인 영향을 끼친다. 따라서 스마트팩토리는 설비 기기에 대한 결함, 비정상, 고장에 대해 정확히 예측하는 것은 필수이며 이를 위해 예지보전(Predictive Maintenance) 기술이 적용되고 있다[2].

예지보전 기술은 관찰된 데이터를 기반으로 고장이 발생하기 전에 시스템의 효율성, 생산성, 남은 유효 수명 등을 측정하여 관리하는 시스템으로 고장이 발생하기 전에 예지보전을 결정하는 효율적인 방식으로 상태 기반 유지보수(Condition-Based Maintenance)이 대표적이다[3]. 상태 기반 유지보수란 기기의 상태를 파악하고 예측하는 방식으로 손상이나 고장이 감지된 경우에만 유지보수를 시행한다. 따라서 비용적인 부분에서 효율이 크다는 장점이 있으며, 최근에는 IoT 기술의 발달을 통해 더욱 유연하고 범용적으로 적용되고 있다[4~6].

하지만 협동 로봇의 경우 다양한 작업을 수행하도록 프로그래밍 되며 기존 연구들과 같이 기본적으로 센싱된 데이터로만 동작을 분석하기에는 작업 프로그램에 따른 데이터 패턴이 다양하여 분석 복잡도가 높다. 따라서 협동 로봇의 데이터 생성은 기본적인 센서 데이터와 더불어 동작의 특성을 분석할 수 있는 프로그램 및 명령어의 종류, 상태 등을 동시에 생성해야 한다. 협동 로봇의 데이터 생성에 있어 고려되어야 하는 특성은 다음과 같이 정리된다.

첫째, 협동 로봇은 다관절 기기이기 때문에 각 관절에 따른 상태 정보가 구분되어 생성되어야 한다. 다관절의 협동 로봇은 하나의 관절에서 발생하는 상태가 주변의 관절에게 영향을 주는 특성이 있으므로, 모든 관절에 대한 관찰을 수행해야한다. 따라서 각 관절에 대한 상태 정보가 구분되어, 설비 기기에 대한 체계적인 진단을

[i] 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2020R1A2C1007400).

보조할 수 있어야 한다.

둘째, 협동 로봇의 각 관절에는 다양한 센서(예: 전류, 속도, 위치, 토크 등)가 내장되어 있기 때문에, 이를 고려한 구조적이고 통합된 형태의 데이터 모델이 필요하다. 각 관절에서 센싱된 데이터가 통합된 형태의 모델 없이 개별적이고 독립적으로 생성되는 경우, 데이터 동기화(Synchronization) 문제나 데이터 처리를 위한 프로세서 스케줄링(Scheduling) 문제 등이 발생하게 된다. 따라서 각 관절에 내장된 센서의 특성을 반영한 통합 데이터 모델이 필요하다.

셋째, 협동 로봇은 작업자에 의해 다양한 환경 및 동적 조건을 고려한 프로그램을 기반으로 작업이 수행된다. 이로 인해, 실시간으로 생성되는 센싱 데이터는 일관된 패턴 없이 수행된 작업에 따라 무수히 많은 데이터 패턴을 보이게 된다. 따라서 물리 신호 데이터에 대한 주기적인 센싱 방식과 함께, 수행된 작업 정보에 대한 로그 데이터 생성이 필요하며, 이를 기반으로 추출된 센서 데이터와 해당 부분의 로그 데이터가 함께 분석이 될 수 있어야 한다.

본 논문에서는 협동 로봇의 특성을 고려한 고장 진단을 위한 블랙박스 설계가 목표이다. 여기서 말하는 블랙박스란 협동 로봇에서 고장 진단을 위해 요구되는 데이터가 통합 생성 및 수집되는 소프트웨어를 말한다. 블랙박스 설계를 위해 제조 프로세스 시나리오를 정의하고 이를 기반으로 예지보전과 데이터 관점에서의 요구사항을 도출한다. 그리고 요구사항 기반의 5가지 세부 모듈과 2가지 데이터 생성 체계를 설계하였으며, 기존의 진단 방식에서 수행하던 1레벨 수준(센싱 정보) 정보 제공뿐만 아니라, 2레벨 수준(작업 정보) 및 3레벨 수준(고장 정보)의 정보를 제공함으로써 효과적인 데이터 기반의 고장 진단을 보조함을 검증하였다.

2. 관련 연구

2.1 상태 기반 유지보수를 통한 예지보전 기술

2장에서는 예지보전과 고장진단을 위한 데이터 관점에서 관련 연구를 분석하였으며, 2.1절에는 상태기반 유지보수를 통한 예지보전 기술을 제시한다. 예지 보전을 위한 기술로는 상태 기반 유지보수가 대표적이며 비용을 절감하기 위해 장비의 현재 상태를 진단하고, 남은 잔여 수명을 예측하여 최적의 유지보수 시기를 결정하는 방식이다[4]. 이와 같은 방식은 기기로부터 생성되는 데이터를 기반으로 기기의 상태를 파악하여 고장을 예측하고 유지보수 계획을 수립한다[5]. 또한, 기기의 손상 또는 고장이 감지된 경우에만 유지보수를 수행하기 때문에 불필요한 예방 활동 없이 유지보수 간격을 최적화할 수 있다[6].

상태 기반 유지보수 방식은 크게 2가지로 나뉘는데, 물리적인 고장 메커니즘에 관한 해석 기반의 손상 모델을 통해 상태를 파악하는 모델 중심 접근법(Model-driven Approach)과 데이터의 통계적 특성을 이용하여 정상과 고장을 구분할 수 있는 데이터 중심 접근법(Data-driven Approach)이 존재한다. 과거에는 모델 중심의 접근법을 이용하여 고장 진단을 수행하였지만, IoT 기술의 발달로 인해 설비 기기로부터 물리적 신호 데이터를 수집하는 것이 보편화 되었으며, 이로 인해 데이터 중심 방법이 유연하고 범용적으로 적용할 수 있게 되었다.

데이터 중심 방법으로 [7]의 연구에서는 IoT 기술을 기반으로 의료 제조 공장 환경에서 재봉 기기의 전력 및 작업 상태를 측정할 수 있는 모니터링 시스템을 제안하였으며, [8]의 연구에서는 산업 기계에 대해 CBM 방식의 중요한 측면인 진단과 예후(Diagnostics & Prognostics)와 관련된 연구를 데이터 수집, 데이터 처리, 유지보수 의사 결정의 단계로 검토하였다.

이처럼 단순히 고장이나 결함에 대한 진단을 수행하는 것뿐만 아니라, 수집된 데이터 내에서 유의미한 데이터를 도출하고 이에 대한 처리, 관리의 중요성이 높아지고 있다. 그러나 아직까지 협동 로봇을 위한 예지 보전 관점의 데이터 수집과 저장을 위한 블랙박스의 체계적인 요구사항 정의에 관한 연구는 미비한 실정이다. 따라서 본 논문에서는 고장 진단을 수행하기 위해 필요한 요구사항을 도출하여 이를 기반으로 블랙박스를 설계 및 구현한다.

2.2 데이터 기반 로봇 고장 진단 기술

2.2절에서는 로봇의 고장 및 비정상 상태를 진단하기 위한 관련 연구를 소개한다. 대부분의 연구들의 경우 로봇 내부에 부착하여 추출해내는 센서(전류, 전압, 온도 센서 등)로 데이터를 추출하고 분석하여 고장 및 비정상을 진단하며, 각각의 연구는 표 1과 같다.

표 1 – 데이터 기반 로봇 고장 관련 연구 정리

관련 연구	목적	고장진단 방식	사용 센서
[9]	웹 기반 로봇 제어 및 상태 모니터링	수학 모델	비전 포스 가속도 변환기
[10]	경쟁 학습을 통한 고장 진단 및 예후	HMM 기반 클러스터링	반작용 토크
[11]	6축 로봇의 각 관절 데이터를 통한 고장 진단	CNN 구조의 변이 인코터 기반 모델	전류 관절각
[12]	혼합 모델 기반 정의되지 않는 고장 감지	수학적 모델과 GMM을 기반 모델	전류
[13]	동적 신호로 비지도학습을 통한 기계의 결함 및 오류를 감지	클러스터링	전류 자기장 진동
[14]	음향 신호를 통한 고장 진단 및 예후	사례 기반 추론 접근 방식	소리

이는 현재 데이터 생성 체계로는 특정한 작업 프로그램에 대한 고장 진단을 실현할 수 있지만 범위가 큰 다양한 프로세스로 이루어진 제조 환경에서 협동 로봇의 고장 진단은 한계가 있다. 따라서 협동 로봇에서 발생될 수 있는 고장에 대해 체계적인 진단을 수행하기 위해서는 이와 같은 고장 진단에 필요한 데이터 요구사항 분석과 유의미한 데이터에 대한 통합 생성이 필요하다. 이를 위해 본 논문에서는 협동 로봇의 고장 진단을 수행하기 위한 데이터 요구사항을 도출하고, 이를 기반으로 한 블랙박스를 설계한다.

3. 협동 로봇의 고장 진단 관점 데이터 요구사항 도출

3.1 시나리오 기반 예지보전 관점 요구사항 도출

요구사항의 도출은 크게 제조 시나리오와 데이터 생명주기 기반의 도출 방식으로 수행되며, 3.1절에는 시나리오를 기반으로 요구사항을 도출한다. 본 연구에서는 그림 1과 같이, 실제 자동차 Car-Mat를 제조 과정을 착안하여 제조 시나리오를 설계하였다. 각 단계(Phase)에는 하나 이상의 연관 디바이스(Related Device)가 존재하며, 여러 작업 프로그램을 수행하여 Car-Mat를 제조한다. 설계된 시나리오는 5가지 종류의 디바이스가 사용되며, 총 26가지의 작업이 이루어진다. 시나리오 단계별 작업은 표 2와 같이 정리된다.

제시된 시나리오는 공장 제조 라인에서의 공정 프로세스에 따른 기기의 움직임을 나타낸다. 본 연구에서는 이와 같은 과정에서 고장 진단 블랙박스를 활용할 수 있는 사용자를 다음과 같이 정의하였다.

- Operator(O): 로봇의 수행 작업 관리자
- Collaborator(C): 로봇과 협업을 수행하는 작업자
- Maintainer(M): 로봇의 결함, 고장을 분석 엔지니어
- System Manager(S): 스마트팩토리 관리자

다음은 도출된 사용자에 따라 고장 진단에 필요한 데이터 요구사항을 정의한다. 데이터 요구사항은 내재된 정보의 속성에 따라, 5가지(가동정보, 상태정보, 통신정보, 물리정보, 고장정보)로 구분하여 도출되며, 세부적인 데이터 요구사항 사양은 표 3과 같다. 이 때, 표 3에 도출된 15개의 일반 요구사항은 제조 프로세스 모든 단계에서 공통적으로 필요한 데이터의 종류를 나타내며, 단계별로 선택적으로 필요한 요구사항은 표 3의 일반 요구사항과 연계되어 표 4와 같이 대표적으로 33개의 선택 요구사항으로 도출되었다.

이와 같이 도출된 요구사항은 협동 로봇의 고장 진단을 수행하기 위한 설계 요구사항의 역할을 수행한다. 이에 대한 세밀한 고려를 통해, 통합된 형태의 데이터 규격을 정의할 수 있다. 하지만 해당 요구사항의 경우, 데이터의 활용성 (정제, 활용도, 소멸 등)의 측면은 고려되어 있지 않다. 따라서 3.2절에서는 데이터 생명주기에 따른 요구사항을 제시한다.

표 2 – 스마트팩토리 제조 프로세스 시나리오 단계별 작업

단계		작업 프로그램	연관 디바이스
1	원단 적재	• 원단 준비 • 원단 이동	• 협동 로봇 • 컨베이어벨트
2	원단 절단	• 원단 절단 • 절단된 원단 이동	• 압축 기계 • 컨베이어벨트
3	원단 재봉	• 절단된 원단 준비 • 로고 재봉 • 재봉된 원단 적재 • 재봉된 원단 이동	• 협동 로봇 • 재봉 기계 • 컨베이어벨트
4	원단 처리	• 재봉된 원단 준비 • 재봉된 원단 풀칠 • 풀칠된 원단 처리 • 처리된 원단 적재 • 처리된 원단 이동	• 협동 로봇 • 컨베이어벨트
5	매트 재봉	• 처리된 원단 준비 • 처리된 원단 재봉 • 매트 적재 • 매트 이동	• 협동 로봇 • 재봉 기계 • 컨베이어벨트
6	고무 삽입	• 매트 준비 • 고무 삽입 • 매트 회전 • 후면 고무 삽입 • 완성된 매트 적재 • 완성된 매트 이동	• 협동 로봇 • 압축 기계 • 컨베이어벨트
7	매트 포장	• 상자 준비 • 완성된 매트 준비 • 완성된 매트 포장	• 협동 로봇 • 열처리 기계

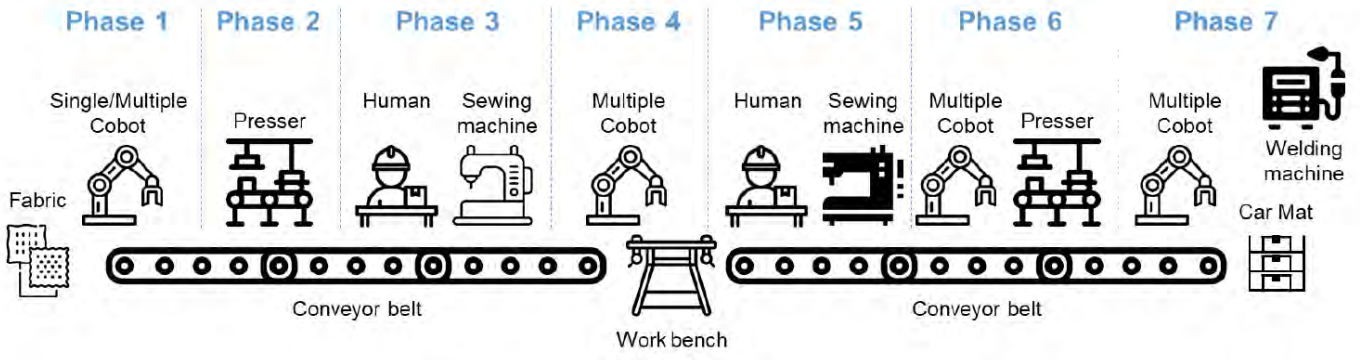


그림 1 –협동 로봇의 블랙박스 요구사항 도출을 위한 스마트팩토리 제조 프로세스 시나리오 (Car-Mat 제조)

표 3 - 시나리오 기반 데이터 속성별 일반 요구사항 도출

데이터 속성	요구사항 인덱스*	일반 요구사항	관련 사용자			
			O	C	M	S
가동정보 (OD)	CR-OD-1	전원 정보가 기술되어야 한다.	V		V	V
	CR-OD-2	프로그램 정보가 기술이 되어야 한다.	V		V	
	CR-OD-3	동작 명령어 정보가 기술되어야 한다.	V		V	
상태정보 (SD)	CR-SD-1	현재 상태 정보를 알 수 있다.	V		V	
	CR-SD-2	상태 전이 정보를 알 수 있다.		V	V	
	CR-SD-3	외부 요인 정보를 알 수 있다.		V	V	
통신정보 (CD)	CR-CD-1	단일 기종 기기 간 협력 정보를 확인할 수 있다.		V	V	V
	CR-CD-2	이기종 기기 간 협력 정보를 확인할 수 있다.		V	V	V
물리정보 (PD)	CR-PD-1	관절의 전류 & 전압이 추출되어야 한다.			V	V
	CR-PD-2	관절각이 추출되어야 한다.			V	V
	CR-PD-3	끝단의 위치 정보가 추출되어야 한다.			V	V
	CR-PD-4	관절 모터 온도가 추출되어야 한다.			V	V
고장정보 (FD)	CR-FD-1	실행 에러가 출력되어야 한다.	V		V	V
	CR-FD-2	이상 동작 정보가 출력되어야 한다.	V		V	V
	CR-FD-3	고장 정보가 출력되어야 한다.	V		V	V

*CR: Scenario-based Common Requirement,
OD: Operating Data, SD: State Data, CD: Communication Data,
PD: Physical Data, FD: Fault & Failure Data

3.2 데이터 생명주기 기반 요구사항 도출

3.2절에서는 데이터 생명주기를 기반으로 요구사항을 도출한다. 일반적으로 데이터는 생성, 정제, 분석, 활용, 소멸, 다섯 단계의 생명주기를 가지며 각 단계의 설명은 다음과 같다[15].

- **생성**: 디바이스에서 데이터가 생성되는 과정
- **정제**: 생성된 데이터를 분석할 수 있는 형태로 규격화 하는 과정
- **분석**: 정제된 데이터를 가지고 의미 있는 해석을 위해 분석을 하는 과정
- **활용**: 분석 결과를 기반을 예지보전 관점으로 생성된 정보를 활용하는 과정
- **소멸**: 생성된 데이터 및 정보 중 활용성이 떨어질 경우 제거하는 과정

표 4 - 시나리오 기반 단계별 선택 요구사항 도출

단계	선택 요구사항	연계 요구사항
1	진공 펌프의 설치 정보를 알 수 있어야 한다.	CR-S-3
	원단의 잡기 및 놓기 동작의 실패 정보가 출력되어야 한다.	CR-F-3
3	그리퍼 및 진공 펌프의 설치 정보를 알 수 있어야 한다.	CR-S-3
	협력 로봇 간의 작업 수행 정보 송수신을 확인할 수 있어야 한다.	CR-C-1
	재봉 로봇의 작업 수행 정보 송수신을 확인할 수 있어야 한다.	CR-C-2
	잘린 원단과 재봉된 원단의 잡기 및 놓기 동작의 실패 정보가 출력되어야 한다.	CR-F-2 CR-F-3
	기기 간의 충돌 감지가 되어야 한다.	CR-F-2
4	그리퍼 및 진공 펌프의 설치 정보를 알 수 있어야 한다.	CR-S-3
	협력 로봇 간의 작업 수행 정보 송수신을 확인할 수 있어야 한다.	CR-C-1
	재봉된 원단과 붙힌 원단의 잡기 및 놓기 동작의 실패 정보가 출력되어야 한다.	CR-F-2 CR-F-3
	기기 간의 충돌 감지가 되어야 한다.	CR-F-2
5	진공 펌프의 설치 정보를 알 수 있어야 한다.	CR-S-1
	재봉 로봇의 작업 수행 정보 송수신을 확인할 수 있어야 한다.	CR-C-2
	붙힌 원단과 재봉된 매트의 잡기 및 놓기 동작의 실패 정보가 출력되어야 한다.	CR-F-2 CR-F-3
	기기 간의 충돌 감지가 되어야 한다.	CR-F-2
6	그리퍼 및 진공 펌프의 설치 정보를 알 수 있어야 한다.	CR-S-3
	협력 로봇 간의 작업 수행 정보 송수신을 확인할 수 있다.	CR-C-1
	압축 로봇의 작업 수행 완료 정보 송수신을 할 수 있다.	CR-C-2
	재봉된 매트 완성된 매트의 잡기 및 놓기 동작의 실패 정보가 출력되어야 한다.	CR-F-2 CR-F-3
7	그리퍼 및 진공 펌프의 설치 정보를 알 수 있어야 한다.	CR-S-3
	협력 로봇 간의 작업 수행 정보 송수신을 확인할 수 있다.	CR-C-1
	열처리 기계의 작업 수행 완료 정보 송수신을 할 수 있다.	CR-C-2
	완성된 매트와 상자의 잡기 및 놓기 동작의 실패 정보가 출력되어야 한다.	CR-F-2 CR-F-3
	기기 간의 충돌 감지가 되어야 한다.	CR-F-2

본 연구에서는 이와 같은 데이터 생명주기의 특징을 분석하여, 표 5와 같은 세부적인 데이터 요구사항을 도출하였다. 이와 같은 요구사항은 데이터의 생성부터 소멸까지의 단계에 따라 고려되어야 하는 예지보전 관점들의 특성으로 도출된다. 도출된 요구사항은 협동 로봇의 고장이나 결함을 진단하기 위한 블랙박스 설계의 기반이 되어, 4장에서는 협동 로봇의 블랙박스 설계 내용을 제안한다.

표 5 – 데이터 생명 주기에 따른 요구사항 도출

생명 주기	요구사항 내용
생성	전원, 작업, 명령어 등의 정보가 생성되어야 한다.
	이벤트로 인한 상태전이가 발생할 경우 현재 로봇의 상태 및 전이 변화가 생성되어야 한다.
	협력 작업으로 인한 커뮤니케이션이 발생할 경우 그에 대한 정보가 생성되어야 한다.
	정해진 주기를 기준으로 전류, 전압, 자이로 등의 데이터가 주기적으로 생성되어야 한다.
	로봇에 정의된 오류 및 분석을 통한 정의된 고장 징후가 발생할 경우 정보가 생성이 되어야 한다.
정제	생성된 데이터의 주기를 기준으로 데이터가 정렬하여 동기화 되어야 한다.
	생성된 데이터의 주기를 기준으로 중복 데이터가 제거되어야 한다.
	각 생성된 정보에 대해 정해진 규격에 의해 분류되어야 한다.
분석	비정상 및 고장 동작이 발생할 경우 수행한 작업, 동작에 대한 정보를 사용할 수 있어야 한다.
	비정상 종료에 대한 상태를 통해 이전 상태 및 전이가 분석될 수 있어야 한다.
	통신 로그를 통해 효율적인 협력 작업의 통신 알고리즘이 분석될 수 있어야 한다.
	평균/표준편차와 같은 분석을 통해 비정상 동작 및 결함이 분석될 수 있어야 한다.
	생성된 기기 고장 정보를 통해 원인을 정확히 분석할 수 있어야 한다.
활용	통계를 기반으로 비정상, 결함, 오류의 구간을 검출할 수 있어야 한다.
	추출된 통계데이터를 활용하여 이상 상태 및 동작 패턴이 검출될 수 있어야 한다.
	분석된 알고리즘을 통해 협동 작업 통신의 최적화에 활용되어야 한다.
	분석의 결과를 통해 이상 징후의 임계 값을 설정할 수 있어야 한다.
	향후 발생할 수 있는 추가적인 고장을 예측하고 유연하게 대처할 수 있어야 한다.
소멸	방대한 정보량으로 인해 정제 및 분석을 통해 정보량을 줄여야 한다.
	미활용 및 불용 데이터가 제거되어야 한다.

4. 요구사항 기반 협동 로봇 블랙박스 설계

4.1 협동 로봇 블랙박스 정의

도출한 요구사항을 기반으로 협동 로봇 블랙박스를 설계 하기위해 협동 로봇 블랙박스의 정의와 협동 로봇의 상태 및 동작 정보에 대한 모델링이 필요하다. 먼저 4.1절에는 협동 로봇 블랙박스에 대해 정의한다. 블랙박스란 협동 로봇에서 고장 진단을 위해 관찰된 데이터가 통합 생성 및 수집되는 소프트웨어를 말한다. 따라서 블랙박스의 설계는 3장에서 도출된 데이터 요구사항(가동 정보, 상태 정보, 통신 정보, 물리 정보, 고장 정보)을 기반으로 한다. 이를 반영하여, 블랙박스는 크게 5개의 세부 모듈로 구성되어 있으며, 세부적인 설명은 다음과 같다.

- **가동 정보(OD)** 생성 모듈: 협동 로봇의 전원, 작업 프로그램, 동작 명령어 정보를 생성
- **상태 정보(SD)** 생성 모듈: 협동 로봇의 동작에 의한 상태 정보 및 상태 전이 정보를 생성
- **통신 정보(CD)** 생성 모듈: 협동 로봇과 서버 혹은 다른 기기와의 통신 상태를 생성
- **물리 정보(PD)** 생성 모듈: 협동 로봇에 내장된 센서로부터 물리 신호 정보를 생성
- **고장 정보(FD)** 생성 모듈: 협동 로봇의 고장 및 결함 정보를 생성

4.2 협동 로봇의 상태 및 동작 모델링

4.2 절에서는 협동 로봇 객체를 식별하기 위해 상태 및 동작 모델을 설계한다. 협동 로봇의 경우 크게 대기하는 상태(Idle), 작업을 수행하는 상태(Running), 수행을 종료한 상태(Terminated)로 구성된다. 또한, 협동 로봇은 작업자 및 다른 기기들과 협력 작업을 수행할 수 있는 상황을 고려해야 한다. 따라서 협력 상황에 따른 통신 상태(Communication)도 상태 모델에 포함된다. 그리고 종료 상태의 경우, 고장 진단을 위해 오류가 발생해 중단된 상태(Suspension)와 정상적으로 종료된 상태(Terminated_Normal), 비정상적으로 종료된 상태(Terminated_Abnomal)를 세부적으로 설계하였다. 이에 대한 자세한 협동 로봇의 상태 다이어그램은 그림 2 와 같으며 상태 다이어그램의 각 상태에 따른 정의는 표 6 과 같다. 또한, 상태에 따라 정보 생성이 필요한 연관성을 기반으로, 표 6 의 우측과 같이 블랙박스 생성 모듈 연계하였다. 블랙박스의 데이터 생성 체계는 이와 같은 연관성을 기반으로 한다.

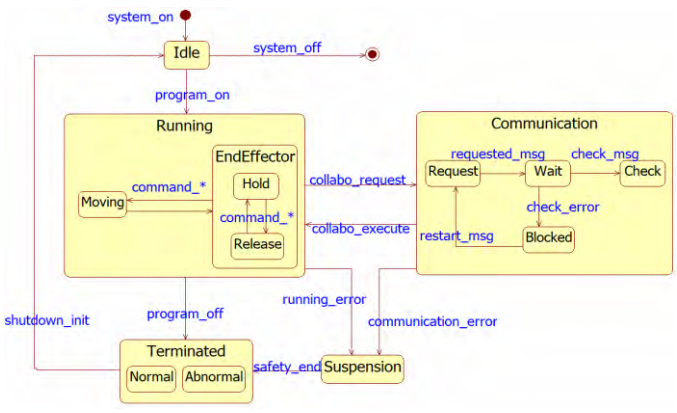


그림2 – 협동 로봇 상태 다이어그램

표 6 – 협동 로봇의 상태 정의 및 블랙박스 모듈 연계

상태		설명	관련 모듈				
			OD	SD	CD	PD	FD
InitialState		협동 로봇의 초기상태	V	V			
FinalState		협동 로봇의 종료 상태	V	V			
Idle		프로그램 시작 및 시스템 종료에 대한 준비 상태		V			
Running	Moving	협동 로봇의 이동 명령어로 인해 움직이는 상태		V			
	Hold	협동 로봇의 엔드이펙터 명령어가 활성화된 상태		V			
	Release	협동 로봇의 엔드이펙터 명령어가 비활성화된 상태		V			
Comm-unication	Request	협력 요구 사항을 수신 및 송신하는 상태		V	V		
	Wait	협력 요구사항에 대해 대기하는 상태		V	V		
	Check	협력 요구 사항이 최종 확인되어 실행될 준비를 하는 상태		V	V		
	Blocked	협력 요구 사항에 대해 조건 부족으로 인해 실패된 상태		V	V		
Suspension		프로그램 동작 중 에러로 인해 중단된 상태		V			V
Terminated	Normal	프로그램이 종료된 상태 (Default)		V			
	Abnormal	프로그램이 에러로 인해 종료된 상태		V			V

4.3 협동 로봇 고장 진단 데이터 모델 설계

표 6에서 분석한 바와 같이, 블랙박스의 세부 모듈 중 물리 정보(PD) 생성 모듈의 경우, 기기의 센서로부터 주기적으로 신호를 센싱해야 하며, 나머지 4가지 모듈(OD, SD, CD, FD)의 경우, 상태의 전이(Transition)가 발생하는 경우 간헐적으로 생성되어야 한다. 주기적으로 생성되는 물리정보와 달리 전이가 발생할 경우에만 생성되는 특징을 통해 네트워크 및 저장 공간에 대한 부담이 경제적으로 효율적인 활용을 할 수 있다. 따라서 본 연구에서는 이러한 특성을 반영하여 데이터 생성 체계를 크게 2가지(센서 데이터 생성 및 이벤트 로그 생성)로 구분하여 구축하였다.

4.3.1 센서 데이터 모델 설계

블랙박스에서 생성되는 센서 데이터는 각 관절에 내장된 여러 센서로부터 수집된 물리 신호 정보를 나타낸다. 협동 로봇의 각 관절에는 다양한 센서(예: 전류, 속도, 위치, 토크 등)가 내장되어 있기 때문에, 이를 고려한 구조적이고 통합된 형태의 데이터 모델링이 필요하다. 따라서 본 연구에서는 제안하는 규격은 크게 주기(TICK), 데이터(DATA)로 분류되는 형태의 통합 센서 데이터 모델을 제안한다. 이에 대한 세부적인 설명은 다음과 같이 이어진다.

① **주기(TICK)**: 센서로부터 물리 신호가 센싱된 절대 시각을 의미한다. 복수개의 센서로부터 센싱된 시각은 주기를 기준으로 동기화 된다. 복수개의 센서와 함께 이벤트 로그 데이터와 병합될 수 있다.
② **데이터(DATA)**: 센서의 종류(예: 위치, 속도, 토크, 전류 등)을 나타내는 종류(TYPE)와 각 센서로부터 어느 관절(JOINT))로부터 수집되었는지 구분되어 측정한 실제 물리 신호 값(VALUE)을 세부요소로 갖는 측정(MEASUREMENT)으로 나뉜다. 예를 들어, 6축 다관절 로봇의 경우 측정 데이터는 6개의 세부요소를 갖는 그룹으로 구성될 수 있다.

4.3.2 이벤트 로그 데이터 모델 설계

블랙박스에서 생성되는 이벤트 로그의 경우 크게 가동, 상태, 통신, 고장 정보를 생성해야 한다. 따라서 본 연구에서는 이에 대한 구분을 수행할 수 있도록 크게 날짜(DATETIME), 레벨(LEVEL), 그룹(GROUP), 주기(TICK), 메시지(MESSAGE)의 총 5가지 형식으로 나뉘는 형태의 로그 데이터 규격을 제안한다. 이에 대한 세부적인 설명은 다음과 같이 이어진다.

① **날짜(DATETIME)**: 이벤트 로그 발생을 기준으로 년도, 월, 일, 시간, 분, 초로 생성된다.
② **레벨(LEVEL)**: 이벤트 로그 발생시 INFO, WARN, FAULT, DEBUG의 4가지 정보로 구분한다. INFO의 경우 상태 및 전이 정보, 프로그램 및 명령어 정보를 나타낸다. WARN와 FAULT의 경우 비정상 또는 고장의 정보이며, DEBUG는 협동 로봇의 명령어 등의 동작이나 통신의 오류 정보를 다룬다.
③ **그룹(GROUP)**: 이벤트 로그 발생 시 MACHINE, FRAMEWORK, CONTROLLER, DETECTOR의 4가지 그룹으로 구분된다. MACHINE의 경우 하드웨어 부분의 모듈, FRAMEWORK는 통신 모듈, CONTROLLER는 제어 부분의 모듈, DETECTOR는 고장 진단 부분의 그룹을 의미한다.
④ **주기(TICK)**: DATETIME과 같이 이벤트 로그 발생을 기준으로 생성되는 절대 시각을 나타낸다. 센서 데이터와 마찬가지로 동시간대 생성된 데이터는 주기 값을 기준으로 수행할 수 있다.
⑤ **메시지(MESSAGE)**: IDENTIFIER와 NOTE로 구성된 문자열 형식의 데이터이다. 도출한 요구사항과 상태 다이어그램을 통해 세부적인 식별자(#BOOT, #STATE, #EVENT, #PROGRAM, #MOTION, #COMM, #ERROR, #ANOMALY, #FAULT)가 정의된다. 식별자에 대한 정보는 NOTE를 통해 문자열 형식으로 정보가 제공된다.

5. 구현 및 평가

5.1 블랙박스 구현

5장에서는 협동 로봇의 고장 진단을 위한 블랙박스를 개발하고 평가한다. 이를 위해 5.1절에서는 블랙박스 구현을 진행하며, Python 3.7 버전을 기반의 환경과 실험에는 6축 협력 로봇인 Niryo One을 사용하였다. Niryo One의 동작 제어의 경우, 제공되는 API를 활용하였고 로봇 팔 끝단에 장착된 센서를 통해, 위치데이터(x, y, z, roll, pitch, yaw)와 각 축의 각도 값(radian), 온도를 내부적으로 추출하였다. 또한, 라즈베리파이 3B+ 보드와 전류 센서를 외부에 장착하여 모터에 인가되는 전류 값을 수집하였고, 10ms 단위로 시계열 데이터를 수집할 수 있도록 환경을 구성하였다. 그리고 테스트 프로그램의 경우, 실제 자동차 공장에서 Car-mat를 제조하는 과정의 Gluing Task를 모티브로 하여 Car-mat의 모양에 따라 로봇 암의 말단부(End-effector)가 해당 모양의 가장자리를 Gluing 하는 방식의 테스트 프로그램을 구성하였다.

프로그램이 동작할 때마다 설계된 데이터 모델 스키마에 맞게, XML 형식으로 생성하였다. 그림 3은 생성된 센서 데이터 및 이벤트 로그를 예시한다. 그림 3(a)는 전류 센서로부터 조인트 별로 수집된 12개의 연속된 센서 값을 나타낸다. 그림 3(b)는 가동 정보, 상태 정보, 통신 정보, 고장 정보를 나타낼 수 있는 이벤트 로그의 일부를 나타내며, 로봇의 부팅이 일어난 순간의 데이터 일부로 4.3.2절과 같이 정의된 메시지 식별자 “#BOOT”와 “EVENT”를 이용하여 기술되었다. 이와 같은 데이터는 동일한 스키마를 공유하는 데이터베이스에 적재될 수 있으며, 주기적으로 생성되는 센서 데이터를 통해 시계열 데이터 분석 뿐만 아니라, 비주기적(전이 발생)으로 생성되는 이벤트 로그를 통해 메시지(MESSAGE) 필드의 식별자를 기준으로 생성된 데이터에 대한 의미적인 분석을 수행할 수 있다.

그림 3 – 블랙박스 센서 데이터 및 이벤트 로그 생성 예시

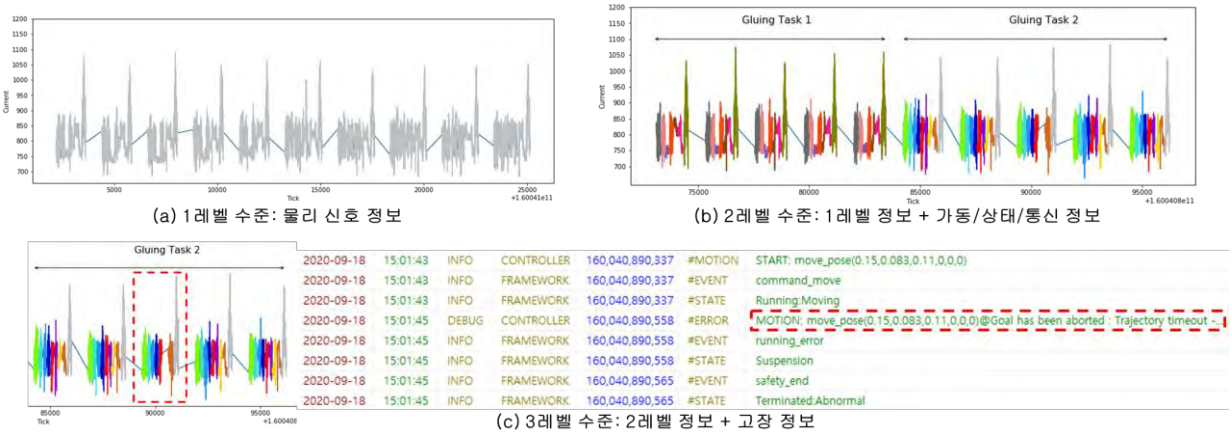
5.2 데이터 생성 결과 분석

본 연구에서는 생성된 센서 데이터(물리 신호 정보)를 기준으로 연계될 수 있는 정보(예: 가동 정보, 상태 정보, 고장 정보 등)와 병합하여 단계별로 분석을 진행하였다. 그림 4는 테스트 프로그램을 구동해 특정 시간(TICK: 160,040,873,117~160,040,896,225)에 대해 제안하는 블랙박스를 사용하지 않는 경우와 블랙박스를 사용하여 데이터 생성할 때 추출된 데이터를 그래프로 표현하였다. 센서 데이터에 담긴 정보의 양에 따라 크게 3가지 단계로 나누었고, 세부적인 설명은 다음과 같다.

① **1레벨 수준(물리 신호 정보)**: 그림 4(a)는 실시간으로 생성된 데이터를 시계열로 정렬하여 가시화한 그림이다. 기존의 상태 기반 유지보수 방법은 이와 같은 형태로 생성된 데이터를 기반으로 진단 및 예측 모델을 개발한다. 하지만 협동 로봇의 경우 수행된 작업에 따라 상이한 패턴을 보이기 때문에, 1레벨 수준의 정보의 양으로는 분석에 한계를 갖는다.

② **2레벨 수준(1레벨 정보 + 가동/상태/통신 정보)**: 센싱 정보에 작업의 정보(가동, 상태, 통신 정보)가 포함된 데이터로 세부적인 데이터 분석이 가능한 수준을 의미한다. 그림 4(b)는 데이터(물리 신호 정보)에 작업 프로그램과 동작 명령어(가동 및 상태 정보)가 수행된 시점을 적용한 것으로, 작업 프로그램과 동작 명령어의 종류에 따라 색상을 구분하였다. 이를 통해, 동일한 작업의 경우 비슷한 수준의 전류 값을 생성함을 확인할 수 있으며, 반복되는 동작에 따라 일관된 패턴을 보임을 확인할 수 있다.

③ **3레벨 수준(2레벨 정보 + 고장 정보)**: 그림 4(c)의 경우, 그림 4(b)의 8번째 수행 프로그램에서 오류가 발생한 상황의 센싱 데이터를 확대하여 보여주고 있다. 구체적으로는, Gluing Task 2의 세 번째 동작을 수행하는 과정에서 두 가지 명령어(보라색, 노란색 부분)가 오류로 인해 동작이 안된 것을 볼 수 있다. 오류가 발생한 시점을 기준으로 해당 명령어에 대한 데이터가 자동으로 제거되게 설정되어 있어 블랙박스를 사용한 경우 두 가지 명령어(보라색, 노란색)의 인덱스 정보를 통해 고장의 원인 및 영향력에 대한 분석이 후처리 필요없이 가능하다. 또, 에러가 발생한 두 가지 명령어(보라색, 노란색)의 다음 동작 명령어(갈색)의 경우 반복 동작에 따른 패턴이 변형된 것을 볼 수 있다. 이러한 시각적인 분석은 1레벨 수준에서는 이뤄질 수 없다. 이를 통해 그림 4(c)의 우측과 같이 센싱 데이터가 발생한 시점의 적재된 이벤트 로그 데이터(상태 정보)를 분석함으로써, 동시간대 생성된 데이터에 대한 의미적인 해석을 수행하고, 앞뒤 구간에 수행된 작업의 패턴을 분석하여 오류의 원인이나 연관성을 파악할 수 있다.



(a) 1레벨 수준: 물리 신호 정보

(b) 2레벨 수준: 1레벨 정보 + 가동/상태/통신 정보

(c) 3레벨 수준: 2레벨 정보 + 고장 정보

<오류로 인한 센싱 데이터 부분(왼쪽)과 동일한 이벤트 데이터 부분(오른쪽)>

그림 4 - 협동 로봇 블랙박스 데이터 생성 결과 분석

6. 결론 및 향후 연구

본 논문에서는 스마트팩토리 협동 로봇 고장 진단을 위한 요구사항 분석 기반 블랙박스를 제안하였다. 제안하는 방식은 협동 로봇의 고장 진단을 수행하기 위한 스마트팩토리의 제조 시나리오 및 데이터 생명주기에 따른 데이터 요구사항을 도출하고, 이를 소프트웨어 설계에 반영하여 블랙박스를 구현하는 방식이다. 본 연구에서는 요구사항 기반의 5가지 세부 모듈과 2가지 데이터 생성 체계를 설계하였으며, 기존의 진단 방식에서 수행하던 1레벨 수준(센싱 정보) 정보를 제공할 뿐만 아니라, 2레벨 수준(작업 정보) 및 3레벨 수준(고장 정보)의 정보를 제공함으로써 효과적인 데이터 기반의 고장 진단을 보조함을 검증하였다.

참고 문헌

[1] EL ZAATARI, Shirine, et al., Cobot programming for collaborative industrial tasks: An overview, Robotics and Autonomous Systems, 116, 162-180, 2019

[2] Lee, Gil-Yong, et al., Machine health management in smart factory: A review, Journal of Mechanical Science and Technology, 32.3, 987-1009, 2018

[3] Sakib, N., Wuest, T., Challenges and opportunities of condition-based predictive maintenance: a review, Procedia CIRP, 78, 267-272, 2018

[4] Telford, S., Mazhar, M. I., Howard, I., Condition based maintenance (CBM) in the oil and gas industry: An overview of methods and techniques, In Proceedings of the 2011 International Conference on Industrial Engineering and Operations Management, 2011

[5] Shin, J. H., Jun, H. B., On condition based maintenance policy, Journal of Computational Design and Engineering, 2(2), 119-127. 2015

[6] Peng, Y., Dong, M., Zuo, M. J., Current status of machine prognostics in condition-based maintenance: a review, The International Journal of Advanced Manufacturing Technology, 50(1-4), 297-313, 2010

[7] Jung, Woo-Kyun, et al., Smart sewing work measurement system using IoT-based power monitoring device and approximation algorithm, International Journal of Production Research, 58(20), 6202-6216, 2020

[8] Jardine, A. K., Lin, D., Banjevic, D., A review on machinery diagnostics and prognostics implementing condition-based maintenance, Mechanical systems and signal processing, 20(7), 1483-1510, 2006

[9] Kwon, Y. J., Chiou, R., Stepanskiy, L, Remote, condition-based maintenance for web-enabled robotic system, Robotics and computer-integrated manufacturing, 25(3), 552-559, 2009

[10] Chinnam, R. B., Baruah, P., Autonomous diagnostics and prognostics through competitive learning driven HMM-based clustering, Proceedings of the International Joint Conference on Neural Networks, IEEE, vol.4, 2466-2471, 2003

[11] Chen, Tingting, et al., Unsupervised Anomaly Detection of Industrial Robots Using Sliding-Window Convolutional Variational Autoencoder, IEEE Access 8, 47072-47081, 2020

[12] Cheng, Fangzhou, et al, High-accuracy unsupervised fault detection of industrial robots using current signal analysis, 2019 IEEE International Conference on Prognostics and Health Management, IEEE, 1-8, 2019.

[13] Cheng, F., Raghavan, A., Jung, D., Motion-insensitive features for condition-based maintenance of factory robots, U.S. Patent Application, 16/158, 175, 2020

[14] Olsson, E., Funk, P., Bengtsson, M., Fault diagnosis of industrial robots using acoustic signals and case-based reasoning, European Conference on Case-Based Reasoning. Springer, Berlin, Heidelberg, 686-701, 2004.

[15] Rahul, K., Banyal, R. K, Data Life Cycle Management in Big Data Analytics, Procedia Computer Science, 173, 364-371, 2020

협동 로봇의 예지보전 요구사항 기반 건전성 평가 기법

김진세[1(O)], 유동연[2], 박예슬[2], 이정원[1,2]
아주대학교 전자공학과[1], 아주대학교 AI융합네트워크학과[2]
jinsae913@gmail.com, dong0125@ajou.ac.kr,
yeseuly@gmail.com, jungwony@ajou.ac.kr

Health Assessment Method Based on the Predictive Maintenance Requirements of a Collaborative Robot

Jinse Kim[1(O)], Dong-Yeon Yoo[2], Ye-Seul Park[2], and Jung-Won Lee[1,2]
Department of Electrical and Computer Engineering, Ajou University[1]
Department of AI Convergence Network, Ajou University[2]

요 약

협동 로봇은 스마트팩토리의 핵심 설비로, 작업자가 쉽게 프로그래밍하여 다양한 작업을 수행할 수 있는 특징을 가진다. 따라서 일반적인 설비 예지보전 방법과는 달리 다양한 작업에 따라 생성되는 데이터의 복잡한 패턴을 반영할 수 있는 형태의 PHM 기술이 적용되어야 한다. 본 연구에서는 협동 로봇의 체계적인 예지보전을 위해 협동 로봇에서 수행될 수 있는 작업을 분류하고, 이에 따른 예지보전 요구사항을 도출하여 협동 로봇의 작업 처리 성능을 평가할 수 있는 건전성 평가 메트릭을 설계하였다. 제안하는 평가 메트릭을 협동 로봇의 TCP 위치 데이터 분석을 통한 자동차 매트 제조 프로그램 성능평가에 적용하여, 협동 로봇의 적재 하중을 초과하는 하중 부여를 통해 유발된 건전성 저하를 분석하였다. 하중 부여 전후의 무하중 동작 구간에 대하여 낮은 수준의 건전성을 지닌 구간이 최소 44.7%에서 최대 79.5% 증가하였으며, 그에 따른 건전성 평가 점수는 5점 기준으로 최소 1.07점에서 최대 2.2점 감소함을 확인하였다.

1. 서 론[1]

고장 진단 및 건전성 관리 기술(PHM: Prognostics and Health Management)은 센서를 이용하여 장비나 기계 시스템의 상태를 모니터링 하고 고장의 징후를 포착하는 진단기술(Diagnostics)로 설비의 상태기반 정비(CBM: Condition-Based Predictive Maintenance), 잔여유효수명(RUL: Remaining Useful Life)의 예측(Prognostics) 등에 사용된다. PHM 기술의 핵심적인 기능은 진단과 예측으로, 이 두 기능은 개발 방법과 기반지식은 유사하지만 목적성에 있어 차이가 있다. 진단은 현재 시점의 상황 또는 문제의 원인 및 특성을 분석하고 해결하기 위해 수행되며, 예측은 누적된 분석 결과 및 데이터를 기반으로 미래 시점의 상태나 발생 가능한 문제를 추정하기 위해 사용된다. 두 방향의 접근이 원활하게 이루어지는 효과적인 PHM 시스템의 경우, 구성 요소의 고장 진단을 통한 실시간 유지보수가 가능하며, 누적된 진단 결과의 분석을 통해 미래에 발생할 수 있는 고장을 미리 예측하여 예방할 수 있다. 또한, 유지보수 및 관리 일정을 최적화함으로써 예비 부품 및 필요한 자원에 대한 조달 시간을 확보하여 상당한 유지비용의 절감이 가능하다.

PHM 기술 적용이 필요한 분야는 고장 발생 시 치명적 손상으로 인한 잦은 예방 정비로 높은 시간비용을 지출하고 있거나 수리를 위한 접근이 제한되는 분야이며, 항공 산업, 자동차 산업, 원자력 산업 등이 그 예이다. 특히, 산업용 설비 기기는 결함이 발생하는 경우 심각한 설비 고장으로 이어질 수 있다. 이를 예방하기 위해 주기적으로 일괄 부품 교체를 시행하는 경우 과도한 보전이 이루어짐과 동시에 잦은 설비 정지로 인해 설비 가동률이 현저하게 낮아지게 된다. 따라서 산업용 설비 기기는 PHM을 기반으로 하는 예지보전 기술의 적용이 필수적이다. 산업 분야에서 PHM 기술이 적용되는 대상에는 작게는 부품 단위의 모터, 기어 등이 있으며[1-2], 크게는 설비 단위의 터빈, 로봇 등이 있다[3-4]. 이에 대한 예지보전 시 부품 및 설비의 상태 및 특성을 반영하는 데이터와 적절한 분석 기법을 바탕으로 일반화된 PHM 기술의 적용이 이루어지고 있다. 예시로 [5]에서는 제조 장치용 회전기에서 생성되는 전류, 진동 신호 및 온도 데이터에 T-평가, X^2-평가 등의 통계 분석 기법을 적용하여 상태 진단 및 수명 예측을 수행하였다.

[1] 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2020R1A2C1007400).

본 연구에서는 스마트팩토리의 핵심 설비 기기인 협동 로봇에 중점을 두었다. 협동 로봇은 접근제한영역을 설정한 상태에서 고속으로 운영되는 기존의 산업용 로봇과는 달리 인간과 작업 공간을 공유하며 물리적 상호작용이 가능한 로봇이다. 산업용 로봇과 비교하여 협동 로봇의 가장 큰 장점은 작업자와 함께 협력을 수행할 수 있도록 충돌 감지와 같은 위험 관리 모듈이 탑재되어 있고, 작업자가 쉽게 프로그래밍하여 공장 내 다양한 작업을 지시할 수 있다는 점이다. 그러나 이러한 장점으로 인해 협동 로봇에서 수집되는 데이터는 수행하는 작업에 따라 복잡한 패턴을 지니게 되며, 로봇의 이상성을 판단하기 위한 절대 기준이 존재하지 않기 때문에 이상 동작에 대한 기준 정립 없이 일반적인 PHM 기술을 적용하여 로봇의 건전성을 진단하기 어렵다. 따라서 협동 로봇의 건전성 평가는 작업의 다양성과 생성되는 데이터의 복잡성을 반영할 수 있는 형태의 PHM 기술이 적용되어야 한다. 협동 로봇은 수행하는 작업의 종류에 따라 관찰 가능한 다양한 데이터와 접목 가능한 기술을 기반으로 종합적으로 평가되어야 하며, 작업의 종류에 따라 수반되는 요구사항을 분석하여 체계적으로 PHM 기술을 적용할 수 있어야 한다.

따라서 본 연구에서는 협동 로봇의 예지보전 요구사항을 스마트팩토리 관점 및 기기 관점에서 도출하고 이를 협동 로봇이 수행하는 협업의 종류에 따라 연계한 작업 처리 성능 평가 기반의 건전성 평가 방법을 제안한다. 도출된 요구사항을 바탕으로 작업 인식 기반 작업의 정확도 및 완성도 평가를 진행하고, 평가 결과를 건전성 평가 지표로 사용하는 협동 로봇의 건전성 평가 메트릭을 설계한다. 이 때, 수집된 데이터의 통계 분석을 기반으로 한 작업 성능 레벨화(Level-wise)를 수행하여 건전성을 평가할 수 있도록 한다.

제안하는 건전성 평가 방법의 검증을 위해 협동 로봇의 엔드 이펙터 또는 도구의 위치로 정의되는 TCP(Tool Center Point) 위치 데이터 기반 협동 로봇 건전성 평가 방법을 설계하였으며, 이를 기반으로 협동 로봇의 적재 하중 이상의 하중 부여로 유발된 건전성 저하를 평가하였다. 평가 결과, 하중 부여 조건의 구동 전후로 건전성 점수가 감소함을 통해 협동 로봇의 건전성 평가 방법 및 평가 메트릭의 효용성을 검증하였다.

2. 관련 연구

일반적인 PHM 기술에 건전성 인자로 사용되는 데이터는 위치, 속도, 각도, 가속도, 진동, 토크, 전류 등이 있다. 이러한 데이터는 PHM 기술이 적용되는 장비의 내부 상태를 반영하여 물리적 특성을 제시할 수 있기 때문에 건전성 평가에 활발히 사용되고 있으며, 건전성 평가에 사용되는 데이터의 종류에 따른 적절한 평가 기법을 선정하여 적용하는 과정이 필요하다. 기존 연구에서는 건전성 평가 기법을 평가 모델링 방식에 따라 물리/수식 모델 기반 접근법과 데이터 기반 접근법으로, 데이터의 처리 방식에 따라 통계 기반, 학습 기반 접근법으로 구분하여 수행하고 있다.

물리/수식 모델 기반 건전성 평가 방식은 산업 시스템 및 설비 기기에 대한 시뮬레이션 모델을 사용하여 예지보전을 위한 건전성 평가를 시행하는 방식이다. 데이터 생성, 특징 추출, 기계 학습 모델 훈련, 예측 모델 개발 및 적용의 과정을 통해 장비의 물리적 특성을 반영한 예지 보전이 가능하다. [6]의 연구에서는 모듈식으로 재구성 가능한 형태의 로봇의 고장 진단을 위해 조인트 모듈 회로 모델을 기반으로 위치, 속도, 토크 데이터를 생성하여 건전성 평가를 수행하였다. 이러한 평가 방식은 적은 데이터로 건전성 평가가 가능하며 먼 미래의 고장 예지에 활용할 수 있다는 장점이 있지만, 확립된 모델이 많지 않아 적용 분야가 제한적이라는 단점이 있다.

반면, 데이터 기반 건전성 평가 방식은 데이터의 분포 및 경향성 분석을 기반으로 장비의 건전성을 평가하는 방식을 말한다. 이는 구체적으로 통계 기반 건전성 평가 방식과 기계학습 기반 건전성 평가 방식으로 구분된다.

통계 기반 건전성 평가 방식은 수집된 데이터를 활용하여 만들어진 데이터 분석 모델을 통해 건전성을 평가하는 방식을 말한다. 평가 과정에서 데이터에 통계 분석 기법을 적용하여 패턴을 추출하고 이를 데이터 분석 모델에 적용한다. [7]의 연구에서는 산업용 로봇의 고장 부품을 식별하기 위해 로봇의 진동 데이터를 활용하고 RMS(Root Mean Square)와 표준 편차를 기준으로 통계 분석하여 건전성을 진단하였다. 이러한 평가 방식은 적용이 쉽고 적용범위가 가장 넓다는 장점이 있지만, 고장 데이터가 매우 많아야 하며 동일한 사용 조건 및 고장의 종류에 대해서만 사용 가능하다는 단점이 있다.

기계 학습 기반 건전성 평가 방식은 주어진 데이터의 분석 결과에서 학습 가능한 규칙이나 새로운 지식을 자동적으로 추출해 기계가 학습하는 효과를 얻고, 이를 통해 건전성을 평가하는 방식을 말한다. [8]의 연구에서는 산업용 로봇의 고장 진단 및 예지를 위해 로봇에서 생성된 전류 및 각도 데이터를 활용하고, 이에 인공 신경망 모델 중 Seq2Seq (Sequence to Sequence) 네트워크 구조를 적용하여 건전성 진단을 수행하였다. 이러한 평가 방식은 물리 모델이 없어도 사용 가능하다는 장점이 있지만, 학습을 위한 많은 데이터가 필요하고, 근접 예측은 가능하나 먼 미래의 예측은 신뢰할 수 없다는 단점이 있다.

이러한 건전성 평가 방법 중 특정 방법을 선정하여 고정적으로 협동 로봇의 예지보전에 적용하는 데에는 한계가 존재하는데, 이는 협동 로봇이 수행하는 다양한

작업과 생성되는 데이터의 복잡한 패턴에 근거한다. 따라서 협동 로봇의 특성을 반영할 수 있도록, 협동 로봇이 수행하는 작업의 종류에 따른 협동 로봇의 예지보전 요구사항을 도출하고 이를 기반으로 하는 협동 로봇의 건전성 평가 방법을 제안한다.

3. 협동 로봇의 예지보전 요구사항 도출

본 장에서는 협동 로봇의 예지보전 요구사항을 도출한다. 먼저, 협동 로봇이 운용되는 스마트팩토리에 대한 요구사항 분석 관점을 정리하고, 분석 관점에 따른 예지보전 요구사항을 도출한다. 또한, 도출된 요구사항을 협동 로봇이 수행하는 다양한 작업의 종류와 연계한다.

3.1 스마트팩토리 요구사항 분석 관점

스마트팩토리 요구사항은 스마트팩토리 운영 시에 고려되어야 하는 전반적인 요구사항을 말하며, 사용자 관점의 요구사항과 시스템 관점의 요구사항으로 나누어진다. [9]의 연구에서는 두 관점에 대해 각각 6가지, 7가지 측면의 세부 분석 관점을 제시하였다. 따라서 본 절에서는 스마트팩토리 요구사항 도출을 위한 세부 분석 관점을 정의하고 이를 협동 로봇 예지보전 요구사항 도출에 적용하고자 한다.

3.1.1 사용자 요구사항 분석 관점

사용자 요구사항은 시스템이 사용자에게 제공해야 하는 서비스와 사용자가 준수해야 하는 제약사항에 대한 요구사항이다. 사용자 요구사항의 분석 관점에 대한 세부적인 정의는 표 1과 같다.

표 1. 사용자 요구사항 분석 관점

측면	정의
기능 (Fc)	센싱, 작동(Actuation), 제어, 통신, 물리적인 것 등의 기능에 관한 요구사항
사용성 (Us)	사람과 시스템 간의 상호작용에 관한 요구사항
신뢰성 (Tr)	보안, 개인정보 보호, 안전, 신뢰성, 복구가능성 등 시스템의 신뢰성에 관한 요구사항
시기 (Tm)	시간과 주파수 신호들의 생성과 전송, 지연관리 등을 포함하는 시스템에서 시간과 빈도(주파수)에 관련된 요구사항
데이터 (Dt)	퓨전, 메타데이터, 타입, 식별을 포함하는 데이터 상호작용에 관한 요구사항
조립 (Cp)	구성 요소의 속성에서 구성 요소 어셈블리의 선택된 속성을 계산하는 기능에 관한 요구사항

3.1.2 시스템 요구사항 분석 관점

시스템 요구사항은 시스템의 기능과 서비스, 동작 제약사항에 대해 개발자의 입장에서 서술한 요구사항이다. 시스템 요구사항의 분석 관점에 대한 세부적인 정의는 표 2와 같다.

표 2. 시스템 요구사항 분석 관점

특징	정의
시스템 확장성 (Sc)	시스템의 기능적·성능적·물리적 규모의 확장성 요구사항
융복합성 (Ca)	시스템 혹은 서브 시스템 간의 융복합성에 대한 요구사항
상호작용성 (Ia)	시스템 간 혹은 시스템과 인간 간의 유연한 상호작용성에 대한 요구사항
고신뢰성 (Dp)	신뢰성(Reliability), 회복성(Resilience), 유지보수성(Sustainability), 안전성(Safety) 및 보안성(Security) 등에 대한 요구사항
실시간성 (Rt)	시스템 운용 시 요구되는 실시간성(Realtime) 및 동기화(Synchronization) 기능에 대한 요구사항
상호운용성 (Io)	대규모 시스템에서 발생되는 이종성을 극복하기 위한 데이터의 상호 호환성 제공에 대한 요구사항
지능성 (It)	시스템의 지능형 서비스 제공을 위한 응용차원의 요구사항

3.2 협동 로봇의 예지보전 기기 요구사항 도출

본 절에서는 3.1절에서 정리한 스마트팩토리 요구사항 분석 관점 중 협동 로봇의 예지보전에 적용 가능한 관점을 적용하여 요구사항을 도출한다. 일반적으로 협동 로봇의 예지보전 시에는 기기 자체의 예지보전에 직접적으로 필요한 기기 요구사항과 예지보전의 수행 및 정상적인 공장 운영을 위해 충족되어야 하는 환경 요구사항이 고려된다. 특히, 본 논문에서는 협동 로봇의 건전성 평가 기법 설계에 반영하기 위하여 협동 로봇의 기기 관점 예지보전 요구사항에 초점을 두어 요구사항 분석을 진행하였다.

협동 로봇의 예지보전을 위한 기기 관점의 요구사항 도출은 3.1절의 스마트팩토리 요구사항 분석 관점 중 협동 로봇의 예지보전에 필수적이며 가장 기본적인 요구사항 분석 관점을 고려하여 수행하였다. 따라서 3.1.1절에서 제시한 사용자 요구사항 분석 관점 중 기능 및 사용성의 관점에서 사용자 요구사항을 도출하였으며, 3.1.2절에서 제시한 시스템 요구사항 분석 관점 중 융복합성, 상호작용성, 고신뢰성의 관점에서 시스템 요구사항을 도출하였다. 이에 대한 세부적인 요구사항 도출 결과는 표 3과 같다.

표 3. 협동 로봇의 예지보전 기기 요구사항

분류	측면	번호	요구사항
사용자	Fc	R-U1	스마트팩토리는 기기들의 작동 기능에 대하여 관리 가능
		R-U2	스마트팩토리 구성요소의 모델링과 시뮬레이션을 통하여 실제 상태를 확인할 수 있음
		R-U3	스마트팩토리의 기기들은 내부 상태에 대하여 인지할 수 있음
		R-U4	스마트팩토리의 기기들은 예측하지 못한 오작동 등이 발생할 수 있음
	Us	R-U5	스마트팩토리 기기들의 상태 정보들은 관리자가 원격 모니터링을 할 수 있도록 사용됨
시스템	Ca	R-S1	스마트팩토리를 구성하는 모듈의 상태 및 변동을 추적할 수 있어야 함
	Ia	R-S2	스마트팩토리의 기기들은 자신의 상태 정보를 관리 서버에 전달할 수 있어야 함
		R-S3	스마트팩토리의 기기들은 자신의 상태 정보를 표시할 수 있어야 함
		R-S4	스마트팩토리의 기기들은 이상 상황에 대하여 서버에 알림을 보낼 수 있어야 함
		R-S5	스마트팩토리의 기기들은 이상 상황에 대해 현장기술자에게 알림을 보낼 수 있어야 함
	Dp	R-S6	스마트팩토리의 구성요소의 상태를 모델링&시뮬레이션을 통해 예측할 수 있어야 함

표 3에서 제시한 요구사항을 PHM 기술 설계에 적용할 시에는 사용자 관점의 요구사항과 시스템 관점의 요구사항으로 분리되어 도출된 요구사항을 동일한 목적성을 가지고 포함관계에 있는 요구사항에 따라 상호 연결시켜 통합하는 과정이 필요하다. 이러한 과정을 통해 재정립한 요구사항은 표 4와 같다.

표 4. 협동 로봇의 예지보전 기기 요구사항 통합

번호	상호 연결	요구사항
C-1	R-U1 + R-S2 + R-S3	협동 로봇의 **작동 기능에 대하여 관리**가 가능하다.
C-2	R-U2 + R-S6	협동 로봇의 모델링과 시뮬레이션을 통하여 **실제 상태를 확인하고 예측**할 수 있다.
C-3	R-U3 + R-U4 + R-S1	협동 로봇은 **내부 상태에 대하여 인지하고 예측하지 못한 오작동을 감지**할 수 있다.
C-4	R-U5 + R-S4 + R-S5	협동 로봇은 이상 상황에 대하여 표시하거나 관리 서버 및 현장 기술자에게 **알림**을 보낼 수 있다.

3.3 협동 로봇 작업 분류에 따른 요구사항 연계

본 절에서는 협동 로봇에서 수행되는 작업의 분류와 이에 따른 요구사항의 연계 과정을 제시한다. 협동 로봇이 수행하는 작업의 종류에는 “공동 조작(Co-manipulation)”, “고정(Fixture)”, “이양(Handover)”, “조립(Assembly)”, “선별 및 운반(Pick-and-place)”, “전달(Fetch)”, “접합(Soldering)”, “조명(Illumination)”, “점검(Inspection)”, “교련(Drilling)”, “표면 마감(Surface Finish)”, “나사결합(Screwing)”이 있다[10].

작업 종류에 따른 협동 로봇의 예지보전 요구사항은 표 4의 4가지 요구사항을 구체화하는 방식으로 표 5과 같이 연계될 수 있다. 각각의 요구사항을 실제로 적용하기 위해 필요한 데이터 속성 (C-1: 프로그램 및 동작 정보, C-2: 작업 정확도 및 완성도, C-3: 물리 신호 정보, C-4: 이상 상황 정보)을 기반으로 분석하여 기술하였다.

이 때, 각각의 요구사항을 충족하면서 건전성 평가를

표 5. 협동 로봇의 작업 분류에 따른 요구사항 연계

요구사항 연계		공동 작업	고정	이양	조립	선별/운반	전달	접합	조명	점검	교련	표면 마감	나사 결합
C-1 분석		협동 로봇이 **실행되는 프로그램 및 동작 정보와 관련 통계 정보(가동 횟수, 시간 등)**의 관리가 가능하다.											
C-2 분석		협동 로봇의 **작업 정확도 및 완성도**를 확인하고 이를 기반으로 로봇의 상태 및 고장을 예측할 수 있다.											
C-3 분석		협동 로봇의 **물리 신호 데이터**를 통해 내부 상태를 인지하고 작업 오류를 감지할 수 있다.											
C-4 분석		감지된 **이상 상황에 대한 정보**를 표시하거나 알림을 보내 공동 작업자 및 관리자의 예지보전이 가능하다.											
세부요구사항	관련 데이터	변위, 가속도	상태, 동작	위치	위치, 시간	위치, 속도	위치, 시간	위치, 속도	거리	영상	위치, 속도	위치, 속도	위치
	적용 방법	물리 모델	MDP*	평균 신뢰도 점수	비용 함수	비용 함수	비전	비전	최소 거리	비전	DMP**	DMP	비전
	관련 연구	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[20]	[21]

* MDP: Markov Decision Process ** DMP: Dynamic Movement Primitives

수행하기 위해 작업의 종류에 따라 적절한 진단 데이터 및 적용 방법의 선정이 요구된다. 특히 C-2를 바탕으로 작업 정확도 및 완성도를 평가하기 위한 작업 별 세부적인 요구사항을 표 5의 하단에 개별적인 요구사항으로 기술하였다. 이는 실제 수행된 사례 연구 기반으로 작성하였으며, 각각의 작업 종류에 대한 관련 연구를 표 5의 하단에 명시하였다. 이와 같이 현재 작업의 정확도 및 완성도 평가에 사용되는 데이터와 적용 방법을 제시하여 기본적인 요구사항으로 고려되도록 하였으며, 이는 향후 추가적인 개발과 확장을 통해 고도화될 수 있다.

4. 협동 로봇의 건전성 평가 메트릭 설계

본 장에서는 협동 로봇의 건전성을 정의하고 이를 협동 로봇의 작업 수행의 정확도 및 완성도를 기반으로 평가하는 방법을 제안한다. 제안하는 방법은 협동 로봇의 예지보전 요구사항을 만족하는 구체화된 건전성 평가 기법으로, 협동 로봇이 수행하는 다양한 작업을 인식하고 작업 오류를 분석하여 협동 로봇의 건전성을 도출한다. 기존의 건전성 평가 기법은 설계된 테스트 프로그램을 구동해야만 건전성 평가를 수행할 수 있었음에 비해, 제안하는 방법은 실제 공장에서 작업자에 의해 설계된 프로그램을 수행하는 과정에서 데이터를 수집하고 작업 성능을 평가하기 때문에 기존의 방식보다 높은 활용도를 가진다.

4.1 협동 로봇의 건전성 정의

협동 로봇의 건전성이란 협동 로봇의 물리적/기능적 상태가 얼마나 정상적으로 의도한 작업을 수행할 수 있는 상태인지에 대한 정량적 평가 결과를 나타낸다. 협동 로봇은 수행하는 작업의 복잡도가 높기 때문에 건전성 진단을 수행할 때마다 다른 테스팅 프로그램을 구동할 시 건전성 저하를 발견하기 어렵다. 또한, 협동 로봇의 건전성 저하에 대한 평가 기준의 경우 판단하는 평가자에 따라 기준이 다르게 설정될 수 있다. 따라서 표 5의 예지보전 요구사항을 반영하여 건전성을 평가할 때, 통제 사항 확립을 통한 체계적인 적용이 필요하다. 본 논문에서는 통제 사항을 다음의 5가지로 정의하였다.

① **데이터 조건**: 동일한 특성(센서의 종류, 측정 단위, 수집 주기 등)을 가지는 데이터를 수집해야 한다.
② **동작 조건**: 동일한 동작의 수행을 기반으로 평가해야 한다.
③ **부하 조건**: 하중, 동작 패턴 등이 일정하게 부여된 상태에서 평가해야 한다.
④ **가동 조건**: 가동 시간이 동일한 상태에서 평가해야 한다.
⑤ **환경 조건**: 동일한 가동 환경(온도, 습도 등)에서 평가해야 한다.

이 때, ④, ⑤의 통제 사항은 평가 수행 환경과 관련된 통제사항으로 건전성 평가 방법 설계가 아닌 건전성 평가 방법의 적용 상황에서 고려되어야 하는 사항이다. 반면, ①, ②, ③의 경우, 건전성 평가 방법을 설계하는 과정에서 필요한 사항으로, 수행되는 작업을 기준으로 분류하여 분석한다면 동일한 통제 조건을 만족하는 상황에서 정밀한 건전성 평가를 수행할 수 있게 된다. 따라서 본 논문에서는 협동 로봇의 작업 처리 성능 평가를 통한 협동 로봇의 건전성 평가 기법을 제시한다.

4.2 협동 로봇 작업 수행 기반 건전성 평가 기법

본 절에서는 협동 로봇의 예지보전 요구사항 및 통제사항을 만족하는 구체화된 건전성 평가 방법으로 작업 처리 성능 평가를 기반으로 하는 건전성 평가 기법을 설계한다. 본 연구에서는 구체적으로 그림 1과 같이 작업 인식(Task Recognition), 작업 오류 분석(Task Failure Analysis), 건전성 평가(Health Assessment)의 3단계로 이루어진 건전성 평가 기법을 제안한다. 제안하는 건전성 평가 기법은 표 5에서 제시한 협동 로봇 예지보전 요구사항을 표 6과 같이 구체화하여 반영한다. 또한, 표 5의 세부 요구사항에서 제시하는 관련 데이터 및 적용 방법을 기반으로 건전성 평가를 수행하며, 단계별 세부 사항은 다음과 같다.

표 6. 건전성 평가 기법의 예지보전 요구사항 연계

번호	요구사항	구체화 방법
C-1	프로그램 및 동작 정보와 관련 통계 정보 관리	수집 데이터 + 작업인식
C-2	작업 정확도 및 완성도 확인 및 로봇의 상태, 고장 예측	평가기준 도출 + 작업성능 레벨화
C-3	협동 로봇의 내부 상태를 인지하고 작업 오류를 감지	수집 데이터 + 작업인식 + 평가기준 도출 + 작업성능 레벨화
C-4	이상 상황에 대한 정보 알림	건전성 평가

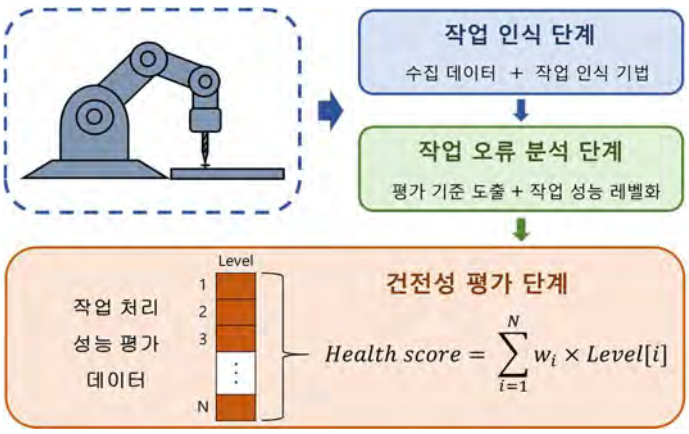


그림 1. 작업 수행 기반 건전성 평가 방법

① **작업 인식 단계**: 작업의 종류 및 환경에 따라 적절한 데이터 및 기법을 선정하여 적용하는 단계를 말한다. 이 단계에서는 다양한 방법(통계 기반 기법, 기계 학습 기반 기법 등)이 목적에 따라 사용될 수 있다. 3.3절에서 제시한 내용과 같이 12가지 협력 작업의 인식에 사용되는 데이터의 종류와 기법은 표 5의 하단과 같다.

② **작업 오류 분석 단계**: 이 단계에서는 작업 인식 방법을 통해 인식된 작업의 정확도 및 완성도를 평가하기 위해 작업 수행 성능 평가 기준을 도출하여 수행한 작업의 성능 수준을 분석한다. 정상 작업 수행 데이터에 대한 평균, 표준편차 등의 통계 분석 결과를 기준으로 설정하고, 작업 수행 데이터와의 차이를 기반으로 작업 성능 레벨을 도출하여 협동 로봇의 이상성을 분석한다. 이는 해당 작업의 정확도 및 완성도의 수준과 정상/비정상 동작 여부를 반영한다.

③ **건전성 평가 단계**: 작업 단위로 분석된 정확도 및 완성도의 수준을 종합하여 협동 로봇의 건전성을 반영하는 지표를 도출한다. 건전성은 각각의 성능 평가 데이터에 적절한 가중치를 곱한 값의 합으로 도출된다. 가중치의 값은 데이터의 처리 방법 및 건전성 도출의 목적에 따라 적절한 값으로 설정된다.

5. 협동 로봇 TCP 데이터 기반 건전성 저하 분석

본 장에서는 제시한 협동 로봇 작업 수행 기반 건전성 평가 방법에 대한 사례 연구로 협동 로봇의 말단 장치 또는 도구의 원점 위치로 정의되는 TCP 위치 데이터를 사용한 건전성 저하 분석 연구를 수행한다. 이와 같은 사례 연구는 표 6에서 분석된 내용을 기반으로 한다. 수행한 작업의 종류는 접합(Soldering)으로 터치 패널을 사용한 면적 도출을 통해 작업을 인식하였다. 또한, 작업의 성능 분석을 위해 면적 데이터와 통계 기반 분석 방법을 사용하였다. 이를 기반으로 협동 로봇의 건전성 저하를 분석하기 위한 3단계의 기법을 제시하였다. 각 단계는 그림 2와 같으며 세부적인 설명은 다음과 같다.

5.1 TCP 위치 데이터 기반 건전성 평가 방법

본 절에서 제안하는 TCP 데이터 기반 건전성 평가 방법은 협동 로봇의 TCP 위치 데이터를 수집하여 작업 단위로 수행된 프로그램 별로 작업 수행 성능을 평가하고, 평가 수준에 따라 건전성을 평가하는 방식이다.

① **작업 인식 단계**: TCP 위치 데이터는 이차원 좌표의 형태로 수집되며, 본 연구에서는 작업 단위로 수집된 TCP의 이동 경로 내부의 면적을 도출하여 이를 건전성 인자로 활용한다. 결과적으로 그림 2의 Step 1과 같이 작업 단위로 면적 데이터(예: T1=220, T2=240, T3=190, T4=245, T5=210)가 도출되며, 협동 로봇의 작업 수행 성능을 반영하는 데이터로 활용한다.

② **작업 오류 분석 단계**: 작업 단위 면적 데이터를 기반으로 작업 수행 성능 평가 기준을 도출하기 위하여 정상 동작 구간 데이터의 평균값을 계산하고, 이를 기준으로 데이터의 분포를 분석하여 작업 수행의 정확도 및 완성도의 수준을 분석한다. 정상 동작 구간의 데이터 중 평균값 μ에 가까운 25%, 50%, 75%, 100%의 데이터를 포함하는 범위에 대하여 평균값과의 편차가 각각 α, β, ɣ, δ일 때, 데이터 x에 대한 성능 평가 결과인 Level(x)의 도출 수식은 (수식 1)과 같다. 이에 대한 적용 예시는 그림 2의 Step 2와 같으며, 데이터와 평균값 사이의 편차의 수준에 따라 작업 성능 레벨(예: T1=Level3, T2=Level4, T3=Level5, T4=Level2, T5=Level4)이 도출된다.

③ **건전성 평가 단계**: 면적 데이터 단위로 도출된 N개의 성능 평가 레벨이 존재하는 데이터셋 내에서 5개의 성능 레벨에 대해 각각 a_i 개의 데이터가 level[i]로 평가될 때, 협동 로봇의 건전성 평가 점수는 (수식 2)에 근거하여 도출된다. 이러한 수식을 통해 데이터셋 단위의 성능평가 결과에 대한 평균 성능 수준을 도출하여 건전성 점수로 사용할 수 있다. 이는 구간별 레벨의 분포에 따라 1에서 5 사이의 값을 가지며, 점수가 낮을수록 건전성이 저하된 상태를 나타낸다. 분석 결과에

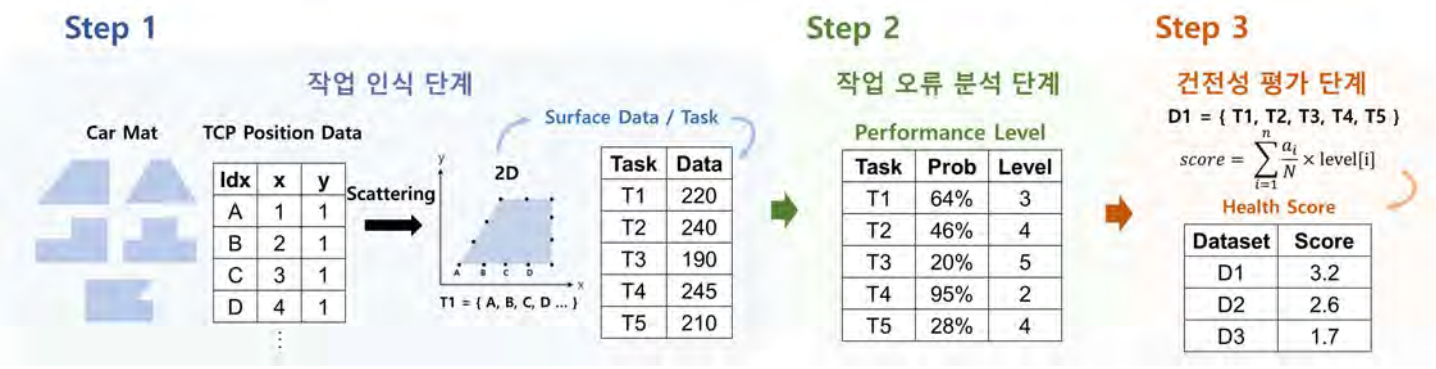


그림 2. 협동 로봇 TCP 데이터 기반 건전성 저하 분석 기법

대한 예시는 그림 2의 Step 3와 같으며, 작업 수행 성능 데이터의 집합(예: D1, D2, D3)에 대하여 건전성 점수(예: D1=3.2, D2=2.6, D3=1.7)가 도출된다.

$$level(x) = \begin{cases} 5 & if \ |x-\mu| \le \alpha \\ 4 & if \ \alpha < |x-\mu| \le \beta \\ 3 & if \ \beta < |x-\mu| \le \gamma \\ 2 & if \ \gamma < |x-\mu| \le \delta \\ 1 & if \ \delta < |x-\mu| \end{cases} \quad \text{(수식 1)}$$

$$score = \sum_{i=1}^{5} \frac{a_i}{N} \times level[i] \quad \text{(수식 2)}$$

5.2 부하로 인한 건전성 저하 분석

본 절에서는 5.1절에서 제시한 TCP 위치 데이터 기반 건전성 평가 기법을 검증하기 위하여 협동 로봇의 건전성 저하를 유발하고 이를 분석한다. 일반적으로 예지보전 기술에서 효용성을 평가하기 위해서는, 부품이나 기기 자체에 결함을 주입하고 해당 결함이 발생하는 패턴을 모델링하여 데이터를 기반으로 예측된 결함에 대한 정확도 분석을 수행한다. 본 연구에서는 결함이 발생할 수 있는 부하의 레벨을 다르게 주어, 각 레벨의 분포도를 기반으로 결함 상황 예측의 정확도를 평가하도록 한다.

협동 로봇의 건전성 저하는 일정 수준 이상의 하중을 부여하는 조건에서의 동작을 통해 유발된다. 본 실험에서는 이러한 특성을 반영하여 무하중 조건(Unload), 적재 하중 이하의 하중 부여 조건(Load), 적재 하중을 초과하는 하중 부여 조건(Overload), 무하중 구동을 재개하는 조건(Postload)의 4단계로 협동 로봇의 건전성 저하를 의도적으로 발생시켜 결함을 주입한다.

정상 구동 조건인 초기 두 단계(Unload, Load)에서 생성된 면적 데이터를 기반으로 작업 수행 성능 평가 기준을 도출하고, 이를 전체 데이터에 적용하여 데이터 단위 작업 수행 성능 수준을 도출한다. 그림 3은 설계된 5개의 테스트 프로그램에 따라 도출된 성능 평가 결과를 나타내며, 붉은색의 선은 하중 조건에 따른 단계의 구분을 나타낸다. 건전성 저하 분석을 위해 표 7과 같이 단계별 성능 평가 데이터 중 낮은 수행성능 레벨(Level1-2)을 나타내는 데이터의 비율을 분석하고, 표 8과 같이 단계별 성능평가 데이터에 (수식 2)를 적용하여 건전성 점수를 도출하였다. 전반적으로 단계에 따라 건전성 점수가 감소하는 양상을 보였으며, 특정 프로그램(P1, P5)의 경우, 적재 하중 이하의 하중 부여(Load) 시 복잡한 작업 수행의 안정성 증가에 따른 작업 성능 향상에 의해 점수가 증가하였다. 또한, 적재하중을 초과하는 하중 부여(Overload) 시 과부하로 인해 모든 프로그램에 대해 건전성 점수가 감소하였다. 건전성 저하 분석을 위해 하중 부여 전후(Unload, Postload)를 비교하였을 때, 낮은 수준의 성능을 지닌 구간이 최소 44.7%에서 최대 79.5% 증가하였고 건전성 점수가 최소 1.07점에서 최대 2.2점 감소하였다. 이를 통해 협동 로봇의 건전성 저하를 확인하였으며, 제안하는 건전성 평가 기법의 적절성을 검증하였다.

표 7. 단계별 Level 1-2 성능 평가 데이터 분포

	Unload	Load	Overload	Postload
P1	50%	0%	100%	100%
P2	30%	20%	100%	100%
P3	10%	40%	88.9%	89.5%
P4	25%	25%	50%	89.5%
P5	50%	0%	95%	94.7%

표 8. 작업 프로그램 단위 건전성 점수

	Unload	Load	Overload	Postload
P1	3.2	3.8	1.1	1.0
P2	3.6	3.4	1.85	0.75
P3	3.95	3.05	2.16	2.0
P4	3.7	3.3	2.75	1.79
P5	2.8	4.2	1.65	1.73

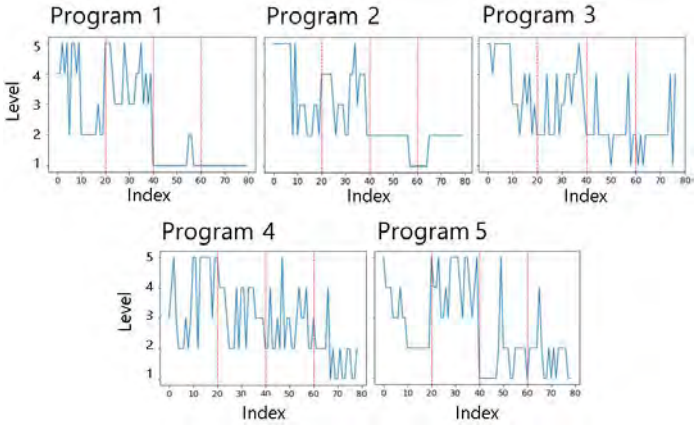


그림 3. 작업 수행 성능 평가 결과 그래프

6. 결론 및 향후 연구

본 논문에서는 협동 로봇이 수행하는 작업의 종류에 따른 예지보전 요구사항을 도출하고 이를 기반으로 협동 로봇의 작업 처리 관점의 건전성 평가 기법을 제안하였다. 이는 복잡한 작업 패턴을 지닌 협동 로봇의 동작을 통계적으로 분석하여 협동 로봇의 건전성 평가를 지원한다. 이를 협동 로봇의 TCP 위치 데이터 기반의 건전성 저하 분석에 적용하여 적재 하중을 초과하는 하중 부여로 인해 유발된 협동 로봇의 건전성 저하를 확인하였으며, 평가 기법의 적절성을 검증하였다. 향후 연구에서는 다양한 협동 로봇의 작업 및 추가적인 데이터 종류를 선정하여 건전성 저하 분석 실험을

진행하고, 조인트 단위의 세부적인 건전성 평가가 가능하도록 평가 기법을 확장 적용 및 검증하여 사례연구를 다양하게 구성할 예정이다.

참고 문헌

[1] Taewan Hwang, Keunsu Kim, Sujii Kim, Byungjoo Jeon, Byeng D. Youn, " Health index extraction for ball bearing using defect frequency" The Korean Society of Mechanical Engineers, vol. 2016, No. 12

[2] Ju Seong Shin, Ju Hyun Kim, Jong Geol Kim, Maolin Jin, " Development of a Lifetime Test Bench for Robot Reducers for Fault Diagnosis and Failure Prognostics" The Korea Fluid Power Systems Society, vol.16, No. 3, pp. 33-41, 2019.

[3] Yunhan Kim, Jungho Park, Jong Moon Ha, Byeng Dong Youn, Jin-Gyun Park, " Robust Fault Detection Method of an Industrial Robot under Various Operating Conditions" The Korean Society of Mechanical Engineers, vol. 2017, No. 11,

[4] Seung Jun Back, Young Kap Son, Jun Hee Kim, Jong Cheol Lee, " Reliability Assessment of a Remote Operation Robot for Excavators" The Korean Society of Mechanical Engineers, vol. 2013, No. 3, pp. 245-246, 2013.

[5] Samata, Shuichi, et al. "System for predicting life of a rotary machine, method for predicting life of a manufacturing apparatus which uses a rotary machine and a manufacturing apparatus." U.S. Patent No. 6,898,551. 24 May 2005.

[6] J. Yuan, etc, "Power Efficiency Estimation-Based Health Monitoring and Fault Detection of Modular and Reconfigurable Robot," in IEEE Transactions on Industrial Electronics, vol. 58, No. 10, pp. 4880-4887, 2011.

[7] Algburi, R.N.A.; Gao, H. “Health Assessment and Fault Detection System for an Industrial Robot Using the Rotary Encoder Signal”. *Energies, 12,* 2816, 2019.

[8] Yeong-Hyeon Lee, Kyung-Jun Kim, Seung-Ik Lee, Dong-Ju Kim, “Seq2Seq model-based Prognostics and Health Management of Robot Arm”, Korea Information Electron Communication Technology, Vol.12, No.3, pp. 242-250, 2019.

[9] Hoe Seung Chin, Yeong-Hui Seo, Won-Tae Kim, “Functional, non-functional Software requirements analysis of Cyber Physical System environment,” in Software Policy & Research Institute, 2018, pp. 69-74

[10] El Zaatari, Shirine, et al. "Cobot programming for collaborative industrial tasks: An overview." Robotics and Autonomous Systems 116, pp. 162-180, 2019

[11] S. Lichiardopol, N. van de Wouw, and H. Nijmeijer, “Control scheme for human-robot co-manipulation of uncertain, time-varying loads,” in 2009 American Control Conference, 2009, pp. 1485-1490.

[12] S. Nikolaidis, P. Lasota, R. Ramakrishnan, and J. Shah, “Improved human-robot team performance through cross-training, an approach inspired by human team training practices,” International Journal of Robotics Research, vol. 34, no. 14, pp. 1711-1730, 2015.

[13] C.-M. Huang, M. Cakmak, and B. Mutlu, “Adaptive coordination strategies for human-robot handovers.” in Robotics: Science and Systems, 2015.

[14] L. Johannsmeier and S. Haddadin, “A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes,” IEEE Robotics and Automation Letters, vol. 2, no. 1, pp. 41-48, 2017.

[15] V. Gabler, T. Stahl, G. Huber, O. Oguz, and D. Wollherr, “A game theoretic approach for adaptive action selection in close proximity human robot collaboration,”, in IEEE International Conference on Robotics and Automation, 2017.

[16] G. Maeda, A. Maloo, M. Ewerton, R. Lioutikov, and J. Peters, “Anticipative interaction primitives for human-robot collaboration,” in 2016 AAAI Fall Symposium Series, 2016

[17] M. Wongphati, H. Osawa, and M. Imai, “Gestures for manually controlling a helping hand robot,” International Journal of Social Robotics, vol. 7, no. 5, pp. 731-742, 2015.

[18] C. Lenz, M. Rickert, G. Panin, and A. Knoll, “Constraint task-based control in industrial settings,” in 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 3058-3063.

[19] I. E. Makrini, K. Merckaert, D. Lefeber, and B. Vanderborght, “Design of a collaborative architecture for human-robot assembly tasks,” in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 1624-1629.

[20] K. R. Guerin, S. D. Riedel, J. Bohren, and G. D. Hager, “Adjutant: A framework for flexible human-machine collaborative systems,” in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 1392-1399.

[21] A. Cherubini, R. Passama, P. Fraisse, and A. Crosnier, “A unified multimodal control framework for human-robot interaction,” Robotics and Autonomous Systems, vol. 70, pp. 106-115, 2015.