

한국정보과학회
Korean Society of Information Science and Technology

제 24 권 제 1 호
Vol. 24 No. 1



2022



제 24 회 한국 소프트웨어공학 학술대회 논문집

Proceedings of the 24th Korea Conference on
Software Engineering (KCSE 2022)

- 일시: 2022년 1월 19일(수) ~ 1월 21일(금)
- 장소: 강원도 평창 한화리조트(휘닉스파크점)

주최: 한국정보과학회, 한국정보처리학회

주관: 한국정보과학회 소프트웨어공학 소사이어티
한국정보처리학회 소프트웨어공학연구회

후원:  한국전자통신연구원
Electronics and Telecommunication Research Institute,  (주)비트컴퓨터,
한국소프트웨어기술진흥협회(KOSTA), T3Q(주),
(주)유정인베스트먼트, (주)다한테크, (주)브이웨이,
브이플러스랩(주), 슈어소프트테크(주), (주)이에스지,
(주)코어다트랩, STA 테스트컨설팅(주),
TTA 소프트웨어시험인증연구소,
서강대 지능형블록체인연구센터,
KAIST 빅데이터 엣지 클라우드서비스 연구센터

초대의 글

소프트웨어공학 학술대회(KCSE 2022) 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 1999년부터 매년 개최하는 학술대회입니다.

이번 제 24 회 학술대회는 “신뢰할 수 있는 인공지능을 위한 소프트웨어공학 기술”을 주제로, 기조 연설, 튜토리얼, 신진 연구자 발표, 우수 논문 발표 등의 초청 세션과 소프트웨어공학 분야의 각계에서 제출한 45 편의 엄선된 논문으로 구성하였으며, 2022년 1월 19일부터 3일간에 걸쳐 진행하게 되었습니다.

이번 학술대회는 오프라인 중심, 온라인 병행으로 진행하고자 계획하고 있습니다만 추후 온라인 중심, 오프라인 병행으로 변경될 수 있음을 말씀드립니다. 온라인이든 오프라인이든 소통 방식과는 무관하게 이제는 시간과 공간의 한계를 넘어서는 교류의 시대가 되었습니다. 이번 KCSE 2022 학술대회가 소프트웨어공학을 연구하고, 적용하는 모든 연구자 그리고 전문가 여러분께 즐겁고 활기찬 학술 교류 및 기술 협력의 장이 될 수 있도록 철저한 방역과 다양한 프로그램으로 진행하고자 하오니 여러분의 많은 관심과 참여를 부탁드립니다.

제 24 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: 이정원 교수(아주대)

조직위원: 고인영 교수(KAIST), 김순태 교수(전북대), 김정아 교수(가톨릭관동대),
 류덕산 교수(전북대), 백종문 교수(KAIST), 유준범 교수(건국대),
 이선아 교수(경상대), 이찬근 교수(중앙대), 한종대 교수(상명대),
 정효택 박사(ETRI), 민상윤 대표(솔루션링크),
 전진욱 사장(비트컴퓨터), 박병훈 대표(T3Q),
 박준성 회장(한국소프트웨어기술진흥협회)

학술위원장: 이주용 교수(UNIST)

학술위원: 김동선 교수(경북대), 김문주 교수(KAIST), 김미정 교수(UNIST),
 김윤호 교수(한양대), 김진대 교수(서울과학기술대),
 김진현 교수(경상대), 김태호 박사(ETRI), 김택수 박사(삼성전자),
 남재창 교수(한동대), 류덕산 교수(전북대), 박수진 교수(서강대),
 배경민 교수(POSTECH), 서영석 교수(영남대), 오학주 교수(고려대),
 유신 교수(KAIST), 윤희진 교수(협성대), 이우석 교수(한양대), 이우진 교수(경북대),
 정우성 교수(서울교대), 지은경 교수(KAIST), 채흥석 교수(부산대),
 최윤자 교수(경북대), 홍신 교수(한동대)

문의사항 연락처

학술대회 홈페이지 : <http://www.sigsoft.or.kr/KCSE2022/>

온라인 학술대회 홈페이지: <http://kcse2022.userinsight.co.kr/>

조 직 : 이정원 교수 (Email: jungwony@ajou.ac.kr, Tel. 031-219-1959)

학 술 : 이주용 교수 (Email: jooyong@unist.ac.kr, Tel. 052-217-2123)

KCSE 2022 프로그램

1 월 19 일 (수)				
시간	행 사 내 용			
12:00-13:00	KCSE 2022 등록			
	튜토리얼 T1 좌장: 남재창 (한동대) 장소: 그랜드홀 2	튜토리얼 T2 좌장: 유준범 (건국대) 장소: 세미나실 1	튜토리얼 T3 좌장: 이병정 (서울시립대) 장소: 세미나실 2	워크숍 W1
13:00-14:30 (90 분)	프로그램 자동 수정: 기초와 원리, 그리고 패턴 추출 김동선 교수 (경북대)	Challenges on System IP Design Automation Software 박창규 연구원 (삼성전자)	신뢰성 있는 인공지능 품질평가방안 및 사례 이승은 선임 (와이즈스톤)	KAIST 빅데이터 옛지 클라우드 서비스 연구센터 장소: 세미나실 4 (14:00-16:30)
14:30-14:40	휴식			
	튜토리얼 T4 좌장: 유신 (KAIST) 장소: 그랜드홀 2	튜토리얼 T5 좌장: 이준영 (CryptoLab) 장소: 세미나실 1	튜토리얼 T6 좌장: 이병정 (서울시립대) 장소: 세미나실 2	
14:40-16:10 (90 분)	프로그램 합성 소개 이우석 교수 (한양대)	Towards Secure Interoperation in Android 황성재 교수 (성균관대)	소프트웨어 개발에서의 프로그래밍 심리학 이재용 교수 (한서대)	
16:10-16:20	휴식			
	개회식 장소: 그랜드홀 2 사회: 이정원 조직위원장 (아주대)			
16:20-16:40 (20 분)	개회사: 홍장의 회장 (한국정보과학회 소프트웨어공학 소사이어티) 김정아 운영위원장 (정보처리학회 소프트웨어공학연구회)			
	기조강연 1 장소: 그랜드홀 2 사회: 이주용 학술위원장 (UNIST)			
16:40-17:30 (50 분)	SE 관련 교육과 인재양성 경기대학교 권기현 교수			
17:30-17:40	휴식			
	신진 연구자 초청 세미나 N1 좌장: 김진대 (서울과기대) 장소: 그랜드홀 2	신진 연구자 초청 세미나 N2 좌장: 박창규 (삼성전자) 장소: 세미나실 1	신진 연구자 초청 세미나 N3 좌장: 이우석 (한양대) 장소: 세미나실 2	
17:40-18:30 (50 분)	머신러닝 소프트웨어를 위한 자동화 테스트 기술 김미정 교수 (UNIST)	SW 가시화를 위한 메타모델과 모델변환 손현승 교수 (목포대)	A Robust Foundation for the Correctness of LLVM Compiler 이준영 박사 (CryptoLab)	
18:30-19:30	석 식			

1 월 20 일 (목)	
시 간	행 사 내 용
	논문 발표
	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p style="text-align: center;">A1: 결함 분석 및 예측 좌장: 류덕산 (전북대) 장소: 그랜드홀 2</p> </div> <div style="width: 45%;"> <p style="text-align: center;">A2: 보안 좌장: 고인영 (KAIST) 장소: 세미나실 1</p> </div> </div>
09:30-10:50 (80 분)	<div style="display: flex;"> <div style="width: 50%; padding-right: 10px;"> <p>[최우수 일반 논문] 사전 학습 언어 모델들을 활용한 버그 담당자 자동 할당 성능 향상 왕대성, 성훈, 김민하, 이찬근 (중앙대)</p> <p>[일반 논문] 해석 가능한 시계열 이상탐지 모델을 활용한 생산 능력 계획 수립 방안 강민정, 이상민 (광운대)</p> <p>[최우수 일반 논문] 오픈 소스 기계학습 어플리케이션에 대한 결함 사례 조사 최윤호, 이창공, 방제형, 남재창 (한동대)</p> <p>[학부생 논문] 양방향 메트릭 선택 기반 비지도 학습 결함 예측 백인선, 권수진, 박성수, 박수은, 남재창 (한동대)</p> </div> <div style="width: 50%;"> <p>[우수 단편 논문] 이더리움 스마트 컨트랙트의 업그레이드 전략에 따른 가스 영향도분석 김진영, 김미수, 이은석 (성균관대)</p> <p>[초청 논문] JEST: N+1-version Differential Testing of Both JavaScript Engines and Specification (ICSE 2021) Jihyeok Park, Seungmin An, Dongjun Youn, Gyeongwon Kim, Sukyoung Ryu (KAIST)</p> <p>[학부생 논문] 생체 정보의 비식별화를 위한 지문 탐지 모델 배은기, 김한동, 권동균, 한종대 (상명대)</p> <p>[우수 학부생 논문] 텍스트 기반의 CAPTCHA 보완 모델 고유라, 김태호, 한종대 (상명대)</p> <p>[학부생 논문] 기업형 블록체인 플랫폼인 하이퍼레저 베수의 기술적 배경 임영택, 박민하 (동국대), 주성호, 배병일 (한전)</p> </div> </div>
10:50-11:00	휴식

	B1: CPS 좌장: 김진현 (경상대) 장소: 그랜드홀 2	B2: 결함위치 추정 및 결함예측 좌장: 남재창 (한동대) 장소: 세미나실 1
11:00-12:00 (60 분)	<p>[최우수 일반 논문] 협동 로봇의 센싱 데이터 특징 기반 테스트 데이터셋 구성을 통한 학습 모델 신뢰성 분석 김진세, 배수빈, 박예슬, 이정원 (아주대)</p> <p>[초청 논문] Efficient SMT-Based Model Checking for Signal Temporal Logic (ASE 2021) Jia Lee, Geunyeol Yu, Kyungmin Bae (POSTECH)</p> <p>[단편 논문] 히스토리 기반 타임뱅크 봉사 파트너 추천 알고리즘 선다영 (선문대), 손서락 (타임뱅크코리아), 양혜란 (타임뱅크코리아), 유원상 (선문대)</p>	<p>[초청 논문] HOTFUZ: cost-effective higher-order mutation-based fault localization (STVR) Jong-In Jang (KAIST), Duksan Ryu (전북대), Jongmoon Baik (KAIST)</p> <p>[초청 논문] Ahead of Time Mutation Based Fault Localisation using Statistical Inference (ISSRE 2021) Jinhan Kim, Gabin An (KAIST), Robert Feldt (Chalmers 공대), Shin Yoo (KAIST)</p> <p>[일반 논문] 소프트웨어 결함 예측을 위한 설명 가능한 인공지능 적용 최지원, 이재욱, 류덕산, 김순태 (전북대)</p>
12:00-13:40	중식	
	C1: 프로그램 이해 및 분석 좌장: 이주용 (UNIST) 장소: 그랜드홀 2	C2: AI와 SE I 좌장: 김진현 (경상대) 장소: 세미나실 1
13:40-15:00 (80 분)	<p>[최우수 일반 논문] 코드 변경에 대한 검출 기법과 사람의 묘사 차이 연구 김무준, 김범철, 김진대 (서울과학기술대)</p> <p>[초청 논문] JSTAR: JavaScript Specification Type Analyzer using Refinement (ASE 2021) Jihyeok Park, Seungmin An, Shin Wonho, Yusung Sim, Sukyoung Ryu (KAIST)</p> <p>[초청 논문] Observation-based approximate dependency modeling and its use for program slicing (JSS) Seongmin Lee, David Binkley (Loyola 대학), Robert Feldt (Chalmers 공대), Nicolas Gold (UCL), Shin Yoo (KAIST)</p> <p>[단편 논문] FPGA 네트워크 기능 가속기를 지원하기 위한 빠르고 효율적인 소프트웨어 스택 부준혁, 김장우 (서울대)</p>	<p>[단편 논문] 딥 러닝을 활용한 메소드와 주석 간의 적합성 검증 모델 박재현, 이동건, 장용훈, 서영석 (영남대)</p> <p>[초청 논문] SymTuner: Maximizing the Power of Symbolic Execution by Adaptively Tuning External Parameters (ICSE 2022) Sooyoung Cha, Myungho Lee, Seokhyun Lee, and Hakjoo Oh (고려대)</p> <p>[학부생 논문] 개인정보보호를 위한 SGX 기반 학습데이터 비식별화 인공지능 학습 프레임워크 김남령, 옥지원, 김성민 (성신여대)</p> <p>[단편 논문] 딥러닝 기반 얼굴인식 손실 함수 기술 동향 임현택, 김태윤, 김창석 (고스트페이스)</p> <p>[단편 논문] 경량 뉴럴 네트워크 탐색을 위한 가지치기, 지식증류, 그리고 양자화에 기반한 하이브리드 압축 방법 이재원, 이상민 (광운대)</p>

15:00-15:10	휴식		
	D1: 프로그램 수정 좌장: 김진대 (서울과학기술대) 장소: 그랜드홀 2	D2: AI 와 SE II 좌장: 류덕산 (전북대) 장소: 세미나실 1	D3: SW 테스트 및 품질관리 좌장: 지은경 (KAIST) 장소: 세미나실 2
15:10-16:30 (80 분)	<p>[초청 논문] NPEX: Repairing Java Null Pointer Exceptions without Tests (ICSE 2022) Junhee Lee, Seongjoon Hong, Hakjoo Oh (고려대)</p> <p>[우수 단편 논문] 의심스러운 조건문을 이용한 결함 위치 식별 및 수정 개선 강성민, 유신 (KAIST)</p> <p>[단편 논문] 멀티 체크 버그 정정을 위한 버그 블럭과 CodeBERT 활용 김지성, 호혜민 (서울시립대), 양근석 (경남대), 이병정 (서울시립대)</p> <p>[단편 논문] 복잡한 버그의 템플릿 생성을 위한 Change Action 측면에서의 과거 수정 이력 분석 허진석, 이은석 (성균관대)</p> <p>[우수 학부생 논문] Kubernetes 를 활용한 멀티 클라우드 모니터링 솔루션 박신영, 김병권, 배유진, 심석인 (동국대)</p>	<p>[일반 논문] 홍부 방사선 영상 학습 성능 개선을 위한 오픈 데이터셋 품질 평가 기법 김양곤, 박예슬, 선주성, 이정원 (아주대)</p> <p>[단편 논문] 그래프 합성곱 신경망을 활용한 취약점 탐지 벤치마크 최유라, 권영우 (경북대)</p> <p>[단편 논문] Keypoint-RCNN 을 이용한 반려견 대퇴골 탈구인식을 위한 수의영상판독 시스템 김민경, 김지인 (건국대)</p> <p>[학부생 논문] 수업 질문 자동응답을 위해 KoBert 를 이용한 챗봇과 Doc2Vec 을 이용한 챗봇과의 성능비교 허주은, 이선아 (경상대)</p> <p>[학부생 논문] 다중 피부병변 검출 심층학습을 위한 피부병변 합성 기반 데이터 증강 기법 박범진, 이영찬 (선문대), 장현재 (㈜에프앤디파트너스), 유원상 (선문대)</p>	<p>[일반논문] Actor-Critic 계열 강화학습 알고리즘의 행동 정확성 평가 방법 김영재, 홍장의 (충북대)</p> <p>[우수 학부생 논문] CFG 생성 타당성 검사를 위한 테스트 자동화 연구 정소희, 김동우, 최윤자 (경북대)</p> <p>[단편 논문] 모바일 어플리케이션 자동화 테스트를 위한 딥러닝 기반 UI 경로 지도의 생성 방혜원, 박정민, 한영준 (숭실대)</p> <p>[단편 논문] Concolic 테스팅과 Fuzzing 을 결합한 유닛테스팅 자동화 기술 김윤삼 (브이플러스랩), 김문주 (브이플러스랩/ KAIST)</p> <p>[단편 논문] 동영상 스트리밍 서비스에 HTTP/1.1, HTTP/2 프로토콜 적용별 프로토콜 성능 평가 도지영, 이성원 (경희대)</p>
16:30-16:40	휴식		
16:40-17:30 (50 분)	<p>기조강연 2 장소: 그랜드홀 2</p> <p style="text-align: center;">Software Defined Weapon - Open Innovation 을 통한 신뢰성있는 자율화 무기체계의 개발 LIG 넥스원 김동환 연구위원</p>		<p>사회: 홍장의 대회장 (충북대)</p>
17:30-18:00	<p>우수논문상, 공로상, 감사장 수여식 장소: 그랜드홀 2</p>		<p>사회: 이정원 조직위원장 (아주대) & 이주용 학술위원장 (UNIST)</p>
18:10-20:00	석식		

1 월 21 일 (금)				
시 간	행 사 내 용			
	논문 발표			워크숍 W2:
	E1: 요구사항 및 아키텍처 I 좌장: 고인영 (KAIST) 장소: 그랜드홀 2	E2: 요구사항 및 아키텍처 II 좌장: 유준범 (건국대) 장소: 세미나실 1	E3: SW 모델링 및 테스트 좌장: 이주용 (UNIST) 장소: 세미나실 2	
9:25-10:50 (85 분)	<p>[우수 일반 논문] 소프트웨어 요구사항 분류 모델의 성능 향상을 위한 불균형 데이터 처리에 관한 연구 최종우, 이영준, 임채균, 최호진 (KAIST)</p> <p>[우수 일반 논문] 효율적인 아키텍처 패턴 적용을 위한 소프트웨어 요구사항 분류체계에 관한 연구 최종우, 민상윤 (KAIST)</p> <p>[단편 논문] 코드가시화의 고도화 연구 김장환, 문소영, 김영철 (홍익대)</p> <p>[단편 논문] 자카드 지수를 이용한 한국어 요구사항 유사도 분석 자동화 장우성, 문소영, 김영철 (홍익대)</p> <p>[단편 논문] Use case 명세서 기반 자동 가이드 코드 생성 정세준, 문소영, 김영철 (홍익대)</p>	<p>[일반 논문] Status Quo of Software Architecture and Technologies for Cloud-Edge Collaborative Intelligence Xiangchi Song, 고인영 (KAIST)</p> <p>[우수 일반 논문] AIAG-VDA FMEA 방법론 기반 SFMEA : 효과적인 자동차 SW 안전분석 김소연, 권기현 (경기대)</p> <p>[일반 논문] 엣지 클라우드 환경에서 다목적 최적화 알고리즘을 활용하여 비용 및 서비스 지연을 최소화하는 마이크로서비스 배포 방법 김민협, 고인영 (KAIST)</p> <p>[산업체 논문] 소프트웨어 수출을 위한 시장주도 요구공학 적용 사례 이종길, 길문주, 오정섭 (엔에스이)</p>	<p>[일반 논문] 다양한 디스플레이 환경을 위한 적응형 UI 레이아웃 프레임워크 윤진, 이은영, 이상현 (삼성전자), 백종문 (KAIST)</p> <p>[단편 논문] 멀티태스킹 PLC 프로그램의 테스트를 위한 주기 별 테스트 시퀀스 실행 제어 방법 은형석, Lingjun Liu, 지은경, 배두환 (KAIST)</p> <p>[산업체 논문] 반도체 공정 유지 보수 비용 절감을 위한 공정 레시피 데이터 학습 및 장비 결함 검출 최민호(피에스케이 / 아주대), 유동연, 이정원 (아주대)</p> <p>[학부생 논문] 아동·청소년을 위한 인공지능 스피커: 심리 상담 통합 시스템 개발 김수아, 이예빈, 정수민, 최미소, 최혜빈 (이화여대)</p> <p>[학부생 논문] 스마트 팩토리-가스라이터 제조 공정에서의 스티커 부착과 관련된 불량 검출 프로그램 개선 신관수, 송호준, 조영진, 신연순 (동국대)</p>	<p>서강대 지능형블록체인 연구센터</p> <p>장소: 세미나실 4 (10:00~11:50)</p>
10:50-11:00	휴식			
	신진 연구자 초청 세미나 N4 좌장: 김미정 (UNIST) 장소: 그랜드홀 2	신진 연구자 초청 세미나 N5 좌장: 유신 (KAIST) 장소: 세미나실 1	신진 연구자 초청 세미나 N6 좌장: 지은경 (KAIST) 장소: 세미나실 2	
11:00 -11:50 (50 분)	오픈소스 소프트웨어에서의 버그 정정 관리를 위한 지능형 프레임워크 양근석 교수 (경남대)	비형식 요구사항 기반 유스케이스 추출을 통한 노력 추정 방법 박보경 교수 (진주교대)	소프트웨어 신뢰도 보증을 위한 효율적인 도달 가능성 분석 이낙원 박사 (KAIST)	
11:50-12:10 (20 분)	폐회식 장소: 그랜드홀 2 사회: 이정원 조직위원장 (아주대)			

KCSE 2022 튜토리얼

튜토리얼 T1: 프로그램 자동 수정: 기초와 원리, 그리고 패턴 추출

- ◆ 일시: 1월 19일(수) 13:00~14:30
- ◆ 장소: 그랜드홀 2
- ◆ 제목: 프로그램 자동 수정: 기초와 원리, 그리고 패턴 추출
- ◆ 연사: 김동선 교수 (경북대학교 컴퓨터공학부)
- ◆ 튜토리얼 초록:

소프트웨어는 필연적으로 다양한 형태의 오류를 내재하고 있다. 오류는 대상 소프트웨어의 크기, 개발단계, 복잡도 등에 따라서 조금씩 다른 특성을 가지지만, 결과적으로 개발 비용을 증가시키고, 소프트웨어의 신뢰성을 낮추는 등 많은 악영향을 미치게 된다. 자동화된 오류 위치 식별 연구, 오류 우선 순위 결정 자동화 등, 디버깅을 자동화하기 위한 노력 꾸준히 이어져 왔다. 프로그램 (오류) 자동수정(Automated Program Repair; APR)은 디버깅 자동화의 최종 단계이지만, 다른 기술에 비해서 비교적 최근에 본격적인 연구가 시작되었다. 본 튜토리얼에서는 프로그램 자동 수정이 발전해온 역사와 주요 기술들을 설명하고, 그것들이 작동하는 원리에 대해서 살펴본다. 특히, 오류 수정 확률이 높은 패턴 기반의 프로그램 자동 수정 방법에 대해서 자세히 살펴본다. 그리고, 간단한 프로그램을 이용해서 프로그램 자동 수정이 이루어지는 방법을 직접 실습한다.
- ◆ 약력:
 - 현 경북대학교 컴퓨터학부 조교수
 - 룩셈부르크대학교 SnT 책임연구원
 - 홍콩과학기술대학교 박사후 연구원
 - 서강대학교 학사/석사/박사
- ◆ 연구분야:

Automated Program Debugging, Program Repair, Software Testing, Fuzzing.

튜토리얼 T2: Challenges on System IP Design Automation Software

- ◆ 일시: 1월 19(수) 13:00~14:30
- ◆ 장소: 세미나실 1
- ◆ 제목: Challenges on System IP Design Automation Software
- ◆ 연사: 박창규 연구원 (삼성전자)
- ◆ 튜토리얼 초록:

SOC(System on a Chip)은 AP 를 중심으로 현대 산업에서 휴대폰, 차량, IoT, 뉴럴프로세싱에 이르기까지 범위를 확장해나가는 대표적인 시스템 반도체 제품군입니다. 지난 수년간 소프트웨어 개발자들의 각종 산업 도메인으로의 진출이 매우 활발해졌습니다. 본 튜토리얼에서는 우리 소프트웨어 개발자들이 반도체 현장에서 겪을 수 있는 시행착오를 제가 경험한 사례를 바탕으로 소개드립니다. 특히, SoC 개발 과정내 설계자동화의 주요사례를 바탕으로 구체적인 도메인 엔지니어와의 협업사례를 소개합니다. 또한 본 튜토리얼에서 소개된 사례를 바탕으로 반도체 도메인에 대한 이해를 조금이나마 확장 하실 수 있는 기회를 제공합니다.
- ◆ 약력:
 - 삼성전자 반도체 부문 시스템 LSI
 - 삼성전자 반도체 부문 소프트웨어연구소
 - 아주대학교 전자공학부 박사
 - 아주대학교 전자공학부 학사
- ◆ 연구분야:

IoT, SOA, Emotion Recognition, Healthcare, Domain Specific Automation S/W, Web based Visualization

튜토리얼 T3: 신뢰성 있는 인공지능 품질평가방안 및 사례

- ◆ 일시: 1월 19일(수) 13:00~14:30
- ◆ 장소: 세미나실 2
- ◆ 제목: 신뢰성 있는 인공지능 품질평가방안 및 사례
- ◆ 연사: 이승은 선임 (와이즈스톤)
- ◆ 튜토리얼 초록:

인공지능은 이미 사람이 해결하던 여러 작업을 스스로 판단하고 수행하는 다양한 분야에 활용되고 있습니다. 하지만 모든 제품 및 서비스의 완성도는 품질에 따라 결정되듯이 인공지능 제품도 예외는 아닙니다. 다양한 분야에 활용되기 때문에 그에 따른 품질이 중요합니다. 본 튜토리얼에서는 인공지능 제품 품질을 어떻게 평가할 수 있는지, 또 그에 따른 사례에 대해서 말씀드리고자 합니다.
- ◆ 약력:
 - 에이아이플러스인증 시험기관 실무자
 - KOLAS 공인시험기관 공인시험원
 - 동국대학교 산업시스템공학 학사
- ◆ 연구분야:

인공지능, 소프트웨어 품질 평가, 데이터 품질 평가

튜토리얼 T4: 프로그램 합성 소개

- ◆ 일시: 1월 19일(수) 14:40-16:10
- ◆ 장소: 그랜드홀 2
- ◆ 제목: 프로그램 합성 소개
- ◆ 연사: 이우석 교수 (한양대)
- ◆ 튜토리얼 초록:

프로그램 합성은 사용자로부터 원하는 프로그램에 대한 조건을 입력받아 자동으로 바람직한 프로그램을 생성하는 기술이다. 최근들어 기술수준이 눈에 띄게 발전하고 있으며 일반인을 위한 자동 프로그래밍 뿐 아니라 프로그램 검증, 최적화, 하드웨어 및 소프트웨어 보안, 코딩 교육 등 다양한 응용에 사용되고 있다. 본 튜토리얼에서는 프로그램 합성 알고리즘들을 개괄적으로 소개하고, 용도에 맞게 사용할 수 있는 프로그램 합성 도구들과, 필요에 맞는 합성 도구를 만들 수 있는 프레임워크를 소개한다.
- ◆ 약력:
 - 2018 - 현재 : 한양대학교 ERICA 소프트웨어학부 조교수
 - 2017 - 2018 : University of Pennsylvania 박사 후 연구원
 - 2016 - 2017 : Georgia Tech 박사 후 연구원
 - 2009 - 2016 : 서울대학교 컴퓨터공학박사
 - 2005 - 2009 : 서울대학교 컴퓨터공학학사
- ◆ 연구분야:

프로그래밍 언어

튜토리얼 T5: Towards Secure Interoperation in Android

- ◆ 일시: 1월 19일(수) 14:40-16:10
- ◆ 장소: 세미나실 1
- ◆ 제목: Towards Secure Interoperation in Android
- ◆ 연사: 황성재 교수 (성균관대)
- ◆ 튜토리얼 초록:

안드로이드 플랫폼은 서로 다른 모듈간의 연동을 통해 사용자에게 다양한 서비스를 제공한다. 이러한 다양한 모듈과의 연동은 애플리케이션 기능 확장이라는 강력한 기능을 제공함과 동시에, 신뢰할 수 없는 외부 모듈로 인한 보안 취약점을 드러내기도 한다. 본 세미나에서는 안드로이드 플랫폼 상의 모듈간 연동에서 발생할 수 있는 보안 취약점을 소개한다. 안드로이드 앱 사이의 연동, 앱과 시스템 유틸리티 사이의 연동, 앱을 개발하는 서로 다른 프로그래밍 언어 사이의 연동에서 발생할 수 있는 보안 문제를 소개하고, 이러한 보안 문제를 해결하는 정적분석 및 자동화 테스트 프레임워크 연구에 대해 소개한다.
- ◆ 약력:
 - 2021.09. ~ 현재: 성균관대학교 소프트웨어학과 조교수
 - 2017.02. ~ 2021.08.: LG 전자 VS 사업본부, 책임연구원
 - 2014.08. ~ 2017.02.: LG 전자 MC 사업본부, 선임연구원
 - 2011.06. ~ 2012.02.: 한국인터넷진흥원, 주임연구원
 - 2018.02. ~ 2021.02.: 한국과학기술원 박사과정 (지도교수: 류석영)
 - 2012.03. ~ 2014.02.: 한국과학기술원 석사과정 (지도교수: 김용대)
 - 2005.03. ~ 2008.02.: The University of Manchester 학사
- ◆ 연구분야: Software Testing, Software Security, Empirical Study

튜토리얼 T6: 소프트웨어 개발에서의 프로그래밍 심리학

- ◆ 일시: 1월 19일(수) 14:40-16:10
- ◆ 장소: 세미나실 2
- ◆ 제목: 소프트웨어 개발에서의 프로그래밍 심리학
- ◆ 연사: 이재용 교수 (한서대)
- ◆ 튜토리얼 초록: 소프트웨어 개발자들은 매우 빈번한 상호작용을 해야 하는 동시에 혼자만의 깊은 내적 정신작용으로 복잡하고 어려운 문제를 해결한다. 프로그래머의 모든 행위는 성격심리, 사회심리, 인지심리, 학습심리로 해석된다. 최근 PPIG의 연구는 개발자의 성격과 SDLC와의 관련성을 밝혀냈으며 미국, 영국, 캐나다의 일부 대학들이 성격유형을 고려한 팀 구성을 통하여 프로그래밍을 가르치고 있다. 최근의 연구된 프로그래밍 심리학은 대학에서의 프로그래밍 학습에 유용한 수단이 될 수 있으며 기업에서는 조직, 팀, 개인의 발전을 도모하는 수단으로 활용하여 기업 전체와 개인의 역량을 증대시킬 수 있다. 본 튜토리얼에서는 HCI의 한 분야인 프로그래밍 심리학을 개괄하고 성격검사를 통하여 성격과 각 소프트웨어 개발 주기와의 적합성을 설명한다. 이는 SDLC의 전체 공정을 경험하여야 하는 개발자에게 전체 공정을 효과적으로 탐색하도록 돕는다. 또, 가장 효율적인 SDLC의 경험 순서를 확인함으로써 개발자의 부담을 감소시켜 개발 능력을 고양시킬 수 있다. 이를 통하여 개인의 발전을 넘어서 팀과 기업의 생산성을 높일 수 있다는 점을 확인한다.
- ◆ 약력:
 - 2000~현재 한서대학교 컴퓨터공학과, 무인항공기학과 드론응용전공 교수
 - 1993~1999 수원여자대학 인터넷학과 조교수
 - 1991~1993 과학기술연구원 시스템공학연구소 연구원
 - 인하대학교 전자계산공학 학사, 석사, 박사
 - HYCU 상담심리학 석사
- ◆ 연구분야: 프로그래밍 심리학, A.I. 교육, HCI, 공학심리학, 인간무인이동체상호작용

KCSE 2022 기초강연

기초강연 I

- ◆ 일시: 1월 19일(수) 16:40-17:30
- ◆ 장소: 그랜드홀 2
- ◆ 제목: 소프트웨어 안전 엔지니어 양성 방안
- ◆ 연사: 권기현 교수 (경기대)
- ◆ 초록:

소프트웨어가 중심인 디지털 대전환 시대에서 소프트웨어 안전은 아무리 강조해도 지나치지 않다. 왜냐하면 소프트웨어가 전 산업의 두뇌 역할을 감당하고 있기 때문이다. 인체 두뇌 장애가 사람에게 막대한 피해를 주듯이, 소프트웨어가 장애를 일으키면 소프트웨어 의존도가 높은 기업일수록 막대한 경제적 손실을 입는다. 이러한 이유로 위험 요인(hazard)을 식별하고, 위험 요인으로부터 발생할 수 있는 피해를 허용 가능한 수준까지 낮추는 완화 대책을 구현하여 사고로부터 재산, 환경, 인명을 보호할 수 있는 소프트웨어 안전 엔지니어(Software Safety Engineer)를 채용하는 기업이 늘고 있다. 소프트웨어 안전 엔지니어의 주요 업무로는 소프트웨어 결함을 예방하는 안전 프로세스 구축, V&V(Verification and Validation) 활동을 통한 결함 검출 및 제거, 또한 런타임 장애를 허용하는 안전 아키텍처를 설계하고 구현하는 일이다. 뿐만 아니라 발생 가능한 소프트웨어 장애 유형을 식별하고, 장애 유형별 근본 원인 파악 및 완화 대책을 구현하고, 완화 대책이 의도한대로 잘 작동하는지를 모니터링 할 수 있어야 한다. 정부에서도 소프트웨어 안전의 중요성을 인지하여 2020년 12월에 소프트웨어 안전 확보를 위한 지침을 과학기술정보통신부 고시로 제정하면서, 소프트웨어 안전 총괄 담당자 지정을 권고하고 있어서 소프트웨어 안전 전문 인력에 관한 산업체의 수요가 증가할 것 예상된다. 그러나 아쉽게도 소프트웨어 안전 전문 인력을 체계적으로 양성하는 교육 프로그램이 국내 대학에 거의 없는 실정이다. 이러한 시점에, 연사가 소속된 대학교에서 진행중인 소프트웨어 안전 인재 양성을 위한 기초교육, 융합교육, 전공교육 프로그램을 소개한다.

- ◆ 약력:

권기현 교수는 경기대학교 전산계산학과 이학사를 1985년에 취득하였고, 이어서 중앙대학교 전자계산학과 이학석사와 공학박사를 각각 1987, 1991년에 취득하였다. 그후 1991년부터 현재까지 경기대학교 AI 컴퓨터공학부 교수로 재직중이다. 주요 연구 분야로는 정형 검증, 기능 안전 및 소프트웨어 안전성이며 기능 안전 전문가로 TÜV Functional Safety Engineer, Functional Safety Professional, Functional Safety Expert 자격을 모두 취득하였다. 주요 경력으로는 1999년과 2006년 두 차례에 걸쳐 미국 카네기멜론대학 전산학과 Edmund Clarke 교수 연구실에서 모델 체킹을 연구하였고, 경기대학교 전산정보원장, 한국정보과학회 소프트웨어공학 소사이어티 회장, 한국연구재단 전문위원, 정보통신산업진흥원 대기업참여제한심의위원, 소프트웨어와사회안전협회 이사 등을 역임하였다. 또한, 2021년부터 경기대학교 소프트웨어 중심대학 사업단장을 맡아서 SW 혁신을 통한 K-안전 융합 인재 양성에 주력하고 있다

기조강연 II

◆ 일시: 1월 20일(목) 16:40-17:30

◆ 장소: 그랜드홀 2

◆ 제목:

Software Defined Weapon - Open Innovation을 통한 신뢰성있는 자율화 무기체계의 개발

◆ 연사: 김동환 연구위원 (LIG 넥스원)

◆ 초록:

ENIAC 컴퓨터가 탄도계산을 위해 처음 개발된 이후, 무기체계 소프트웨어를 개발하고 유지보수하는 비용(Total Cost of Ownership)이 천문학적인 숫자에 이르자 미국방성은 가장 신뢰성이 있다는 Ada 라는 프로그래밍언어를 개발하여 1980년대 표준으로 사용하였다. 또한 미국방성의 지원으로 설립된 Software Engineering Institute는 CMMi, Software Product Line 및 사이버보안 분야 등에서 전세계의 기준이 되는 성과를 내었다. DARPA 는 TCP/IP, GPS 와 같은 기술을 개발하여 전세계 사람들의 삶을 풍요롭게 하는 것에 크게 기여하였다. 이러한 성과는 개방적인 문화를 통한 Open Innovation 을 통해 가능한 일이었다. AI 및 소프트웨어로 특징지어지는 4 차산업시대 들어서 미군은 더 적극적으로 실리콘밸리의 스타트업 들을 찾아가서 신속하게 적용할 수 있는 기술을 찾고 있다. 우리 국방도 4 차산업시대에 스마트국방을 내세우며 각 군 및 국방관련 연구소를 중심으로 다양한 산학 프로그램을 수행중에 있으며 적극적으로 Open Innovation 을 실천하고 있다. 특히 현대전에서 정보전 및 원격전은 매우 중요한 개념으로서 이를 실현하기 위한 자율화 무기체계(Autonomous Weapon System)의 개발에 총력을 기울이고 있다. 그러나 반대로 딥러닝 기반의 블랙박스 시스템은 그 신뢰성 및 안정성에서 검증이 어려워 LAWS(Lethal Autonomous Weapon System)와 같은 연구개발에 윤리적인 문제로 산학이 참여하지 않겠다는 비토를 놓는 현상도 초래되고 있다. 그렇지만 오히려 적성국에서는 세계적인 우려와는 달리 AI 를 무기체계의 적극 반영하여 게임체인저가 되려는 것 또한 피할 수 없는 상황이다. 이러한 시점에서, 연사가 소속된 회사에서는 Open Innovation 을 통한 신뢰성있는 자율화 무기체계 개발을 위한 연구개발 플랫폼(KMPLG : Korea MUM-T platform Leading Group)을 구축하고자 노력하고 있어, 이외의 구현을 위한 국내외의 신뢰성있는 자율화 무기체계의 연구개발 동향 및 문제점을 소개하고 추진방향에 대해 소개하고자 한다.

◆ 약력:

김동환 연구위원은 인하대학교 전산계산학과 이학사를 1985 년에 취득하였고, 이어서 KAIST 에서 소프트웨어공학 전공으로 전산학과 석사를 취득하였다. 그후 1987 년부터 국방과학연구소 선임연구원으로 9 년간 연구소에서 근무하면서, KAIST 에서 데이터베이스 전공으로 박사과정을 수료하고, 전자계산기조직응용기술사 자격증을 취득하였으며, 우수연구원으로 국방부장관상을 수상하였다. 국방과학연구소 퇴직후 일반 기업에서 1995 년부터 2009 년까지 Location-based service 및 차량항법시스템 개발을 주도하여 아이나비로 신소프트웨어대상인 정보통신부장관상을 수상하였다. 2010 년부터는 현재까지 LIG 넥스원의 연구위원으로서 함정전투체계, 사이버전 및 자율형 무기체계 개발 등의 시스템 분석설계 및 교육을 수행하고 있으며, 국산 소프트웨어를 무기체계에 적극적으로 적용한 공로로 IT 융합기업인상인 산업통상부장관상을 수상하였다. 현재는 주요 연구 분야로 신뢰성있는 자율형 무기체계와 인간과의 협업을 위한 MUM-T 체계 개발과 교육을 담당하고 있으며, 한국정보과학회 종신회원으로서 국방소프트웨어연구회 운영위원, 국방품질연구회 소프트웨어분과 운영위원으로 활동하고 있다

KCSE 2022 신진 연구자 초청 발표

신진 연구자 초청 발표 N1

- ◆ 일시: 1월 19일(수) 17:40-18:30
- ◆ 장소: 그랜드홀 2
- ◆ 제목: 머신러닝 소프트웨어를 위한 자동화 테스트 기술
- ◆ 연사: 김미정 교수 (UNIST)
- ◆ 초록:

머신러닝 기술을 사용하여 인공지능 시스템을 구축하는 데 관심이 증가함에 따라 머신러닝 알고리즘을 쉽게 통합할 수 있도록 Tensorflow, PyTorch 등 다양한 라이브러리가 출시되었다. 이런 라이브러리도 다른 소프트웨어 처럼 수많은 버그가 포함돼 있으며 해당 버그는 머신러닝 모델의 정확성과 성능에도 영향을 미친다. 따라서 머신러닝 소프트웨어의 더 나은 안정성을 위해 자동화 테스트의 필요성이 대두되고 있지만, 머신러닝 라이브러리 함수 중 상당수가 특정 제약 조건을 따르는 구조화 된 입력값을 기대하기 때문에 기존 테스트 연구 방법으로는 머신러닝 소프트웨어를 효과적으로 테스트 할 수 없다. 본 세미나에서는 API 문서에서 함수 제약 조건을 추론하고 잘못된 테스트 입력값을 분류하여 이 문제를 해결하는 테스트 접근 방식을 제시한다.
- ◆ 약력:
 - 울산과학기술원 조교수, (2021.03.~현재)
 - Purdue University 박사후연구원
 - Hong Kong University of Science and Technology, 컴퓨터공학과, 박사
 - Georgia Institute of Technology, 컴퓨터과학과, 석사
 - University of Illinois at Urbana Champaign, 컴퓨터과학과, 학사
 -
- ◆ 연구분야:

Software Testing, Software Security, Empirical Study

신진 연구자 초청 발표 N2

- ◆ 일시: 1월 19일(수) 17:40-18:30
- ◆ 장소: 세미나실 1

- ◆ 제목: SW 가시화를 위한 메타모델과 모델변환

- ◆ 연사: 손현승 교수 (목포대)

- ◆ 세미나 초록:

고품질의 SW 개발은 소스 코드와 개발 프로세스에 대한 지속적인 관리가 필요하다. CMMi, SP 와 같은 프로세스 인증은 고품질 SW 개발의 전통적인 방법이다. 그러나 수작업을 통한 SW 프로세스의 품질관리 수행은 많은 인력과 자원이 소모된다. SW 가시화(Visualization)는 수작업 관리의 방안의 대안로 프로세스, 아키텍처, 문서를 시각화하고 자동화한다. 가시화는 역공학 기법을 통해 SW 개발의 가장 어려운 점인 SW 비가시성을 극복함으로써 SW 개발의 전체 과정을 파악하며, 이를 통하여 SW 품질을 측정하고 관리한다. 그러나 SW 가시화를 구축하기 위해서는 이종의 도구들의 통합이 필요하다. 본 발표에서는 SW 가시화를 실현하기 위한 모델 변환과 메타 모델에 대해서 소개한다. 모델 변환 및 메타모델은 이종의 데이터와 플랫폼을 연결하기 위한 방법으로 SW 가시화와 같이 이종의 플랫폼을 이용한 시스템 통합에 필요한 기술이다. 본 발표는 메타모델의 기본적인 설계 방법과 모델 변환에 대한 자세한 기술을 설명할 것이고 이를 활용한 SW 가시화 응용 사례들을 살펴볼 것이다.

- ◆ 약력:

- 2021. 03 ~ 현재 : 목포대학교 컴퓨터공학과 조교수
- 2017.10 ~ 2021. 02: (주)모아소프트 SBAS 사업부 책임연구원
- 2015.09 ~ 2017.12: 선문대학교 IT 교육학부 시간강사
- 2009.02 ~ 2015.08: 홍익대학교 전자전산공학과 박사 (지도교수: 김영철)
- 2007.03 ~ 2009.02: 홍익대학교 전자전산공학과 석사 (지도교수: 김영철)
- 1999.03 ~ 2007.02: 홍익대학교 컴퓨터정보통신공학과 학사

- ◆ 연구분야:

메타모델, 모델변환, 테스트 자동화, 소프트웨어 품질 관리

신진 연구자 초청 발표 N3

- ◆ 일시: 1월 19일(수) 17:40-18:30
- ◆ 장소: 세미나실 2
- ◆ 제목: A Robust Foundation for the Correctness of LLVM Compiler
- ◆ 연사: 이준영 박사 (CryptoLab)
- ◆ 세미나 초록:

Intermediate representation (IR) is a language that is used internally by a compiler to represent programs. Translation to an IR should preserve guarantees from the source language's specification because they enable various optimizations. This naturally makes an IR a language that is rich with high-level information. In LLVM, the semantics of important high-level features in IR was not rigorously defined. It caused compiler optimizations in LLVM to use different interpretations, and bad interactions between the optimizations resulted in miscompilation bugs that are hard to fix. To solve this problem, the IR's semantics must be defined precisely. Then, optimizations that are incorrect with respect to the chosen semantics must be fixed. Both processes are challenging because LLVM is a large, fastly evolving software.

This talk proposes (1) formal semantics of LLVM IR that resolves critical problems that we have found in the old IR semantics, making it consistent (2) a translation validation framework for LLVM's optimizations to validate the new semantics. We show that the old semantics of undefined behavior and memory model in the IR cannot explain important optimizations in LLVM. We propose new semantics that solves this problem. Next, we present Alive2, a translation validation framework for LLVM based on the new semantics. Alive2 relies on an automatic theorem prover to validate optimizations without any hints from LLVM. It supports most of integer and float operations, memory operations, function calls, and branches. To make validation practical, resources used by the tool is bounded.

The new formal semantics of undefined behavior has been adopted by LLVM. The 'freeze' instruction that is proposed by us is officially added into LLVM 10.0, and the official document is updated to use our semantics. Also, critical problems in the old memory model we have found were shared with compiler developers, and patches have landed in LLVM to fix it. Alive2 has found more than 50 miscompilation bugs in LLVM so far and is used daily by LLVM developers.

- ◆ 약력:

Juneyoung Lee received Ph.D. in Computer Science and Engineering from Seoul National University in 2021, and B.S. from Computer Science and Engineering from POSTECH in 2014. His research interests are the formal semantics of real-world programming languages and compiler verification. He is a recipient of the Distinguished Paper Award of PLDI'21. He actively worked as a contributor of the LLVM project, an open-source compiler infrastructure. Many compilers -- Clang (Apple's C/C++ compiler), NVIDIA's CUDA compiler, Intel's ICC, Swift, Rust -- rely on LLVM to perform compiler optimizations and generate code to various targets. He was a keynote speaker of 2020 LLVM Developers' Meeting, which is one of two seminal conferences for LLVM developers. Also, he actively contributed to Alive2, an automatic validation framework for compiler optimizations. Alive2 mathematically proves that a given compiler transformation is correct. The Alive2 online is daily used in the code review process of the LLVM project. He is currently working for CryptoLab as a military service agent. CryptoLab is a company developing a fast homomorphic encryption framework and post-quantum cryptography library.

신진 연구자 초청 발표 N4

- ◆ 일시: 1월 21일(금) 11:00 -11:50
- ◆ 장소: 그랜드홀 2
- ◆ 제목: 오픈소스 소프트웨어에서의 버그 정정 관리를 위한 지능형 프레임워크
- ◆ 연사: 양근석 교수 (경남대)
- ◆ 세미나 초록:

소프트웨어의 복잡성 증가로 소프트웨어 유지보수에서 디버깅의 비용이 약 610 억 달러 (연간)의 비용이 소요되고 있으며 시간으로 환산하면 약 6 억 2 천만의 개발자 시간이 소프트웨어 디버깅에 소요되고 있다. 만약 자동화된 소프트웨어 버그 정정 프레임워크가 존재한다면 개발자의 디버깅 시간을 줄여 개발 생산성을 확보할 수 있고 소프트웨어 유지보수의 비용을 줄일 수 있다. 본 발표에서는 오픈소스 소프트웨어에서의 버그 정정 관리를 위한 지능형 프레임워크를 소개한다. 소프트웨어 버그 심각도 예측, 소프트웨어 버그 개발자 추천, 소프트웨어 버그 로컬라이제이션, 소프트웨어 버그 정정을 하나의 프레임워크를 통합하여 소프트웨어 유지보수를 효율적으로 할 수 있는 방법을 살펴본다.
- ◆ 약력:
 - 2021. 3 ~ 현재 조교수, 컴퓨터공학부, 경남대학교
 - 2020. 9 ~ 2021. 2 연구교수, 서울시립대학교
 - 2017. 12 ~ 2020. 8 AI 개발자, KB 국민은행(본부)
 - 2016. 2 ~ 2017. 7 AI 개발자, MBC 문화방송사(본사)
 - 2015. 2 ~ 2016. 1 SW 개발자, 서울신문사(본사)
 - 2016. 2 ~ 2020. 8 박사, 컴퓨터학과, 서울시립대학교
 - 2013. 3 ~ 2015. 2 석사, 컴퓨터학과, 서울시립대학교
 - 2006. 3 ~ 2013. 2 학사, 컴퓨터공학부, 한국기술교육대학교

신진 연구자 초청 발표 N5

- ◆ 일시: 1월 21일(금) 11:00 -11:50
- ◆ 장소: 세미나실 1
- ◆ 제목: 비형식 요구사항 기반 유스케이스 추출을 통한 노력 추정 방법
- ◆ 연사: 박보경 교수 (진주교대)

◆ 세미나 초록:

고품질 소프트웨어를 개발하기 위해서는 소프트웨어 개발 초기 단계에서 요구 사항에 대한 정확한 분석이 필요하다. 그러나 대부분의 자연어 위주의 비형식 요구 사항은 분석이 어렵다. 이러한 문제를 해결하기 위해, 자연어로 작성된 비형식 요구사항에서 유스케이스 추출 방법을 제안한다. 자연어 요구 사항을 이해하기 위해 Fillmore 의 격문법(Case Theory)을 개선했다. 제안하는 방법은 개선된 격문법을 통해 자연어 요구 사항을 분류하고 구문 분석을 통해 문장의 구조와 관계를 식별하기 위해 요구 사항을 분석한다. 이 정보에서 동사를 나열하고 주동사를 추출한다. 8개의 개선된 사례를 기반으로 한 문장의 인수를 요구사항으로 분석한다. 분석된 정보는 시각적 모델링, 즉 변환된 모델을 통해 유스케이스를 추출한다. 마지막으로 추출된 유스케이스를 기반으로 소프트웨어 노력 추정을 계산한다. 이 방법은 이해 관계자가 쉽게 이해할 수 있도록 처리할 수 있는 자연어 요구 사항을 수정하지 않고 유스 케이스를 추출할 수 있다. 또한 유스케이스 점수(Usecase Point) 기법을 기반으로 소프트웨어 노력을 추정할 수 있다.

◆ 약력:

- 2008 년 2 월: 홍익대학교 컴퓨터정보통신(학사)
- 2012 년 8 월: 홍익대학교 일반대학원 소프트웨어공학(석사)
- 2020 년 8 월: 홍익대학교 일반대학원 소프트웨어공학(박사)
- 2021 년 3 월~8 월: 유원대학교 스마트 IT 학과 조교수
- 2021 년 8 월~현재: 진주교육대학교 컴퓨터교육과 조교수

◆ 연구분야:

요구공학, 소프트웨어 가시화, 저전력 및 성능 가시화, AI 기반 소프트웨어 개발 방법론, 컴퓨터교육

신진 연구자 초청 발표 N6

- ◆ 일시: 1월 21일(금) 11:00 -11:50
- ◆ 장소: 세미나실 2
- ◆ 제목: 소프트웨어 신뢰도 보증을 위한 효율적인 도달 가능성 분석
- ◆ 연사: 이낙원 박사 (한양대학교 박사후 연구원)

- ◆ 세미나 초록:

도달 가능성 분석은 프로그램이 안전하지 않은 상태 (e.g., 에러 위치)에 도달할 수 없다는 것을 증명함으로써 소프트웨어의 신뢰도를 보증하는 방법이다. 그런데 모델 체킹과 같은 자동화된 분석 방법이 있음에도 불구하고 프로그램의 높은 복잡도로 인하여 최신의 기법조차 너무 긴 분석시간을 소모하거나 혹은 도달 가능성 분석에 실패하곤 한다. 본 발표에서는 도달 가능성 분석을 위한 최신 기법들과 그 기법들의 문제점을 소개하고 기존 기법들의 속도를 향상시킨 TOUR 를 소개한다. TOUR 는 도달 가능성 분석을 위한 최신 기법인 추상 도달가능성 기반 모델 체킹 (ARMC)에 에러 위치 지향 탐색 전략을 적용함으로써 분석 속도를 향상 시킨다. TOUR 를 적용한 결과 652 개의 복잡한 프로그램들을 대상으로 기존 ARMC 기법보다 15%많은 프로그램들을 15% 적은 시간만에 분석하여 속도가 향상된 것을 확인 하였다.

- ◆ 약력:

- 2022.01~현재: 한양대학교 산학협력단 박사후 연구원
- 2015.03~2022.02: 한국과학기술원 전산학부 박사
- 2013.03~2015.02: 한국과학기술원 전산학부 석사
- 2006.03~2013.02: 건국대학교 컴퓨터공학부 학사

- ◆ 연구분야:

소프트웨어 검증, 소프트웨어 모델 체킹, 소프트웨어 시험, 소프트웨어 신뢰성 공학

우수 국제학회/학술지 초청 논문발표

- ◆ **JEST: N+1-version Differential Testing of Both JavaScript Engines and Specification**
 - The 43rd ACM/IEEE International Conference on Software Engineering (ICSE 2021)
 - Jihyeok Park, Seungmin An, Dongjun Youn, Gyeongwon Kim, Sukyoung Ryu (KAIST)
 - Session: A2 (보안)

- ◆ **Efficient SMT-Based Model Checking for Signal Temporal Logic (ASE2021)**
 - The 36th IEEE/ACM International Conference on Automated Software Engineering (ASE 2021)
 - Jia Lee, Geunyeol Yu, Kyungmin Bae (POSTECH)
 - Session: B1 (CPS)

- ◆ **HOTFUZ: cost-effective higher-order mutation-based fault localization (STVR)**
 - Software Testing, Verification and Reliability, 2021
 - Jong-In Jang (KAIST), Duksan Ryu (전북대), Jongmoon Baik (KAIST)
 - Session: B2 (결함위치 추정 및 결함예측)

- ◆ **Ahead of Time Mutation Based Fault Localisation using Statistical Inference**
 - The 32nd International Symposium on Software Reliability Engineering (ISSRE 2021)
 - Jinhan Kim, Gabin An (KAIST), Robert Feldt (Chalmers 공대), Shin Yoo (KAIST)
 - Session: B2 (결함위치 추정 및 결함예측)

- ◆ **JSTAR: JavaScript Specification Type Analyzer using Refinement**
 - The 36th IEEE/ACM International Conference on Automated Software Engineering (ASE 2021)
 - Jihyeok Park, Seungmin An, Shin Wonho, Yusung Sim, Sukyoung Ryu (KAIST)
 - Session (C1 프로그램 이해 및 분석)

- ◆ **Observation-based approximate dependency modeling and its use for program slicing (JSS)**
 - Journal of Systems and Software
 - Seongmin Lee, David Binkley (Loyola 대학), Robert Feldt (Chalmers 공대), Nicolas Gold (UCL), Shin Yoo (KAIST)
 - Session (C1 프로그램 이해 및 분석)

- ◆ **SymTuner: Maximizing the Power of Symbolic Execution by Adaptively Tuning External Parameters**
 - The 44th ACM/IEEE International Conference on Software Engineering (ICSE 2022)
 - Sooyoung Cha, Myungho Lee, Seokhyun Lee, and Hakjoo Oh (고려대)
 - Session (C2 AI 와 SE I)

- ◆ **NPEX: Repairing Java Null Pointer Exceptions without Tests (ICSE 2022)**
 - The 44th ACM/IEEE International Conference on Software Engineering (ICSE 2022)
 - Junhee Lee, Seongjoon Hong, Hakjoo Oh (고려대)
 - Session (D1 프로그램 수정)

KCSE 2022 논문 목차

일반 논문

클라우드-엣지 협업 지능을 위한 소프트웨어 아키텍처와 기술 현황.....	Xiangchi Song, 고인영 1
다양한 디스플레이 환경을 위한 적응형 UI 레이아웃 프레임워크.....	윤진, 이은영, 이상현, 백종문 11
사전 학습 언어 모델들을 활용한 버그 담당자 자동 할당 성능 향상.....	왕대성, 성훈, 김민하, 이찬근 21
Actor-Critic 계열 강화학습 알고리즘의 행동 정확성 평가 방법.....	김영재, 홍장의 23
해석 가능한 시계열 이상탐지 모델을 활용한 생산 능력 계획 수립 방안	강민정, 이상민 33
코드 변경에 대한 검출 기법과 사람의 묘사 차이 연구.....	김무준, 김범철, 김진대 41
홍부 방사선 영상 학습 성능 개선을 위한 오픈 데이터셋 품질 평가 기법.....	김양근, 박예슬, 선주성, 이정원 43
오픈 소스 기계학습 어플리케이션에 대한 결함 사례 조사.....	최윤호, 이창공, 방제형, 남재창 52
효율적인 아키텍처 패턴 적용을 위한 소프트웨어 요구사항 분류체계에 관한 연구.....	최종우, 민상윤 54
소프트웨어 요구사항 분류 모델의 성능 향상을 위한 불균형 데이터 처리에 관한 연구.....	최종우, 이영준, 임채균, 최호진 56
협동 로봇의 센싱 데이터 특징 기반테스트 데이터셋 구성을 통한 학습 모델 신뢰성 분석.....	김진세, 배수빈, 박예슬, 이정원 58
AIAG-VDA 방법론 기반 SFMEA: 효과적인 자동차 SW 안전분석.....	김소연, 권기현 60
엣지 클라우드 환경에서 다목적 최적화 알고리즘을 활용하여 비용 및 서비스 지연을 최소화하는 마이크로서비스 배포 방법	김민협, 고인영 62
소프트웨어 결함 예측을 위한설명 가능한 인공지능 적용.....	최지원, 이재욱, 류덕산, 김순태 70

단편 논문

복잡한 버그의 템플릿 생성을 위한 Change Action 측면에서의 과거 수정 이력 분석.....	허진석, 이은석 79
동영상 스트리밍 서비스에 HTTP/1.1 HTTP/2 프로토콜 적용별 프로토콜 성능 평가.....	도지영, 이성원 83
멀티 체크 버그 정정을 위한 버그 블럭과 Code BERT 활용.....	김지성, 호혜민, 양근석, 이병정 86
모바일 어플리케이션 자동화 테스트를 위한 딥러닝 기반 UI 경로 지도의 생성.....	방혜원, 박정민 2, 한영준 90
딥러닝 기반 얼굴인식 손실 함수 기술 동향.....	임현택, 김태윤, 김창석 94
이더리움 스마트 컨트랙트의 업그레이드 전략에 따른 가스 영향도 분석.....	김진영, 김미수, 이은석 98
의심스러운 조건문을 이용한 결함 위치 식별 및 수정 개선.....	강성민, 유신 102

경량 뉴럴 네트워크 탐색을 위한 가지치기, 지식증류, 그리고

양자화에 기반한 하이브리드 압축 방법.....이재원, 이상민 105

FPGA 네트워크 기능 가속기를 지원하기 위한 빠르고 효율적인 소프트웨어 스택.....부준혁, 김장우 109

딥 러닝을 활용한 메소드와 주석 간의 적합성 검증 모델.....박재현, 이동건, 장용훈, 서영석 113

그래프 합성곱 신경망을 활용한 취약점 탐지 벤치마크.....최유라, 권영우 117

자카드 지수를 이용한 한국어 요구사항 유사도 분석 자동화.....장우성, 문소영, 김영철 121

Use case 명세서 기반 자동 가이드 코드 생성.....정세준, 문소영, 김영철 125

코드가시화의 고도화 연구.....김장환, 문소영, 김영철 128

Keypoint-RCNN 을 이용한 반려견 대퇴골 탈구인식을 위한 수의영상판독 시스템.....김민경, 김지인 130

히스토리 기반 타임뱅크 봉사 파트너 추천 알고리즘.....선다영, 손서락, 양혜란, 유원상 134

멀티태스크 PLC 프로그램의 테스트를 위한 주기 별 테스트

시퀀스 실행 제어 방법.....은형석, Lingjun Liu, 지은경, 배두환 137

Concolic 테스팅과 Fuzzing 을 결합한 유닛 테스트 자동화 기술.....김윤삼, 김문주 141

산업체 논문

소프트웨어 수출을 위한 시장주도 요구공학 적용 사례.....이종길, 길문주, 오정섭 143

반도체 공정 유지 보수 비용 절감을 위한 공정 레시피 데이터 학습 및 장비 결함 검출.....최민호, 유동연, 이정원 149

학부생 논문

개인정보보호를 위한 SGX 기반 학습데이터 비식별화 인공지능 학습 프레임워크.....김남령, 옥지원, 김성민 154

CFG 생성 타당성 검사를 위한 테스트 자동화 연구.....정소희, 김동우, 최윤자 161

Kubernetes 를 활용한 멀티 클라우드 모니터링 솔루션.....박신영, 김병권, 배유진, 심석인 166

아동·청소년을 위한 인공지능 스피커: 심리 상담 통합 시스템 개발.....김수아, 이예빈, 정수민, 최미소, 최혜빈 174

생체 정보의 비식별화를 위한 지문 탐지 모델.....배은기, 김한동, 권동균, 한종대 182

스마트 팩토리 - 가스라이터 제조 공정에서의 스티커 부착과 관련된

불량 검출 프로그램 개선.....신관수, 송호준, 조영진, 신연순 188

수업 질문 자동응답을 위해 KoBert 를 이용한 챗봇과 Doc2Vec 을 이용한

챗봇과의 성능비교.....허주은, 이선아 195

양방향 메트릭 선택 기반 비지도 학습 결함 예측.....백인선, 권수진, 박성수, 박수은, 남재창 200

기업형 블록체인 플랫폼인 하이퍼레저 베수의 기술적 배경.....	임영택, 박민하, 주성호, 배병일	209
다중 피부병변 검출 심층학습을 위한 피부병변 합성 기반 데이터 증강 기법.....	박범진, 이영찬, 장현재, 유원상	216
텍스트 기반의 CAPTCHA 보완 모델.....	김태호, 고유라, 한중대	221

클라우드-엣지 협업 지능을 위한 소프트웨어 아키텍처와 기술 현황

Xiangchi Song^o, 고인영

한국과학기술원 전산학부

{xcsong, iko}@kaist.ac.kr

Status Quo of Software Architecture and Technologies for Cloud-Edge Collaborative Intelligence

Xiangchi Song^o, In-Young Ko

School of Computing, Korea Advanced Institute of Science and Technology

Abstract

With the advent of the Internet of Things (IoT) era, the traditional cloud computing model has been unable to cope with the data processing needs of more and more edge devices, and then the concept of edge computing emerged accordingly. The proposal of this technology has effectively helped the cloud to reduce the pressure on computing. But due to the limitations of the computing power of edge devices compared to the cloud, we proposed *Cloud-Edge Collaborative Intelligence*. This article mainly introduces the background and current development of this technology, and analyzes the relationship between its internal components and current artificial intelligence. Then two collaborative models to achieve this goal are introduced, namely cloud-edge collaboration and edge-edge collaboration, and the core technologies required are analyzed in detail. Then the four components to realize this architecture are presented, which are data collaboration, resource management collaboration, application collaboration and task scheduling collaboration. Finally, we summarize how the related technologies and models mentioned in the article improve and promote the development of cloud-edge collaborative intelligence, and put forward a prospect for the future development of this technology. We believe that this technology will surely promote a major change in the industrial internet in the future.

1. Introduction

With the advent of the IoT era, the number of 5G equipment holdings has increased exponentially, combined with increasing diversified service scenarios. According to IDC's forecast, by 2025, the total amount of global data will be greater than 180 ZettaBytes, and more than 70% of the data will be generated by the IoT devices [1]. In addition, owing to the significant amount of data transmission, the corresponding bandwidth requirements will increase rapidly. Simultaneously, owing to the concept of mobile broadband, large-scale machine type communication, and mission-critical internet [2], challenges are raised for our data processing performance. For example, the data processing rate must meet the extremely high traffic density requirements of the network. Regarding the application scenarios of a large number of terminals in the IoT, the network is required to support hundreds of billions of device connections, not solely to ensure extremely low latency, but also low power consumption and low cost.

To address this challenge, edge computing, combined with the existing centralized big data processing solution centered on the cloud computing model was developed. Edge computing is

the concept of running computing tasks with computing resources that are close to the data source, to effectively reduce the latency and data transmission bandwidth, and ease the load on cloud data centers [3]. In the future, technical barriers must be broken and design must be realized in the scenario as soon as possible.

In this study, we focused on *cloud-edge collaboration technology* [4], which is a new idea of integrating cloud computing, edge computing and artificial intelligence (AI). In this technical framework, calculations are performed with computing resources that are close to the data source, and by allocating a part of the calculation and decision-making to edge devices, we can effectively distribute the computing load, reduce the demand for data transmission bandwidth, and alleviate the load on the cloud data centers. Therefore, the usability of the cloud-edge interactive system is enhanced, data interaction between edge nodes is improved, and integrity and availability are guaranteed.

Data privacy, which has become an important issue in cloud computing can be also effectively dealt with using edge computing. Because edge devices undertake major parts of the algorithm training and data storage, it will significantly reduce

the amount of data upload and download to and from a cloud, which can as well reduce the probability of data being stolen or lost during the intermediate exchange process, and protect its security and privacy.

We anticipate proposing solutions suitable for this technology for users, improve the intelligent collaboration framework, and enable more edge intelligent scenarios to be realized. Therefore, in this study, we summarize and discuss the theoretical background, architecture models, and scenarios of cloud-edge collaboration.

In the next section, we first introduce the cloud-edge collaboration architecture, and then analyze the internal connection of its components and its relationship with the current AI technologies. We discuss the current issues of cloud-edge intelligence and explain existing solutions. Further, in Section 3, we present two collaborative models of the cloud-edge collaboration architecture and analyze their core technologies. In Section 4, we explain the four components that are required to realize cloud-edge collaborative intelligence and analyze their models. After introducing the relevant models and technologies, we discuss the interaction and development significance of these technologies and cloud-edge collaborative intelligence in Section 5. Finally, we summarize the main content of this paper, and then discuss the outlook of this cloud-edge collaboration technology in Section 6.

2. Background

2.1 The relationship between edge and cloud

As the computing power of edge devices is still relatively weak compared to cloud centers, large-scale learning models still need to be deployed in the cloud. Edge devices and networks can only perform some basic information capture, storage, and command execution. In the new cloud-edge collaboration model, AI algorithms will gradually migrate from the cloud to the edge, and the edge will address part of the training tasks of the intelligent algorithms, to realize the intelligence of edge data and improve the accuracy of results.

In the interaction process, in addition to the participation of the cloud and edge, Cisco [5] proposed a concept of fog computing that is different from edge computing in 2012, and then Iorga et al. [6] elaborated on the properties, service models and deployment strategies of fog computing in 2018. Fog emphasizes the formation of cloud to-things continuum [7] between the data computing center and data source, and provides users with the service of computing and storage on the

premise of ensuring maximum resource utilization and quality of service, to enable the entire system transform from a complete computing and storage system to a partial interactive service system, with redundant data no longer being stored, and it will be an emphasis on interaction and decision-making capabilities. Compared with the cloud, fog has a shorter latency and higher efficiency, because it can receive information from multiple endpoints through a single device for processing. Compared with the edge, fog is more scalable and can be utilized in large networks with multiple endpoints.

2.2 Artificial intelligence and cloud-edge collaboration

In the article OpenEI: An Open Framework for Edge Intelligence [8], edge intelligence is defined as the ability to enable edge devices execute intelligent algorithms. Because we stated that the intelligent algorithm based on deep learning cannot be afforded by the terminal device, usually this part of the task load exists in the cloud computing center with strong computing power. Therefore, Zhang et al. [8] mentioned four standards for measuring the strength of the cloud-edge relationship in this calculation and feedback process: accuracy, latency, energy and memory footprint. For the proposal of gradually mature learning algorithms, and the continuous optimization of resource allocation and task scheduling methods based on the cloud-edge collaborative computing architecture [9], the accuracy of task feedback and effect of resource allocation are gradually improving; however, the balance between accuracy and time consumption is the point that is required to consider in the future.

Owing to the development of technology, the number of edge devices and types of data have increased significantly. The gap between the data created by edge devices and amount of data that can be processed in the cloud has gradually widened; hence, we conclude that AI is necessary for the development of the technology discussed in our article. Gartner [10] proposed that by 2023, cloud-based AI will increase five times over 2019, making AI one of the top cloud services. They proposed that in the future, containers and serverless computing [11] will enable machine learning models to become independent functions before the public, thereby reducing costs while quickly responding to changes in the environment and making the most accurate decisions.

Cloud-edge collaboration is actually a common research area of AI and edge computing, and Edge AI [12] is the product of both. The judgment of AI can only be performed after the data is transmitted back to the cloud center, but the transmission time

is fatal for some fields; hence, the theory of AI decision-making in edge devices is proposed. Owing to the rapid growth of 5G coverage areas and device types, assigning different edge devices to different types of AI training, and connecting them to form a new edge network and real-time decision-making after data collection, will definitely be a major breakthrough in current technology.

2.3 Current issues and problems

We reviewed more than 60 journal and magazine papers published by researchers in the field in recent years. These articles can be divided into several categories: summary of current status, platform and framework development, model and algorithm improvements, actual scene application and technology development prospects, etc. By summarizing the existing works, we determined the following four typical issues and challenges. In view of the four challenges, we analyze them according to the current existing technology and try to come up with practical solutions in theory.

1) Trade-off between high accuracy and low latency

For any algorithm, we all hope that the results obtained are as accurate as possible, and the feedback time interval is as short as possible. However, owing to the limitation of network bandwidth and huge amount of data transmission generated daily, this challenge has become the focus of our current research. With the segmentation of deep learning models and proposal of large-scale optimization frameworks, we are able to obtain more accurate data and predictions [13]. However, simultaneously, balancing the relationship between accuracy and latency or determining the trade-off between the two is a problem we need to consider.

Regarding the accuracy of the algorithm and time feedback, we need to segment the deep learning model [14] and reduce the computing pressure on the edge by deploying large-scale intelligent algorithms and learning networks in the cloud. But simultaneously, two new challenges have emerged. The first is how the model should be segmented to maximize the efficiency of the entire system and the advantages of our cloud-edge collaborative intelligent technology; the second is whether we can assign the algorithm to different edge nodes and use the collaboration of the edge nodes to complete the entire system training process. Because the focus is different from what we discuss in this paper, we will not discuss these two issues temporarily.

2) Contradiction between the complexity of intelligent

tasks and execution power of edge devices

Because intelligent tasks often need to be divided into multiple sub-tasks to complete a goal, and multiple systems are called to cooperate to execute together, it has a great contradiction with the single function of the current edge device. The system requires data collection, pre-processing, calculation, control and feedback [15], but most edge devices are restricted by a lot of issues; hence, they only have the ability to complete a single sub-task. Therefore, the need to collaborate with the cloud to perform tasks together takes us back to the question on how to make this process more feasible.

Because of the large number of tasks and complex procedures at this stage, the interaction and cooperation of edge devices are required, and the frequency of collaboration between cloud and edge devices also increase. Because the function of a single edge device is single and its computing power is weak, the coordination mode between the edges needs to be altered, and the speed of information exchange also needs to be increased. Simultaneously, it is necessary to consider the issue of equipment deployment, because spatial distribution is also an important issue affecting performance, and the overall optimization of the system cannot be separated from the performance improvement of each component.

3) Balance between service quality and privacy protection

Owing to the development of edge computing, most tasks can be completed by edge nodes in cooperation with each other. However, it is difficult for edge devices to bear the pressure of large-scale AI algorithms. Therefore, to improve the quality of service, core data processing and algorithm training still need to be performed on the cloud, and this process will inevitably lead to privacy exposure challenges. Even if the edge device storage has the advantages of mobility, distribution, virtualization, and resource dynamics [16], there will be information gaps because of the lack of communication between data. Therefore, in certain scenarios, there is a balance between the quality of service provision and exposure of privacy rights.

With the implementation of the EU GDPR [17], data security issues have a major impact on the development of the current field. Simultaneously, for the convenience of reading and using data information, more cooperative service providers choose to share data. Therefore, we need to enhance the synergy between edge and edge, and cloud and edge under the premise of ensuring data security, to ensure that there will be no risk of user and device information leakage during this technological development.

4) Adaptiveness

Because the upper limit of the entire collaborative system's performance is fixed, requirements such as high accuracy or low latency cannot be met simultaneously, therefore for different user needs, we should choose to design specific architectures and operating standards for specific industries, or adjust our current edge network framework to meet the needs of the scenario, which is currently also an important challenge. Under these two completely opposite needs, whether we need to make edge intelligence technology more adaptable or make a targeted system is the challenge we can see in our daily lives.

Because IoT development provides edge devices with a certain degree of adaptability, we must break up the algorithm as much as possible, and try to simplify the execution tasks of the edge devices so that when choosing a new system, we can directly select the type of equipment we need for reorganization, to ensure the maximum adaptability of the entire system to different types of environments.

3. Cloud-Edge Collaborative Intelligent Technology

In July 2019, the China Cloud Computing Open Source Industry Alliance officially released the "Nine Application Scenarios of Cloud Computing and Edge Computing Collaboration" white paper [18] and proposed that an edge and cloud are dependent on each other and operate in coordination. They also provided application scenarios and typical cases, showing the landing form of edge computing. Driven by the development of the industrial internet, the system has gradually changed from "the cloud is only responsible for algorithm training and collaborative deployment, and the edge is only responsible for collecting data and executing instructions" to "the cloud transfers part of the training and storage tasks to the edge, and the overall system strengthens the coordination capabilities" mode.

In this section, we analyze cloud-edge collaborative intelligence technology from three aspects: development status, collaborative method and related key technologies, mainly based on two common collaboration modes of this technology: cloud-edge and edge-edge collaboration modes.

3.1 Cloud-Edge Collaboration

3.1.1 Development status

This collaborative approach is the basic structure of this technical field, and the current theory is relatively sufficient. In this section, we mainly discuss the impact of two collaboration methods around different cores on the system, several types of

collaboration based on this model and related technologies. We utilize the master-slave model to temporarily name these two methods: Cloud-centric and Edge-centric. We will run the current system through two different perspectives, examine the challenges in the current model, and summarize the current feasible solutions.

3.1.2 Cloud-centric cloud-edge collaboration method

In this collaborative mode, we need to determine a suitable cutting point for the neural network model or training algorithm, split it, and leave the computationally complex work in the cloud. The neural network cutting technology utilized here will be analyzed in detail in the follow-up part. Because this approach is cloud-oriented, the edge is only an auxiliary execution; therefore, how much computing power needs to be deployed at the edge is a question to consider in the future. The research on the cloud-centric part is still immature, and there is no systematic network segmentation method; hence, will not discuss it in detail for the time being.

3.1.3 Edge-centric cloud-edge collaboration method

The opposite of the above is the cloud-edge collaboration approach with the edge as the core. Because the edge is particularly critical for data capture, storage, and utilization. The difference from the aforementioned is that the cloud is solely responsible for the initial training, and the rest of the training and prediction will be handed over to the edge for execution. With the continuous expansion of 5G network architecture and the gradual improvement of edge computing platform functions [19], the locality of data will be better utilized to meet people's individual needs. As the standards around the edge side such as the edge cloud have been initially formed, future consideration is to improve the computing capabilities of the edge, further improve the level of cloud-edge collaborative services, and meet the needs of agile connection, business implementation, data optimization, data security and privacy protection [20].

3.1.4 Related key technology

1) Neural network segmentation

Regarding the existing challenge, which is neural network segmentation, owing to our consideration of cloud and edge computing power, we need to segment the neural network or train the model, and allocate the heavier part to the cloud, but there is still no fixed and mature measurement method. Neurosurgeon proposed by Kang et al. [14] is currently the most complete and representative solution. They analyzed the feasibility of large-scale intelligence and its transfer cost and

calculated the delay and energy consumption of implementing the most advanced DNN on the cloud and mobile devices. By providing eight in-depth hierarchical characterizations, it was determined that the DNN layer will lead to different calculation and data size characteristics according to its type and configuration. Furthermore, using the DNN calculation division across the cloud and mobile edge, a systematic method was designed to determine the best point of segmenting calculation. Then they developed a set of models that can predict the delay and power consumption of the DNN layer according to its type and configuration. Finally, they created the neurosurgeon system, which can significantly improve the end-to-end delay, reduce mobile energy consumption, and improve the throughput of the data center system.

2) Computing power network

It was just stated that to achieve the training and storage requirements of edge devices, it is a key issue to allocate tasks reasonably to reduce computing power; therefore, we propose the technology of computing power network. For AI applications, Lei et al. [21] proposed a computing power network solution based on the deep integration of cloud-network-edge. As illustrated in Figure 1, we propose an AI computing power network deployment process based on Lei's theory. On the basis of the original model, we added a task scheduling part, and regarded resource allocation as a sub-task of task scheduling. After analyzing user needs, the user specifies the AI service, or independently selects the corresponding AI service based on the analysis result, the system will allocate resources and manage the tasks, then deploy them on the designated user applications. In this way, the computing power is allocated, the storage method and network connection are planned simultaneously, and the node allocation and network scheduling are established. After this process, according to different needs of users, the entire system can automatically analyze and allocate resources to complete the coordinated processing of the entire application.

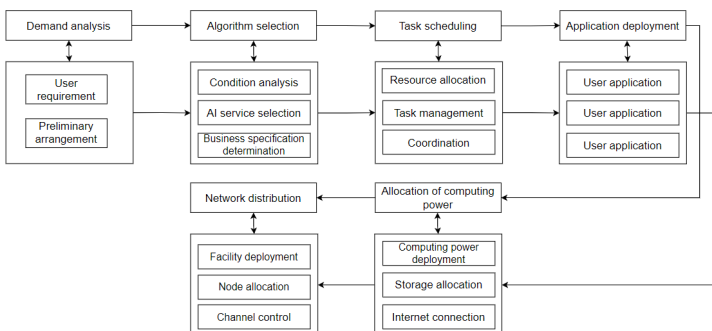


Figure 1. AI computing power network deployment process

As the computing power network is very complex and still in the initial stage of development, Jiang [22] proposed that it can be divided into generations by computing power measurement, starting from “building a computing power network system, verifying the key technology of computing power network, standardization and industrial advancement,” to developing computing power network technology.

3.2 Edge-Edge Collaboration

3.2.1 Development status

This kind of collaboration is different from the aforementioned cloud-edge collaboration, which fully embodies the cooperation between edge devices. In this mode, edge devices need to undertake certain training and data processing tasks, but as earlier stated, the computing power of a single edge is limited, and edge-edge collaborative training is required. Simultaneously, each edge device is called an "edge node," to connect the nodes to each other to form a network, covering our work area or the scope of daily life. Two methods are presented: considering data security, the first is the federated collaborative training method, because the edge device is responsible for parts of data collection and storage security. The second is the distributed training method, the collaborative training process between the edges.

3.2.2 Federated collaborative training method

Because conventional machine learning requires data collection and training in the cloud, data security issues are serious; therefore, for data privacy protection, a federated training collaborative method is proposed with edge nodes as the center. After the edge device obtains the local model and uploads it to the cloud, the cloud is responsible for connecting the single local models into a comprehensive model, to enable us to enhance our training and collaboration capabilities by integrating the resources of distributed nodes. In 2020, Daniel and Stefano [23] launched "Federated Analytics". This is a collaborative data science without data collection. This method can reuse the federated learning infrastructure to help engineers measure the quality of federated learning models based on real-world data when there is no data in the data center. In this way, we can even delete the learning part when there is no available data, to evaluate the quality of the federated learning model based on the displayed data, and further improve the privacy of federated training. Simultaneously, through federated training collaboration, we can also solve the data island using the optimization and synthesis of a single model, make full use of

resources, improve system accuracy and system interaction.

3.2.3 Distributed deep-learning training method

In this distributed training mode, each edge node will undertake part of the system's training tasks, and the data used for training is generated by the edge device. After all edge devices complete their tasks, they upload them to the cloud server to generate the final result. This theory has made preliminary progress. For example, Abadi et al. [24] proposed a system suitable for training and inference of deep neural networks, Microsoft [25] also released an open-source deep-learning toolkit, and Zou [26] proposed a new training strategy for distributed deep neural network for edge computing. Zhu et al. [27] also analyzed the challenges of distributed deep learning training, and some common deep learning frameworks have also begun to accept distributed training tasks. After distributed collaborative training between edge nodes, the accuracy of the final training is increased, and the model will be more complete.

3.2.4 Related key technology

1) Federated learning

In 2016, McMahan et al. [28] proposed a method that distributes training data on mobile devices and aggregates locally-computed updates of a shared machine-learning model. The release of this decentralized method officially marked Google's proposal of the concept of federated learning. Figure 2 illustrates the federated learning architecture. In 2019, Google [29] issued a document to launch a scalable production system based on TensorFlow for mobile devices.

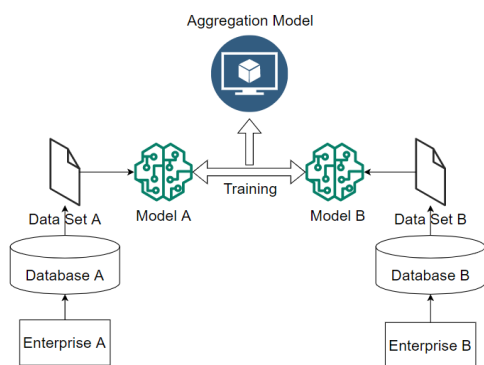


Figure 2. Federated Learning Architecture

We mentioned that the significant advantage of federated learning is that it can guarantee data security. Yang [30] mentioned that federated learning is the only way to break data islands. By aligning encrypted samples, encryption model training, and effect incentives, we completed a "closed-loop" learning mechanism, which not only protects privacy but also rewards institutions that contribute significant data. Zhou et al.

[31] compared federated learning methods and provided defense measures for security threats. Yang et al. [32] conducted a detailed analysis and summary of privacy protection methods in federated learning, such as data perturbation and differential privacy. All the teams have conducted research on the current federated learning privacy protection and security mechanisms and proposed future challenges and prospects.

It can be observed that although there are still several security issues, such as data quality issues and interaction efficiency issues that need to be resolved in federated learning, with the further improvement of model reliability and security, the advantages of federated learning will become more prominent.

2) Mobile edge computing

The current distributed mobile cloud computing collaborative architecture is relatively mature. Li et al. [33] summarized and discussed the distributed mobile cloud computing collaborative architecture and management mechanism for 5G. Regarding the collaborative process between the edges, to improve edge computing capabilities, improve the accuracy of training models, and increase privacy protection, mobile edge computing (MEC) is still a key technology. Liang et al. [34] summarized several international standard models and frame designs of MEC and sorted out key issues such as MEC resource allocation. MEC ISG [35] has published the relevant specifications on the basic technical requirements and reference architecture of MEC, and divided the Networks, Mobile edge host and Mobile edge system levels under MEC. The architecture diagram is illustrated in Figure 3.

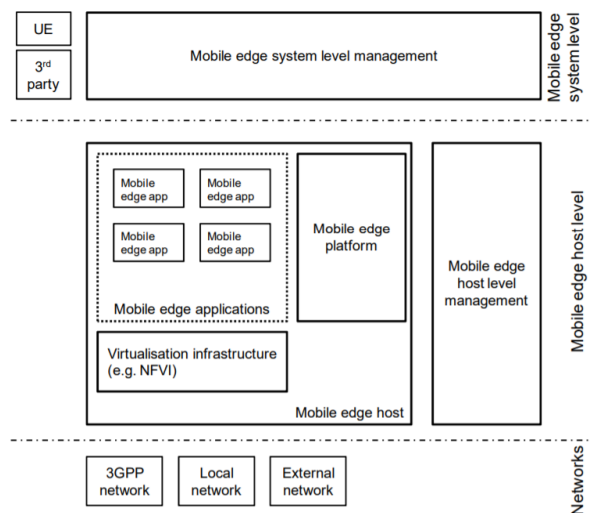


Figure 3. Mobile Edge Computing Framework (adapted from [35])

Regarding the physical deployment of the MEC host, Kekki

et al. [36] listed feasible solutions for the physical location of the MEC according to various requirements in terms of operation, performance, or security. The MEC can be configured with the local user plane function (UPF) and the base station, can be collocated with a transmission node or the local UPF and network convergence point, or coexist with the core network function.

4. Four components of cloud-edge collaboration

In this section, we will discuss four components of cloud-edge collaboration, namely data, resource management, application, and task scheduling collaborations. Their relationship with cloud-edge collaboration is illustrated in Figure 4.

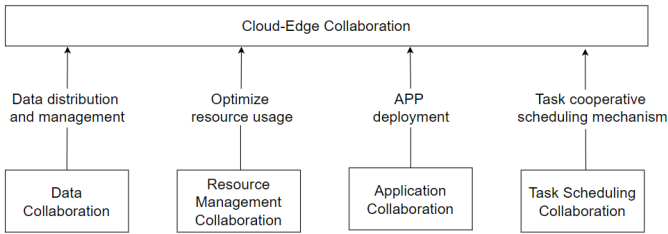


Figure 4. The Relationship between the Four Components and Cloud-Edge Collaboration

4.1 Data collaboration

Because the edge device is responsible for collecting and storing data in our collaborative technology, we first need to consider the distribution and management of data when considering the model. In this process, the edge device is responsible for collecting massive amounts of information, pre-processing it in accordance with corporate requirements or industry specifications, and uploading the processed results to the cloud for secondary processing, or completing part of the training tasks assigned by the cloud. Therefore, the process of data collection, transmission, processing, and efficient storage is a key consideration in the development of the internet industry. Dai et al. [37] proposed that in this environment, data storage and management are distributed. We transform the database server into a database node. Data management and collaboration are actually collaborations between nodes. The management mechanism is as illustrated in Figure 5. In this way, distributed data processing applications provide distributed data processing services responsible for data interaction, storage and synchronization, and flexible data collaboration mechanisms for cloud-edge collaborative applications through database nodes.

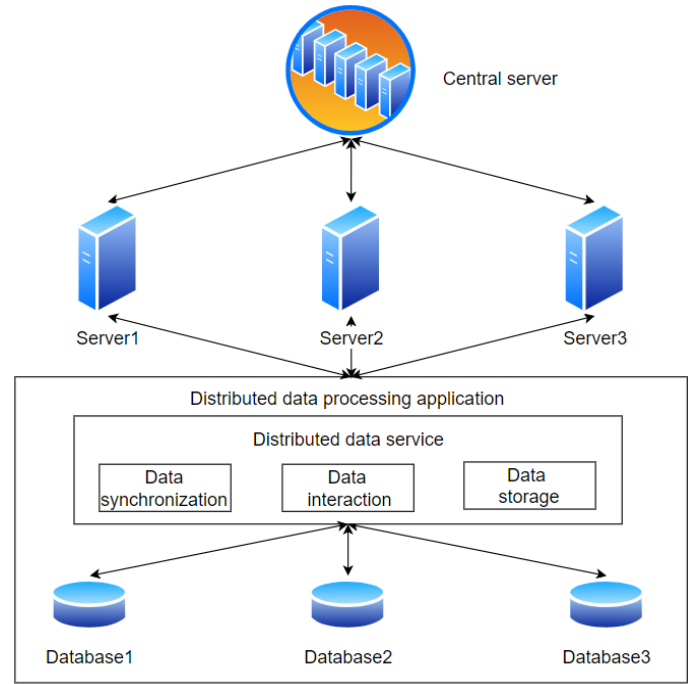


Figure 5. Implementation Mechanism of Distributed Data Collaborative Management

Through the data collaborative mechanism, we can significantly improve the ability to process large amounts of data in the cloud-edge collaborative environment, which provides a new possibility for the development of collaborative application mechanisms in the future. It also portrays the importance of data collaboration for cloud-edge collaborative architecture.

4.2 Resource management collaboration

One of the core issues of edge computing is the rational utilization and allocation of resources. While meeting the demand, allow the advantages of cloud, edge, and terminal, maximize the computing power of the three, optimize resource usage, data transmission, and task scheduling in all aspects, to achieve the maximum efficiency of the cloud-edge collaboration model. Deng et al. [38] have summarized the current research progress of resource management and collaboration. These methods include improved scheduling and decision-making algorithms to determine the optimal task scheduling strategy.

Using resource management collaboration, we can address issues such as the limited capabilities of edge devices, allow the advantages of the proximity of the edge and the terminal, meet the needs of users for the quality of service, and more effectively implement edge computing scenarios.

4.3 Application collaboration

The realization of application collaboration can make better use of the powerful computing capabilities and rich services of cloud-edge collaborative intelligence, increasing the complexity of edge applications, and simplifying the development, management and operation of Apps.

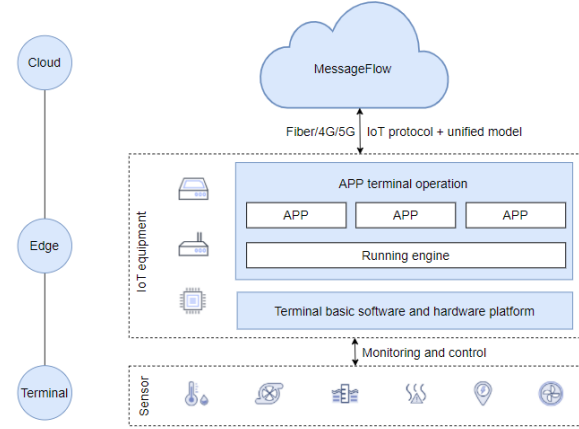


Figure 6. Huawei Cloud-Edge Application Collaborative Solution (adapted from [39])

Cloud-edge-terminal application collaboration, enabling business integration, as illustrated in Figure 6 [39], is an application collaboration solution proposed by Huawei Cloud Applications. Through cloud development and end-to-end deployment, the app can be developed quickly, reducing the difficulty of terminal-to-edge app development and deployment, and quickly responding to business needs. Simultaneously, the multiple deployment method can shield the difference of the terminal-to-edge hardware platform. By advancing the realization of application collaborative technology, we can provide a unified development framework and development platform to achieve centralized management, control application deployment, and reduce security risks.

4.4 Task scheduling collaboration

Because all calculations are performed to achieve tasks, the ultimate goal of improving the collaboration model is to achieve task collaboration. Dai et al. [37] proposed a collaborative scheduling mechanism, a novel method of data processing mechanism in a cloud-edge computing environment. As illustrated in Figure 7, after inputting the task information, through information integration, the task collaborative scheduling management system is responsible for a series of processes such as task analysis, algorithm selection and data management. After resource coordination, the final task will be allocated to different data processing applications for implementation and execution. In this process, existing resources can be coordinated, and system efficiency can be

maximized using this collaborative mechanism.

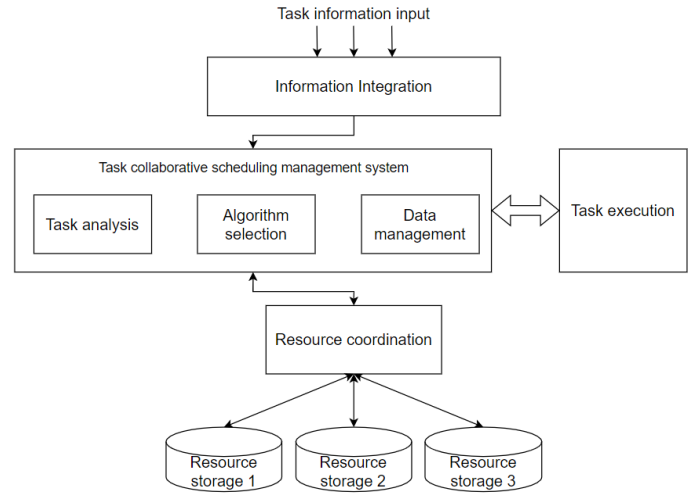


Figure 7. Implementation Mechanism of Dynamic Task Collaborative Scheduling

Using task scheduling collaboration, the power consumption of task execution will be greatly reduced, and the stability of resource scheduling will also be enhanced. For the overall cloud-edge collaborative mechanism, it will be a big performance improvement.

5. Discussion

At this stage, the rapid development of emerging AI technologies has largely affected the development direction of related computer fields. It can be stated that the advent of several emerging technologies is the product of the development and transformation of the industrial Internet, and the development of AI has promoted information interaction and performance optimization of the edge device network, and has also promoted the deep integration of cloud and edge computing, i.e., the cloud-edge collaborative intelligence mentioned in this article is a product under the assumption that conventional cloud computing is not enough to support the current development process.

We introduced core components and framework architecture that are required to realize cloud-edge collaborative intelligence. The development of these components will further promote the improvement of the comprehensive performance of cloud-edge collaborative intelligence. Simultaneously, AI and cloud-edge collaborative intelligence complement each other, and the gradual maturity of cloud-edge collaborative intelligence will also improve surrounding related technologies and help more AI designs become a reality.

6. Summary and outlook

In this study, we summarized the current cloud-edge collaborative intelligence technology and its related branches, combed key issues, introduced the basic concepts and development process of related technologies, and explained their degree of integration with the field of AI. We summarized a few researches on framework design and development methods, and in conclusion, also listed four components of cloud-edge collaboration and cited some examples.

As the government and academia pay more attention to this field, the increase in bandwidth and network efficiency, launch of development platforms and frameworks of major companies and software vendors, and improvement of basic hardware facilities, cloud-edge collaborative intelligent technology will break through current limitations. It will gradually apply to all major fields, perfect and realize the current and possible intelligent scenes in the future. Although the current field is still in the early stage of development, and a few theories are still in the conceptual stage, as the technology matures, the author believes that cloud-edge collaborative intelligent technology will surely promote a major change in the industrial internet in the future.

Acknowledgement

This work is supported by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (Grant 21CTAP-C163793-01).

References

- [1] Matt Zwolenski, and Lee Weatherill “The digital universe: Rich data and the increasing value of the internet of things.” *Journal of Telecommunications and the Digital Economy* 2.3 (2014): 47-41.
- [2] Wang Hucheng, et al. “The research status and development trend of 5G network technology.” *Telecommunications Science* 31.9 (2015): 149-155.
- [3] Ejaz Ahmed, et al. “Bringing computation closer toward the user network: Is edge computing the solution?.” *IEEE Communications Magazine* 55.11 (2017): 138-144.
- [4] Jaewon Moon, et al. “Cloud-edge collaboration framework for IoT data analytics” *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE.
- [5] Flavio Bonomi, et al. “Fog computing and its role in the internet of things” *Proceedings of the first edition of the MCC workshop on Mobile cloud computing* (2012).
- [6] Michaela Iorga, et al. “Fog computing conceptual model” (2018).
- [7] Zhao Ziming, et al. “Edge computing: Platforms, applications and challenges.” *Journal of Computer Research and Development* 55.2 (2018): 327.
- [8] Zhang Xingzhou, et al. “OpenEI: An open framework for edge intelligence” *39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE (2019).
- [9] Zhao Longqian “Research on resource allocation and task scheduling methods based on cloud-edge collaborative computing architecture.” *MS thesis*. Hunan University of Technology (2020).
- [10] Katie Costello “Gartner predicts the future of artificial intelligence technology.” *Gartner*, <https://www.gartner.com/cn/information-technology/articles/gartner-predicts-the-future-of-ai-technology>. Accessed 21 November 2021.
- [11] Thomas Rausch, et al. “Towards a serverless platform for edge {AI}” *2nd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 19)* (2019).
- [12] Yen-Lin Lee, Pei-Kuei Tsung, and Max Wu “Technology trend of edge AI” *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. IEEE.
- [13] Zhou Zhi, Yu Shuai, and Chen Xu “Edge intelligence: A new paradigm for edge computing and artificial intelligence fusion.” *Big Data* 5.2 (2019): 0.
- [14] Kang Yiping, et al. “Neurosurgeon: Collaborative intelligence between the cloud and mobile edge.” *ACM SIGARCH Computer Architecture News* 45.1 (2017): 615-629.
- [15] Zhang Xingzhou, Lu Sidi, and Shi Weisong “Research on collaborative computing technology in edge intelligence.” *Artificial Intelligence* (2019): 5.7.
- [16] Wang Qianqian “Research on security and privacy protection technology in edge computing.” *Journal of Jinling Institute of Technology* (2020).
- [17] Voigt, Paul, and Axel Von dem Bussche “The eu general data protection regulation (gdpr).” *A Practical Guide, 1st ed. Cham: Springer International Publishing*: 10 (2017): [3152676](https://doi.org/10.1007/978-3-319-52676-6).
- [18] OpenSource cloud alliance for induseRy(OSCAR) “Nine application scenarios of cloud computing and edge computing collaboration” (2019)..

- [19] Ma Hongyuan, et al. “5G edge computing technology and application prospects.” *Telecommunications Science* 35.6 (2019): 114-123.
- [20] Dong Enran, and Sun Zongzhe “Cloud edge collaboration helps industry digital transformation” *Information and communication technology and policy* 46.5 (2020): 76.
- [21] Lei Bo, et al. “A new solution for edge computing based on cloud, network, and edge integration: Computing power network.” *Telecommunications Science* 35.9 (2019): 44-51.
- [22] Jiang Lintao “Cloud computing, edge computing and computing power network” *Information and communication technology* 14.4 (2020): 4-8.
- [23] Daniel Ramage, and Stefano Mazzocchi “Federated analytics: Collaborative data science without data collection.” *Google AI blog*, <https://ai.googleblog.com/2020/05/federated-analytics-collaborative-data.html>. Accessed 28 November 2021.
- [24] Martín Abadi, et al. “Tensorflow: A system for large-scale machine learning.” *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)* (2016).
- [25] Frank Seide, and Amit Agarwal “CNTK: Microsoft’s open-source deep-learning toolkit.” *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge Discovery and Data Mining* (2016).
- [26] Zou Ying “Research on Distributed Deep Neural Networks for Edge Computing.” *MS thesis*. Huazhong University of Science and Technology (2019).
- [27] Zhu Hongrui, et al. “Overview of distributed deep learning training networks.” *Journal of Computer Research and Development* 58.1 (2021): 98.
- [28] Brendan McMahan, et al. “Communication-efficient learning of deep networks from decentralized data” *Artificial intelligence and statistics*. PMLR (2017).
- [29] Keith Bonawitz, et al. “Towards federated learning at scale: System desig.” *arXiv preprint arXiv:1902.01046* (2019).
- [30] Yang Qiang “AI and data privacy protection: The cracking method of federated learning.” *Journal of Information Security Research* 5.11 (2019): 961.
- [31] Zhou Jun, Fang Guoying, and Wu Nan “Summary of research on federal learning security and privacy protection.” *Journal of Xihua University (Natural Science Edition)* 39.4 (2020): 9-17.
- [32] Yang Geng, and Wang Zhousheng “Research progress of privacy protection in federal learning.” *Journal of Nanjing University of Posts and Telecommunications (Natural Science Edition)* 40.5 (2020): 204-214.
- [33] Li Hongjia, Chen Xin, and Zhou Xu “5G-oriented distributed mobile cloud computing collaborative architecture and management mechanism.” *ZTE Technology* 21.2 (2015): 14-19.
- [34] Liang Guangjun, Wang Qun, Xin Jianfang, et al. “Overview of mobile edge computing resource allocation.” *Journal of Information Security* 6.3 (2021): 227-256.
- [35] ETSI, and MECISG “Mobile edge computing (mec); framework and reference architecture.” *ETSI, DGS MEC3* (2016).
- [36] Sami Kekki, et al. “MEC in 5G networks.” *ETSI White Paper* 28 (2018): 1-28.
- [37] Dai Bingrong, Song Jundian, and Qian Junling “Research on Collaborative Mechanism of Mass Distributed Data Processing in Cloud Computing Environment.” *Diss.* 2013.
- [38] Deng Xiaoheng, et al. “Study on collaboration of edge computing resources based on comprehensive trust.” *Journal of Computer Research and Development* 55.3 (2018): 449-477.
- [39] Huawei Official “Cloud edge application collaboration solution.” *Huaweicloud*, <https://www.huaweicloud.com/zhishi/solution028.html>. Accessed 25 November 2021.

다양한 디스플레이 환경을 위한 적응형 UI 레이아웃 프레임워크

윤진[○], 이은영[○], 이상현[○], 백종문^{○○}

[○]삼성전자 삼성리서치, ^{○○}한국과학기술원

{jinny.yoon, ey928.lee, sh10233.lee}@samsung.com, jbaik@kaist.ac.kr

An Adaptive UI Layout Framework for Various Display Environments

Jin Yoon[○], Eunyoung Lee[○], Sanghyeon Lee[○], Jongmoon Baik^{○○}

[○]Samsung Electronics Samsung Research, ^{○○}Korea Advanced Institute of Science and
Technology

요 약

본 연구에서는 다양한 디스플레이 환경에 따라 적응형으로 동작하는 UI 뷰를 생성하기 위해 필요한 요소들을 제안한다. 그 동안, 모바일 분야와 웹 분야에서는 모바일 혹은 데스크탑을 위한 디스플레이 환경을 중심으로 적응형 레이아웃의 생성 기법을 발전시켜왔다. 하지만, 디스플레이가 다양화 · 대형화되며, 대형 TV · 키오스크 · 전자칠판 등의 디스플레이 환경에서도 사용할 수 있는 적응형 레이아웃 생성 기법을 필요로 하게 되었다. 이에 본 연구에서는 여러 디스플레이 환경에서 적응형 레이아웃을 생성하기 위해 필요한 기술적인 요소들을 확인하였다. 그리고, 타이젠 오픈소스 프로젝트에 본 연구에서 제안하는 요소들을 구현한 후, 그 구현 결과물에 대한 평가를 진행하였다. 첫번째 평가로 디자이너의 디자인 의도대로 동작하는 적응형 UI 레이아웃을 개발하기 위해 필요한 비용을 확인하였고, 두번째 평가로 적응형 UI의 도입으로 개발 생산성이 기존 대비 90% 이상 좋아지는 것을 확인하였다.

1. 서 론

디스플레이는 다양한 형태의 제품에 탑재되고 있다. 집집마다 쉽게 접할 수 있는 모바일 · 데스크탑 · TV 뿐만 아니라 키오스크나 전자칠판 등 다양한 제품에서 다양한 디스플레이 환경을 사용하고 있다. 그리고, 각각의 디스플레이 환경은 화면을 구성하는 픽셀의 개수, 화면의 비율, 그리고 디스플레이의 물리적인 크기가 다르다. 이렇게 서로 다른 디스플레이 환경에서 UI 뷰들을 재사용할 수 있다면 소프트웨어 개발 비용을 낮출 수 있다.

사실, 그 동안 디스플레이 환경에 맞춰 유려하게 보여지는 UI 뷰를 체계적으로 개발하기 위해 여러 가지 방법들을 시도하였다. 소프트웨어 프로젝트 라인 개발 방법에서는 개발 단계를 도메인 엔지니어링 단계와 애플리케이션 엔지니어링 단계로 나눈다. 그리고, 도메인 엔지니어링 단계에서는 여러 제품에서 재사용할 수 있는 결과물을 개발하고, 애플리케이션 엔지니어링 단계에서는 제품별로 파생되어 나오는 결과물을 개발한다[1]. 따라서, 특정 제품을 위한 소프트웨어 결과물은 도메인 엔지니어링 단계의 결과물과 애플리케이션 엔지니어링 단계를 결과물을 적절하게

조합해야 한다. 하지만, 이러한 방식으로는 디스플레이의 종류가 많아질수록 애플리케이션 엔지니어링 단계에서의 UI 뷰 개발 비용도 상당히 늘어날 수밖에 없다.

그래서 다이나믹 프로젝트 라인 엔지니어링에서는 다양한 디스플레이 환경을 위한 복수의 UI 뷰들을 한꺼번에 시스템에 탑재한다. 시스템은 런타임 중에 디스플레이 환경을 파악하고, 그에 적합한 UI 뷰를 선택하여, 화면에 표현하는 방법을 결정한다[2]. 비록 여전히 디스플레이 환경마다 UI 뷰를 개발해야하지만, 디스플레이 환경 간에 바이너리 문치를 공유할 수 있는 가능성을 열어 두었다.

그리고 모바일 시장이 급성장하며, 모바일 제품군의 다양한 디스플레이 환경에서 UI 뷰를 재사용할 수 있는 방법들이 나왔다. 아이폰의 Auto Layout이나 안드로이드의 Constraint Layout은 상대 좌표를 기반으로 여러 디스플레이에 적응할 수 있는 UI 뷰를 개발할 수 있다는 가능성을 제공하였다. 하지만, 모바일 제품군에서의 방법들은 ‘Mobile first’ 원칙에 따라 모바일 친화적인 디스플레이 환경에 집중한다[3][4][5]. 따라서, 그 외의 디스플레이 환경에 모바일 제품군의 방법을 그대로 적용하기에는 한계가 있다.

웹 진영에서는 모바일 제품군의 디스플레이 환경뿐만 아니라 데스크탑의 디스플레이 환경까지도 고려한다. 그래서, 모바일 제품군에서 다루는 디스플레이 환경보다 넓은 범위의 디스플레이 환경에서 유려하게 보여지는 UI 뷰 제작 방법들을 지원한다. 하지만, 앞서 언급한 대로 대형 TV · 키오스크 · 전자칠판 등의 디스플레이 환경까지는 고려하지 않는다. 게다가, 다른 제품 환경과는 다르게 데스크탑 환경에서는 최종사용자가 브라우저 윈도우의 높이와 폭을 자유롭게 늘이거나 줄이면서 UI 뷰를 보정할 기회를 가진다. 하지만, 플윈도우 기반으로 동작하는 제품 환경에서는 최종 사용자들은 UI 뷰를 보정할 기회가 없다. 따라서, 이런 플윈도우 기반 제품 환경에서 최종사용자의 보정없이 유려하게 보여지는 방법이 필요하다.

따라서, 본 연구에서는 모바일 제품군이나 데스크탑의 디스플레이 환경을 넘어 대형 TV · 키오스크 · 전자칠판 등에 부착된 디스플레이 환경에서도 유려하게 보여지는 적응형 UI 뷰를 개발하기 위한 요소를 확인하고자 한다. UI 뷰 개발자는 하나의 디스플레이 환경을 기준으로 개발한 결과물일지라도 다수의 디스플레이 환경에 탑재하여 사용할 수 있어야 한다. 여기서 관건은 디스플레이 환경이 바뀔 때마다 적응형 UI 뷰에 대한 디자이너의 디자인 의도를 만족시켜야 한다는 점과 UI 뷰 레벨에서 코드 수정을 최소로 해야한다는 점이다. 다음 장에서는 이런 점들을 달성하기 위해 필요한 요소들을 확인하고자 한다.

2. 기존의 문제해결 방법

앞서 설명한 바와 같이 기존에도 다양한 디스플레이 환경에서 유려한 UI 뷰를 표현하기 위한 여러 시도가 있었다. 그리고, 이러한 시도는 최소한의 노력과 비용으로 다양한 디스플레이 환경에서 유려하게 동작하는 UI 뷰를 지원하려는 공통된 목표를 가진다. 비록, 디자이너나 개발자가 특정 디스플레이 환경 기준으로 동작하는 결과물을 만들지만, 그 결과물이 다양한 디스플레이 환경에서도 유려하게 보여지기를 희망한다. 그리고 이왕이면 UI 뷰의 코드를 전혀 수정하지 않거나 최소한의 수정만으로 원하는 결과를 얻길 희망한다. 하지만, 공짜 점심은 없다. UI 뷰 코드에 투자하는 노력 대신 UI Framework(이하 UIFW)에서 UI 뷰들이 적응형으로 동작하도록 지원해야 한다. 이를 위해 UIFW는 런타임 중에 디스플레이 환경을 파악하고, 해당 환경에서 가장 유려하게 잘 보여줄 수 있는 방식을 선택한 후 디스플레이에 UI 뷰를 표현한다. 이런 과정을 거쳐 최종사용자는 기능적 · 심미적으로 불편없이 UI 뷰를 사용한다.

그렇기 때문에 다양한 디스플레이 환경에 대응하기 위한 목적으로 UIFW에서는 UI 뷰 개발에 사용할 수 있는 다양한 기능을 제공한다. 그리고, UIFW에서는

이런 기능을 제공하기 위해 다양한 디스플레이 환경에 일관되게 적용할 수 있는 알고리즘이나 원칙을 준비한다. UI 뷰 개발자는 콘텐츠를 표현하고자 하는 의도에 따라 UIFW에서 지원하는 기능을 취사선택하여 개발한다.

본 연구에서는 UIFW이 지니고 있는 수많은 기능 중 다양한 디스플레이 환경에 적응하기 위한 목적을 가진 기능에 관심을 두고자 한다. 각기 다른 제품 환경에서 동작하는 UIFW들은 각각 관심있는 디스플레이 환경에 적응하기 위한 기능을 제공한다. 그래서, UIFW 간에는 유사한 기능들도 있고 차별화되는 기능들도 있다. 본 연구에서는 여러 제품의 UIFW 중 Bootstrap, CSS3, EFL, Native Android UI에서 특징적인 기능을 살펴보고자 한다.

Bootstrap은 트위터에서 개발하기 시작한 HTML, CSS, JS 기반의 UIFW이다[6]. 웹 분야에 존재하는 다양한 디스플레이 환경에서 유연하게 동작하는 UI 뷰를 지원하기 위한 다양한 기반 기능을 제공하는데 집중하고 있다. 그리고, 본 연구에서 주목하는 기능은 Grid system이다[7].

Grid system은 디스플레이 환경에 따라 좌우 영역에 여백을 배치하고, 가운데 영역에 그리드 형태의 콘텐츠를 배치한다. 좌우에 배치되는 여백의 너비는 디스플레이 환경에 따라 조절된다. 그래서 가운데 영역에 배치되는 콘텐츠의 너비는 변하지 않게 된다. 이를 통해 디스플레이 환경이 달라지더라도 콘텐츠 영역의 크기는 유지하는 것이다. 만약, 콘텐츠 영역이 바뀌지 않는다면, 애초에 디자이너가 의도한 바를 그대로 표현할 수 있다.

이를 위해 Grid system에서는 디스플레이 환경을 Extra small, Small, Medium, Large, Extra large, Extra extra large 등 6개 그룹으로 구분 짓는다. 그리고, 각각의 그룹 내에서 디스플레이 환경이 바뀌더라도, 콘텐츠 영역의 너비가 바뀌지 않도록 좌우 여백을 늘이거나 줄인다. 하지만, Extra extra large 그룹에서는 1400 px을 기준으로 그것보다 큰 디스플레이 환경을 한꺼번에 처리한다. 따라서, 모바일 환경이나 데스크탑 환경 뿐만 아니라 대형 디스플레이 환경을 위해 새롭게 그룹을 정의할 필요가 있다.

더불어, 디스플레이의 가로와 세로의 비율이 바뀌거나 디스플레이의 가로 모드 혹은 세로 모드처럼 모드가 바뀌는 경우에 대한 고려도 필요하다. 이러한 변화에는 위에서 언급한 Grid system의 그룹만으로는 유연하게 대응할 수가 없다. 왜냐하면, 콘텐츠 영역의 가로세로비 자체가 달라지기 때문이다. 좌우 여백에 의해 콘텐츠 영역의 너비가 고정된다 하더라도 콘텐츠의 가로세로비가 달라지면, 필연적으로 콘텐츠 영역 내의 아이템들을 재배치해야 한다.

바로 이러한 디스플레이 환경 변화에 대응하기 위해 CSS 3의 Flexible box layout을 차용할 수 있다. Flexible

box layout은 반응형 웹을 위해 flex-start, flex-end, center, stretch, baseline, space-between, space-around, space-evenly 등의 다양한 정렬 방식을 지원한다[8]. 이런 정렬 방식을 활용하여 디자이너가 의도하는 형태의 디자인 결과물을 표현할 수 있다. UI 뷰 개발자는 디자이너의 디자인 의도에 부합하는 정렬 방식을 선택하여 UI 뷰를 개발한다. 그러면, 디스플레이 환경이 변하더라도 사전에 설정한 정렬 방식에 따라 아이템들은 제 위치를 찾을 수 있다.

하지만, 디자이너의 다양한 디자인 의도를 Flexible box layout에서 지원하는 정렬 방식만으로 표현할 수 없는 경우가 있다. 본 연구에서 관심을 갖고 있는 UI 뷰들의 경우도 마찬가지이다. 특히, Flexible box layout 내에 다수의 아이템들을 멀티라인으로 구성하는 경우, 마지막 줄에 위치하는 아이템들을 디자이너의 의도대로 정렬하기가 어렵다. 이를 위해서는 Sentinel 아이템을 세우는 추가 개발이 필요하다.

타이젠 플랫폼에서는 UIFW로 Enlightenment Foundation Libraries (이하 EFL)를 사용하고 있다[9][10]. 그리고, EFL에서는 다양한 디스플레이 환경에 적응하기 위해 base scale과 scale factor 개념을 사용한다. base scale은 디스플레이 환경을 특정짓는 값이고, scale factor는 심미적인 기준을 충족하기 위한 보정값이다.

$$base\ scale = \frac{dpi}{90} \times product\ factor$$

base scale은 디스플레이의 물리적 사이즈와 해상도에 따라 결정되는 dots per inch(이하 dpi)를 기반으로 만든다. 그리고 dpi 값을 임의의 상수 90으로 나누고 product factor를 곱해준다. 이 product factor의 기본값은 '1'이지만, 디스플레이 환경에 따라 심미적인 관점에서 다른 값으로 바꿀 수 있다. 이렇게 결정된 base scale을 기준으로 디자이너는 UI 뷰를 디자인하고, 앱 개발자는 UI 뷰를 개발한다.

한 디스플레이 환경에서 개발된 UI 뷰를 다른 디스플레이 환경에서도 사용하길 희망할 수도 있다. 그럴 때에는 디자인 및 개발 과정에서 사용한 UI 뷰의 base scale 값과 런타임에서 사용하는 디스플레이 환경의 base scale 값이 다를 수 있다. 이럴 경우 다음과 같은 수식을 사용으로 추상적인 크기를 물리적인 크기로 치환한다.

$$physical\ size = abstract\ size \times \frac{base\ scale\ in\ display\ env.}{base\ scale\ in\ UI\ view\ dev.} \times scale\ factor$$

UI 뷰에서는 콘텐츠의 위치와 크기를 표현하기 위해

추상적인 크기를 사용한다. 그리고 해당 값을 물리적인 크기로 변경하기 위해 두 가지 값을 곱해준다. 첫번째 값은 런타임 중인 디스플레이 환경의 base scale과 UI 뷰 개발 시점에서 사용한 디스플레이 환경의 base scale 간의 비율이다. 두번째 값은 디자이너의 심미적인 관점에 따라서 결정할 수 있는 scale factor이다. 이 두 값을 곱하여 물리적인 크기를 확정 짓는다. 이러한 일련의 결정 과정을 EFL UIFW에서 처리하고 있다.

하지만, 이 수식에는 경험에 의해 결정되는 값이 두 개 (base scale, scale factor)가 있다. 그렇기 때문에, 새로운 디스플레이 환경에 맞는 최적의 값을 찾기 위해 상당한 노력을 기울여야 한다. 따라서, 경험적인 요소로 결정되는 인자의 수를 줄일 필요가 있다.

안드로이드에서도 추상적인 단위를 이용하여 디자인 단계 및 앱 개발 단계에서 개별 아이템의 위치와 크기를 정의한다. 타이젠의 경우와 마찬가지로 디자이너와 개발자는 추상적인 단위를 사용하고, 안드로이드의 UIFW에서 추상적인 단위를 물리적인 단위로 치환한다. 추상적인 단위를 물리적인 단위로 치환하기 위해 아래의 수식을 활용한다.

$$px = dp \times \left(\frac{dpi}{160}\right)$$

안드로이드에서 사용하는 추상적인 단위는 density-independent pixels (이하 dp)이다[11]. 여기에 디스플레이 환경에 의해 결정되는 dpi를 임의의 상수 160으로 나눈 값을 곱해준다. 이렇게 디스플레이 환경에 의해 결정되는 px은 아래 표로 예상할 수 있다. 아래 표로 각각의 dpi에서 1 dp에 해당하는 px 값을 확인할 수 있다.

표 1. dpi에 따라 변하는 1 dp의 px 값

dpi	120	160	240	320	480	640
px	0.75	1.00	1.50	2.00	3.00	4.00

하지만, 위의 dp만 가지고는 심미적인 기준을 충족시킬 수 없는 경우도 있다. 따라서, 위의 수식에 심미적인 기준을 충족할 수 있는 인자를 덧붙일 필요가 있다.

3. 제안 기법

다양한 디스플레이 환경에서 유려하게 보여지는 UI 뷰를 생성하기 위해서는 UIFW의 지원이 필요하다. 그리고, 그림 1에서 적응형 UI 뷰를 생성하는 과정에서 참여하는 주요 요소들을 확인할 수 있다. 그림 1에서 중간에 위치하는 UIFW를 중심으로 상단에는 입력 인자가 있고, 하단에는 출력 인자가 있다.

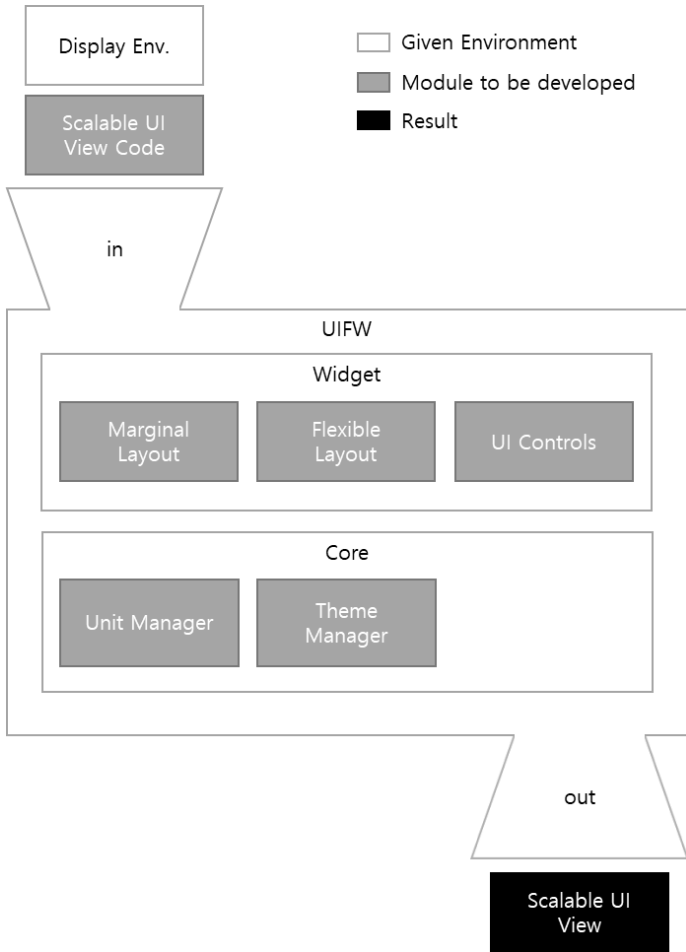


그림 1. 적응형 UI 뷰를 지원하기 위한 시스템

입력 인자로는 2가지 요소가 있다. 첫번째 요소는 디스플레이 환경이다. 이는 제품에 따라 확정되는 디스플레이의 물리적 크기, 해상도, 그리고 가로세로비를 의미한다. 디스플레이 환경은 제품에 따라 주어지는 확정된 요소이다. UIFW에서는 이 값을 근거로 UI 뷰를 출력하게 된다.

또 다른 입력 인자는 다양한 디스플레이 환경에서 유려하게 보여지길 희망하는 UI 뷰의 코드이다. 그리고 이 UI 뷰의 코드에서 그림 1의 UIFW 요소들을 직간접적으로 사용하게 된다. 위젯과 같은 요소들은 UI 뷰에서 명시적으로 사용하여 디자이너의 특별한 디자인 의도를 달성하는데 활용된다. 그리고, 코어 요소들은 UIFW에 내재되어 디스플레이 환경과 UI 뷰의 상태에 따라 간접적으로 사용된다.

일반적으로 UIFW에는 다수의 레이어들이 존재하지만, 여기서는 본 연구에서 관심을 가지고 있는 적응형 UI 뷰를 생성하기 위한 핵심적인 구성요소만을 표현한다. 그림 1의 위젯 레이어에는 앞서 설명했다시피 UI 뷰에서 직접 사용하는 기능들을 담고 있다. Flexible Layout과 Marginal Layout은 UI 뷰에서 좀 더 유연하게 레이아웃을 관리할 수 있도록 지원한다. 여기서 언급하지 않은 레이아웃 위젯들도 UIFW에는 준비되어 있지만, 여기서는 적응형 UI 뷰에 특화된 레이아웃만 설명하기로 한다. 그리고 위젯 레이어에는 또 다른 구성요소인 UI Control도 있다. 여기서 언급한 UI Control은 버튼, 라디오 버튼, 체크박스, 이미지, 텍스트 등 기본적인 UI 구성요소를 의미한다. 이 요소들도 모두 유연하게 동작할 수 있도록 준비되어야 한다.

코어 레이어에는 상위 레이어인 위젯 레이어 혹은 UI 뷰에 포괄적으로 영향을 주는 기능들이 위치하고 있다. 첫번째 구성요소는 Unit manager로 UI 뷰에서 사용한 추상적인 단위를 디스플레이 환경에 따라 물리적인 단위로 치환해주는 역할을 한다. 두번째 구성요소는 Theme manager로 디스플레이 환경에 따라 적합한 UI Control을 선택하는 역할을 담당한다.

출력 인자로는 디스플레이에 출력되는 UI 뷰가 있다. 이 UI 뷰는 확정된 디스플레이 환경에서 유려하게 보여져야 한다. 여기서 유려하다는 것은 디자이너의 디자인 의도를 정확히 반영하고 있다는 것을 의미한다.

지금까지 그림 1을 통해 적응형 UI 뷰를 위해 필요한

□ 디스플레이 ■ 콘텐츠

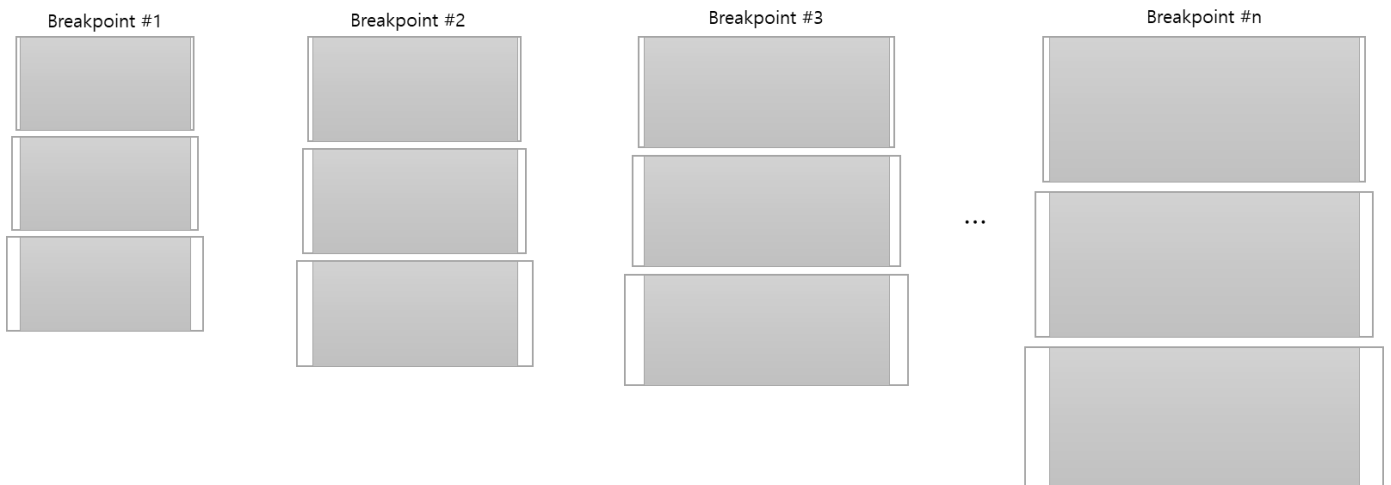


그림 2. Breakpoint 그룹별로 처리하는 디스플레이 환경

요소를 전체적으로 살펴보았다. 이제 주요 요소들을 하나씩 살펴보기로 한다.

앞서 설명한 것과 같이 Bootstrap의 Grid system은 콘텐츠 영역 크기에 대한 변화를 완충하는 기능을 수행한다. 그리고, 본 연구에서는 Grid system의 다른 기능들 보다 이 완충 기능에 집중하고자 하며, 해당 기능을 수행하는 모듈을 Marginal Layout으로 부르기로 한다. 그림 2에서는 이 Marginal Layout이 Breakpoint 그룹별로 동작하는 모습을 확인할 수 있다.

한 Breakpoint 그룹 내에서는 디스플레이가 커지면 커질수록 좌우 여백의 폭이 넓어지지만 콘텐츠 영역의 폭은 변하지 않는 것을 확인할 수 있다. 이를 통해 콘텐츠 영역 내부에 위치하는 수많은 아이템들의 위치와 크기를 보존할 수 있다. 애초에 콘텐츠 영역은 디자이너의 디자인 의도에 따라 개발되었으므로, 해당 영역이 변하지 않도록 최대한 유지한다.

하지만, 디스플레이가 커진다고 무한정 좌우 여백을 늘릴 수는 없다. 그렇기 때문에 복수 개의 Breakpoint를 준비한다. 한 Breakpoint 그룹에서 처리하는 디스플레이 환경은 상호 유사한 크기 수준이지만, 해당 크기 범위를 벗어나면, 다음 Breakpoint 그룹에서 해당 디스플레이를 처리한다. Breakpoint 번호가 높아지면 높아질수록 가운데 콘텐츠 영역의 폭도 넓어진다. 콘텐츠 영역의 폭이 넓어지니 그 안에 위치하는 아이템의 크기나 위치도 달라진다. 하지만, 동일 Breakpoint 그룹 내에서는 비슷한 수준의 디스플레이 환경을 처리하기 때문에 콘텐츠의 폭은 바뀌지 않는다.

따라서, 한 Breakpoint 그룹에 속하는 다양한 디스플레이 환경들 중 오직 하나만 선택하여 디자인 · 개발 · 검증을 수행할 수 있다. 비록, 좌우 여백의 크기는 바뀌지만, 콘텐츠 영역은 변하지 않기 때문에 이러한 일이 가능하다. 물론, Breakpoint의 개수만큼 디자인 · 개발 · 검증 과정이 필요하지만, 디스플레이 환경의 개수에 비하면 그 비용이 낮다.

Flexible layout 곧 Flexible box layout은 콘텐츠 영역의 크기가 변경되는 상황에서 유용하게 사용된다. Marginal layout은 동일 Breakpoint 그룹에 속하는 경우에 한정하여 콘텐츠 영역의 크기를 보존한다. Breakpoint 그룹이 변경되면, 콘텐츠 영역의 크기도 변경된다. 특히, 가로 모드와 세로 모드 간의 차이처럼 극적인 변화도 있다. 디스플레이 환경이 다양한 만큼 콘텐츠 영역의 크기 변화를 막을 수 없기 때문에, 크기 변화에 유연하게 대응할 수 있는 방법이 필요하다.

Flexible layout에서는 UI 뷰에 여러 가지 정렬 방식을 제공하며, 각 정렬 방식이 다양한 디스플레이 환경에서도 유연하게 대응하도록 지원한다. Flexible layout에서는 기본적으로 그림 3에서 보여지고 있는 정렬 방식을 지원한다. 상단 3개의 정렬 방식인 flex-start, flex-end, center은 비교적 널리 사용되고 있는 정렬 방식이다. 그리고 거기에 하단 3개의 정렬 방식인

space-between, space-around, space-evenly를 추가한다. space-between에서는 좌측 첫번째 아이템을 좌측 끝에 붙이고, 우측 첫번째 아이템을 우측 끝에 붙인다. 그리고, 모든 아이템 간의 거리를 동일한 크기로 설정한다. space-around는 모든 아이템들이 자기만의 여백을 좌우 영역에 확보하도록 한다. 여기서는 한 아이템의 좌측 혹은 우측의 여백을 다른 아이템의 우측 혹은 좌측의 여백과 공유하지 않는다. 따라서 두 아이템 사이에 위치하는 여백은 그렇지 않은 여백보다 2배 넓다. 좌측에서 첫번째 아이템의 좌측 여백과 우측에서 첫번째 아이템의 우측 여백은 상대적으로 좁다. space-evenly는 모든 여백의 크기를 동일하게 지정한다. 좌측이나 우측 끝에 위치하는 여백이나 아이템 사이에 위치하는 여백에 차이를 두지 않고 동일한 크기로 맞춘다.

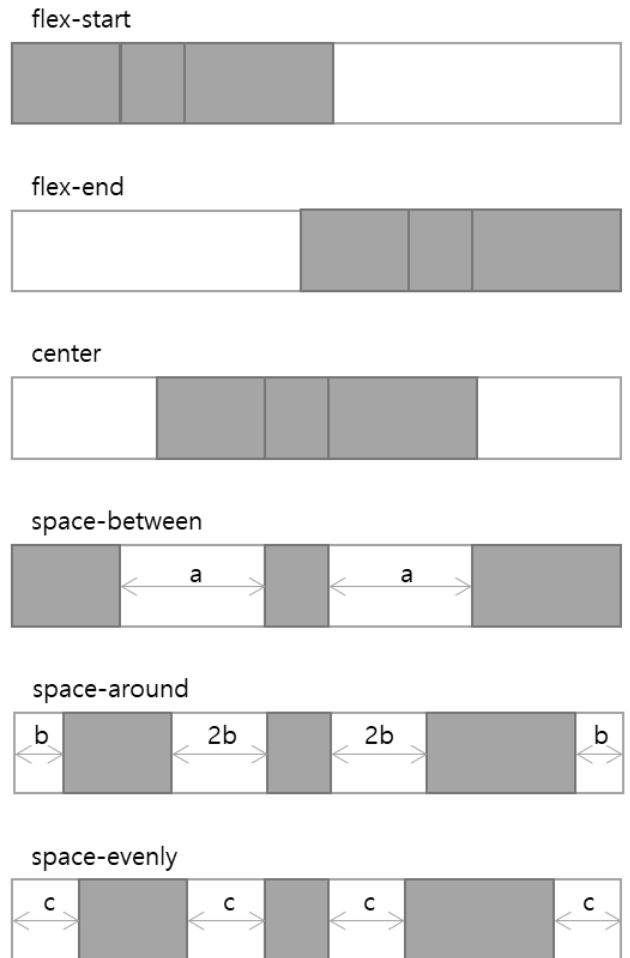


그림 3. Flexible box layout에서 지원하는 정렬 방식

그림 3에서는 가로축 기준으로 정렬 방식을 표현하였지만, 세로축 기준으로도 마찬가지로 방식을 적용할 수 있다. 더불어 아이템의 개수가 늘어나서 여러 줄에 걸쳐 표현해야 하는 경우에도 마찬가지로 방식을 적용할 수 있다.

물론, Flexible layout 만으로 디자이너의 다양한 디자인 의도를 표현할 수 없다. 여기서 언급하지는 않았지만, UI Toolkit에서 일반적으로 제공하는 Linear

layout, List layout, Grid layout, Absolute layout 등 다양한 레이아웃과 조합하여 레이아웃을 구성해야 한다. 그리고, 그렇게 하기 위해서는 레이아웃 간에도 레이아웃 자체 혹은 레이아웃 내에 위치하는 아이템의 위치나 크기에 대한 정보를 주고 받을 수 있어야 한다.

앞서 설명한 Marginal layout과 Flexible layout으로 레이아웃 수준에서의 대응이 가능하다. 하지만, 레이아웃만 보정한다고 하여 심미적인 관점에서 유려한 UI 뷰를 가질 수는 없다. 왜냐하면, 레이아웃 내에 위치하는 개별 아이템들이 각각의 디스플레이 환경에서 적합한 크기로 보여져야 하기 때문이다. 디스플레이의 물리적인 크기가 동일하지만 해상도가 보다 촘촘해지면, 동일 아이템을 더 많은 픽셀로 표현하는 편이 보기 좋다. 만약 아이템을 동일 픽셀로 표현하면, 아이템의 크기는 보다 작아져 시인성이 떨어진다.

디자이너가 특정 디스플레이를 기반으로 디자인 작업을 하고, 특정 디스플레이에서만 동작하는 UI 뷰를 만든다면, 아이템의 위치나 크기를 px 단위로 명시하여 디자인하고 개발해도 문제없다. 하지만, 하나의 UI 뷰를 다양한 디스플레이 환경에서 재사용해야 한다면, 디자인 시안에 px 단위를 사용하지 않는다. 그 대신 UIFW에 의해 px 단위로 치환할 수 있는 추상적인 단위를 사용한다. 그리고 그런 디자인 시안을 기본으로 UI 뷰를 개발하는 개발자도 디자인 시안에 명시된 추상적인 단위를 사용하여 앱을 개발한다. 개발자가 개발하는 코드에는 px 단위가 단 하나라도 언급되지 않는다.

$$px = abstract\ size \times \left(\frac{dpi}{160}\right) \times scale\ factor$$

위의 수식으로 추상적인 크기를 물리적인 크기로 치환할 수 있다. 추상적인 크기에 바로 곱해지는 값은 안드로이드에서 사용하고 있는 dpi이다. 하지만, 거기에 타이젠 플랫폼에서 사용하고 있는 scale factor를 추가한다. scale factor는 산술적으로 결정되는 값을 심미적인 관점으로 보정하기 위한 인자이다. 사실, 물리적으로 동일한 디스플레이라고 할지라도 제품에 따라 디스플레이 환경은 달라질 수 있다. 예를 들어, 디스플레이와 사용자 간의 거리나 터치 스크린으로 동작하는지 유무 혹은 디스플레이에 주로 표현되는 콘텐츠의 종류 등에 따라 UI 뷰는 달리 표현되어야 한다. 따라서, 최종적으로 아이템 크기를 보정할 scale factor와 같은 수단이 필요하다.

추가로, 그림 1에서 언급한 요소 중 하나가 UI Control이다. UI Control은 UI Toolkit에서 제공하는 온갖 종류의 위젯을 의미한다. 이런 위젯들은 개발 편의성이나 UX의 일관성을 위해 UI 뷰에서 널리 사용한다. 그렇기 때문에, 모든 UI Control들도 다양한 디스플레이 환경에서 유연하게 동작해야 한다. 이를 위해서는 UI Control을 개발할 때에 앞서 설명한 유연한

레이아웃이나 추상적인 단위를 적절히 사용할 필요가 있다.

그리고 마지막 요소로 Theme manager가 있다. Theme manager는 디스플레이 환경에 적합한 UI Control을 선택하는 역할을 담당한다. 버튼, 라디오 버튼, 체크박스, 드롭 다운, 슬라이더, 픽커 등의 UI Control들은 오직 한 벌만 있는 것이 아니다. 디스플레이 환경 혹은 제품의 특성에 따라 몇 벌씩 존재할 수도 있다. 그렇기 때문에 여러 벌 존재하는 UI Control 중 현재 디스플레이 환경에 적합한 하나를 선택하여 보여줘야 한다. 그 역할을 하는 것이 Theme manager이다.

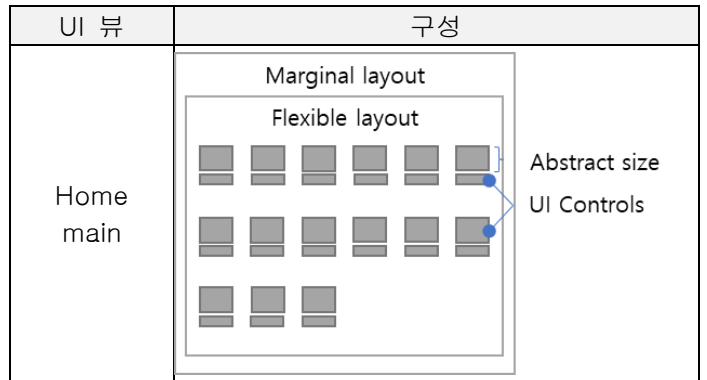
일반적으로 UI 뷰를 개발하는 단계에서는 버튼, 라디오 버튼, 체크박스 등의 UI Control을 명시적으로 결정하지만, UI Control의 Theme까지 결정하지는 않는다. UI Control의 Theme는 제품 단위로 결정되거나 최종 사용자의 선택에 의해 결정된다. UI 뷰에서는 이렇게 결정된 Theme에 따라 시스템에 의해 주어지는 UI Control을 사용하게 된다.

이상과 같이, 다양한 디스플레이 환경에서 유연하게 동작하는 UI 뷰를 지원하기 위해 Marginal layout, Flexible layout, Unit manager, UI controls, Theme manager 등의 요소를 구비해야한다. 이런 요소들을 기반으로 개발된 UI 뷰들이 다양한 디스플레이 환경에서 실제로 유연하게 보여지는지 확인이 필요하다.

4. 비교 실험

Tizen 6.5 Headed 환경에서 동작하는 수많은 UI 뷰들 중에서 대표적인 7가지의 뷰를 선정한다. 사실, 그 외의 뷰들은 대표 뷰들과 유사한 구조를 가지고 있기 때문에, 모든 뷰를 다 다룰 필요는 없다. 따라서, 7가지 뷰들을 앞서 설명한 요소들을 충실히 활용하여 개발하고 평가하고자 한다.

표 2. 대표적인 UI 뷰에 본 연구에서 제안하는 요소들을 적용한 사례



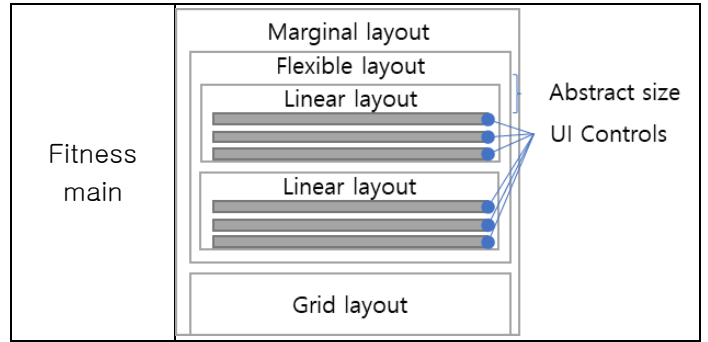
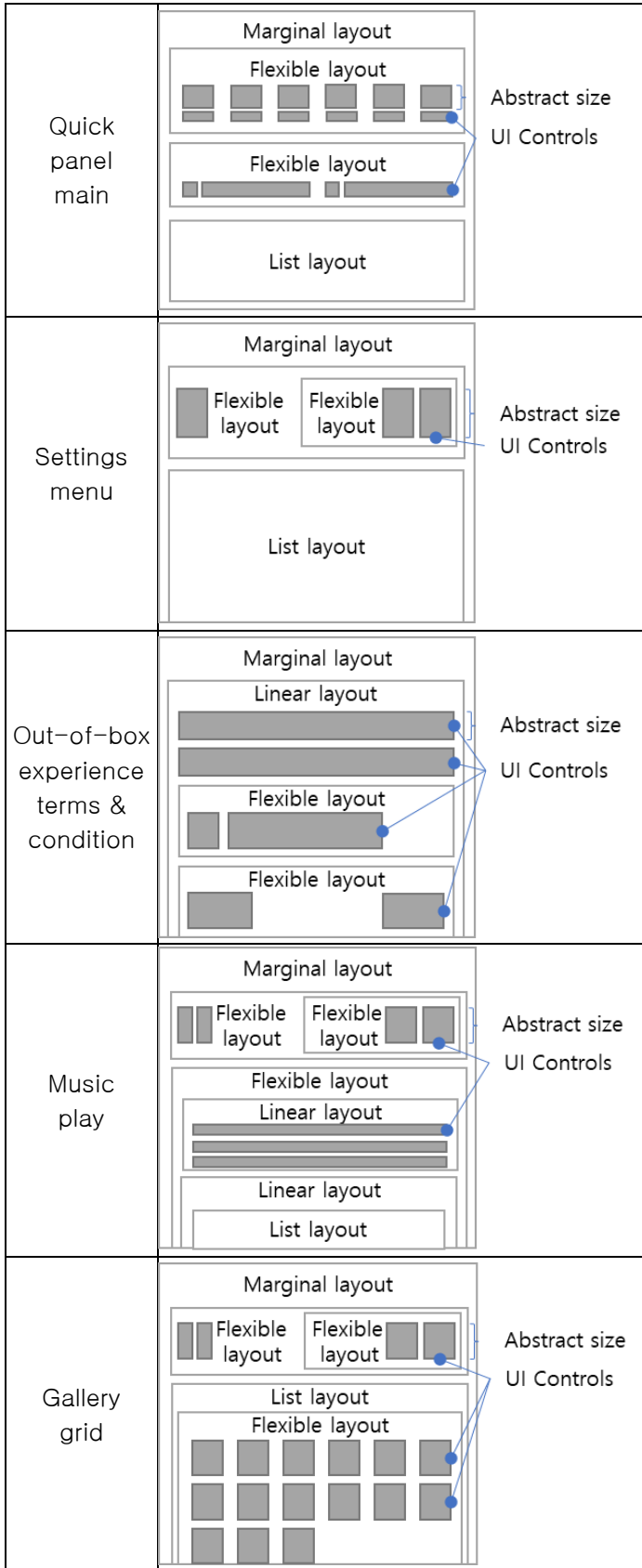


표 2에서 보는 것과 같이 대표적인 UI 뷰들은 최상위 레이아웃으로 Marginal layout을 사용한다. 그리고 그 하위에 Flexible layout, Linear layout, List layout, Grid layout을 중첩으로 사용한다. Linear layout, List layout, Grid layout은 일반적인 UI Toolkit에서 널리 사용하고 있기 때문에 따로 언급하지 않았지만, 상대 좌표 기반으로 아이템을 유연하게 배치할 때 사용한다.

그리고 표 2에 존재하는 질은 색 사각형은 다양한 UI Control들을 의미한다. 각각의 UI Control들은 버튼, 라디오버튼, 체크박스, 드롭 다운, 슬라이더, 픽커, 이미지, 라벨 중 하나이다. 여기서 언급된 UI Control들은 모두 다양한 디스플레이 환경에서 유연하게 동작한다.

대표적인 UI 뷰의 개발 결과물들을 표 3에 명시한 다양한 디스플레이 환경에서 검증한다. 각 디스플레이의 가로 모드와 세로 모드 모두를 검증에 활용한다. 디스플레이 환경마다 디자이너의 디자인 의도대로 UI 뷰들이 표현되고 있는지 확인한다. 여기서 디자인 의도의 달성 여부는 화면에 표시되는 모든 아이템의 위치와 크기가 적합한지 여부로 판별된다.

표 3. 대표 UI 뷰를 테스트한 디스플레이 환경 (Type에 Real은 실물 디스플레이를 의미하고, Emul은 에뮬레이터 환경을 의미)

Type	Resolution	Inch	dpi	Scale factor
Emul	800x480	4.5	207	0.5
Real	1280x720	8	184	2.1
Emul	1280x800	10.1	149	1.1
Emul	1366x768	15.6	100	1.7
Emul	1440x900	19	89	2.2
Emul	1600x1200	19	105	1.8
Real	1920x1080	13	169	3.0
Real	1920x1080	23	96	2.5
Emul	2560x1600	13.3	227	1.4
Emul	3200x1600	29	123	2.5
Emul	3840x1600	40	104	3.4
Emul	3840x2160	40	110	3.5

표 4에서는 표 3에서 언급한 다양한 디스플레이

환경에서 7가지 대표 뷰 중 하나인 쿼드 패널 메인 뷰를 실행한 결과를 확인할 수 있다. HD나 FHD 해상도의 경우에는 실물 디스플레이에서 실행한 결과를 카메라로 촬영하였고, 그 외의 해상도의 경우에는 에뮬레이터에서 실행한 화면을 캡처하였다. 어떤 디스플레이 환경이든 개별 아이템들이 적합한 위치와 크기로 표현되는 것을 확인할 수 있다.

표 4. 다양한 디스플레이 환경에서 쿼드 패널 메인 뷰를 실행한 결과

Display	Portrait	Landscape
800x480		
1280x720		
1280x800		
1366x768		
1440x900		

1600x1200		
1920x1080 13"		
1920x1080 23"		
2560x1600		
3200x1600		
3840x1600		



본 연구에서 제안한 방법을 사용하지 않고, 수작업으로 디자인 의도를 UI 뷰에 반영하던 상황에서의 수정한 LOC를 의미한다. 그리고 신규 구조에 대한 코드 수정량이란, 본 연구에서 제안한 방법을 소스에 적용하기 위해 최초 한 번 수정한 LOC를 의미한다.

표 5에서 보면 알 수 있듯, 개별 앱에 대한 코드 수정량에는 차이가 있다. 하지만, 전체 뷰들에 대한 코드 수정량은 7.8%와 8.7%로 서로 유사하다. 곧, 본 연구에서 제안하는 방법을 최초로 적용하기 위해 드는 비용은 기존 구조에서 디스플레이 환경이 바뀔 때마다 UI 뷰를 수정하는 비용과 유사하다고 볼 수 있다.

그리고, 표 5와 표 6에서 각각의 디스플레이 환경에서 디자이너의 의도대로 보여지는 UI 뷰를 개발하기 위해 필요한 코드 수정량을 확인할 수 있다. 표 5에서는 디스플레이 환경이 바뀌었을 때, 기존 구조에서 새 디스플레이 환경에 적응하기 위해 수정한 코드량과 본 연구에서 제안한 방법을 충실하게 적용하기 위해 수정한 코드량을 비교하였다. 그리고, 표 6에서는 또 다시 디스플레이 환경이 바뀌었을 때, 기존 구조와 신규 구조에서의 코드 수정량을 비교하였다.

표 5. 디스플레이 환경이 변경될 때, 기존 구조로 작성된 뷰들의 코드 수정량과 신규 구조로 뷰를 작성하기 위한 코드 수정량의 비교

UI 뷰	기존 구조 코드 수정량 (수정 LOC / 전체 LOC)	신규 구조 코드 수정량 (수정 LOC / 전체 LOC)
Home main	26 / 312 (8.3%)	20 / 254 (7.9%)
Quick panel main	48 / 606 (7.9%)	67 / 582 (11.5%)
Settings menu	193 / 2,479 (7.8%)	116 / 1,635 (7.1%)
Out-of-box experience terms & condition	N/A ¹	16 / 375 (4.3%)
Music play	92 / 986 (9.3%)	67 / 454 (14.8%)
Gallery grid	15 / 405 (3.7%)	34 / 313 (10.9%)
Fitness main	N/A ¹	15 / 227 (6.7%)
전체	374 / 4,788 (7.8%)	335 / 3,840 (8.7%)

표 5에서 대표적인 UI 뷰 7가지를 대상으로 코드 수정량을 비교한다. 기존 구조에 대한 코드 수정량이란,

¹ 신규로 개발한 앱이라서 기존 구조에서 측정 불가

표 6. 디스플레이 환경이 또 다시 변경될 때, 기존 구조로 작성된 뷰들의 코드 변경 수준과 이미 신규 구조로 작성된 뷰들에 대한 코드 변경 수준 비교

대상	기존 구조 코드 수정량 (수정 LOC / 전체 LOC)	신규 구조 코드 수정량 (수정 LOC / 전체 LOC)
전체 뷰	374 / 4,788 (7.8%)	12/3,840 (0.3%)

표 6에서 본 연구에서 제안한 방법으로 개발한 뷰를 또 다른 디스플레이 환경에 탑재하기 위해, 추가로 개발한 코드 수정량을 확인할 수 있다. 대부분의 뷰에서는 코드 수정이 전혀 없었지만, Music play 뷰에서 디자이너의 요청에 따라 예외적으로 일부 아이템의 크기를 보정하였다. 이런 예외적인 변경을 고려한다고 하더라도 기존 구조보다 현저하게 적은 0.3%의 코드만 변경된다는 것을 확인하였다.

5. 결론 및 향후 연구

본 연구에서는 다양한 디스플레이 환경에서 적응형으로 동작하는 UI 뷰를 개발하기 위한 방법을 제안하였다. 심미적인 관점에서 높은 수준을 달성하기 위해서는 애초에 디자이너가 의도한 대로 디스플레이에 표현해야 한다. 이를 위해 핵심 콘텐츠 영역을 가운데 고정하고 좌우에는 시스템에 의해 보정되는 여백을 둔다. 디스플레이 환경에 따라 좌우 여백에 변화를 주어 가운데 영역에 위치하고 있는 콘텐츠 영역의 변화를 줄일 수 있다. 하지만, 디스플레이 환경에 따라서 가운데 영역의 크기에 영향을 줄 수밖에 없는 경우도 있다. 특히, 가로 모드에서 세로 모드로 변하는 경우에는 가운데 영역에 미치는 정도가 크다. 이럴 경우 가운데 영역에 위치하고 있는 아이템들의 위치나 크기를 자동으로 보정해주는 방법이 필요하다. 이를 위해 아이템들을 적절히 그룹으로 만들고, 동일 그룹에

속하는 아이템들에 대해 하나의 정렬 기법으로 정의해줄 필요가 있다. 그러면, 디스플레이 환경에 의해 콘텐츠 영역이 변하더라도 아이템들은 유연하게 정렬된다. 더불어 개별 아이템들의 위치나 크기를 추상적인 단위로 정의할 필요가 있다. 디자이너나 앱개발자 입장에서는 픽셀 단위로 아이템의 위치나 크기를 정의할 필요가 없다. 앱단에서 정의한 추상적인 크기는 디스플레이 환경에 따라 시스템에 의해 적합한 물리적인 크기로 치환된다. 물리적인 크기는 디스플레이에 표현할 수 있는 픽셀 단위가 된다.

위에 언급한 방법으로 UIFW 및 UI 뷰들을 개발하였다. 여기서 테스트한 UI 뷰들은 일반적으로 많이 사용하는 그리드 중심의 뷰, 리스트 중심의 뷰, 텍스트 블록 중심의 뷰, 혹은 커스텀으로 제작된 레이아웃을 지닌 뷰이다. 이런 뷰들을 4.5인치, 10.1인치, 23인치, 40인치 등 다양한 크기에서, 16:9, 16:10, 3:2와 같은 다양한 비율에서, 그리고 가로 모드나 세로 모드와 같은 다양한 환경에서 실행해보았다. 그리고, 그 결과 애초에 디자이너가 의도한 바와 같이 표현되는 것을 확인하였다.

여기서 중요한 점은 디스플레이 환경에 변화가 생기더라도 UI 뷰 레벨에서는 최소한의 코드 변경으로 적응할 수 있다는 점이다. 본 연구에서 제안하는 방법을 적용하기 이전에는 디스플레이 환경이 변경될 때마다 전체 앱 코드의 7.8%에 해당하는 코드를 수정해야만 했었다. 하지만, 본 연구에서 제안하는 방법을 적용한 후에는 앱 코드의 변화량이 1% 미만(0.3%)인 점을 확인하였다. 곧, 최소한의 수정으로 다양한 디스플레이 환경에 적용이 가능하고, 그 결과물은 디자이너의 의도에도 부합한다.

본 연구에서는 가운데에 위치하는 콘텐츠 영역을 보전하기 위해 좌우 여백을 사용하였다. 하지만, 이 여백에 대한 최적의 크기에 대해서는 알려진 바가 없다. 심미적인 관점에서 좌우 여백이 좁은 경우도 문제가 될 수 있고, 좌우 여백이 너무 넓은 경우도 문제가 될 수 있다. 따라서, 심미적인 관점에서 어느 정도 수준으로 여백을 유지하는 것이 좋은 지에 대한 추가 연구가 필요하다. 이를 통해 좀 더 유려하게 보여지는 UI 뷰를 개발할 수 있을 것이다.

6. 참 조

[1] K. Pohl, G. Bockle, and F. Van Der Linden, Software product line engineering. Springer, 2005, vol. 10.
 [2] Eisenstein, J., Vanderdonckt, J., and Puerta, A. "Applying Model-Based Techniques to the Development of UIs for Mobile Computers", Proceedings on the International Conference on Intelligent User Interfaces(IUI'01), (Santa Fe, New Mexico, United States, 2001) ACM Press, NY, USA, 69-

76.
 [3] Cheri Mullins. 2015. Responsive, mobile app, mobile first: untangling the UX design web in practical experience. In Proceedings of the 33rd Annual International Conference on the Design of Communication (SIGDOC '15). Association for Computing Machinery, New York, NY, USA, Article 22, 1-6.
 [4] Adaptivity and Layout - Visual Design - iOS - Human Interface Guidelines - Apple Developer. (n.d.). 검색일 11월 8일, 2021년, 출처 Apple Developer: <https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/adaptivity-and-layout/>
 [5] Screen compatibility overview | Android Developers. (n.d.). 검색일 11월 8일, 2021년, 출처 Android developers: https://developer.android.com/guide/practices/screens_support
 [6] Bootstrap · The most popular HTML, CSS, and JS library in the world. (n.d.). 검색일 11월 8일, 2021년, 출처 Bootstrap: <https://getbootstrap.com>
 [7] Grid system · Bootstrap v5.1. (n.d.). 검색일 11월 8일, 2021년, 출처 Bootstrap: <https://getbootstrap.com/docs/5.1/layout/grid/>
 [8] CSS Flexible Box Layout Module Level 1. (2018년). 검색일 11월 8일, 2021년 출처 w3c: <https://www.w3.org/TR/css-flexbox-1/>
 [9] Tizen | An open source, standards-based software platform for multiple device categories. (2021년). 검색일 11월 8일, 2021년, 출처 Tizen: <https://www.tizen.org/>
 [10] Enlightenment Main. (2021년). 검색일 11월 8일, 2021년, 출처 Enlightenment: <https://www.enlightenment.org/>
 [11] Support different pixel densities | Android Developers. (2021년). 검색일 11월 8일, 2021년, 출처 Android Developers Documentation: <https://developer.android.com/training/multiscreen/screendensities>

사전 학습 언어 모델들을 활용한 버그 담당자 자동 할당 성능 향상

왕대성⁰, 성훈, 김민하, 이찬근

중앙대학교 컴퓨터공학과

{dswang,ytrewq615,minal1759,cglee}@cau.ac.kr

Improving Automated Bug Triage by using Pretrained Language Models

Dae-Sung Wang⁰, Hoon Seong, Min-Ha Kim, Chan-Gun Lee

Department of Computer Science and Engineering, Chung-Ang University, Seoul,
Republic of Korea

요 약

여러 도메인과의 결합으로 인하여 소프트웨어의 규모와 복잡성이 계속해서 증가하고 있으며 필연적으로 다양한 버그들이 발생하고 있다. 이는 품질, 자원, 비용 등 여러 측면에서 손실을 일으키며 버그를 효율적으로 관리할 필요성이 계속해서 증가하고 있다. 한편 이를 관리하는 버그 리포트 추적 시스템의 경우 많은 사람이 직접 관리하거나, 자동으로 관리할 때도 단어 기반의 모델로 구성되어 있어 여러 맥락을 고려하지 않고, 클래스의 개수가 많아 정확도가 매우 떨어지는 모습을 보인다. 본 논문에서는 BERT 기반의 언어모델들을 기반으로 파인 튜닝된 사전 학습 언어 모델을 이용하여 Top-10 정확도 기준 약 27%의 정확도 향상을 이루어 냈으며, 결과적으로 약 70%의 정확도를 보이는 것을 실험을 통해 확인하였다. 이를 통해 파인 튜닝된 사전 학습 언어 모델 기반의 버그 담당자 자동 할당 시스템에 대한 가능성을 제시하고, 관리하기 어려움이 있었던 버그 담당자 자동 할당 성능을 향상시킬 수 있는 방법을 논의하였다.

1. 서 론

최근 다양한 소프트웨어를 개발하는 데 있어 단순히 한 도메인에만 치중되지 않고 여러 도메인과의 결합을 통해 다양한 소프트웨어 산출물을 만들어 내고 있다[1]. 이에 따라 소프트웨어의 규모와 복잡성이 계속해서 증가하고 있으며, 다양한 버그들이 필연적으로 발생하고 있다. 충분한 개발 자원이 부족한 실제 산업 현장에서는 버그를 관리하기 위한 시간적, 비용적 손실을 개선하기 위하여 버그를 효율적으로 관리하는 방법의 필요성이 계속해서 제시되었다.

소프트웨어공학 분야에서도 여러 방법을 이용하여 이를 관리하는 방법들을 제안하는 연구들이 활발하게 진행되고 있다. 이렇게 제안된 연구 결과로 실제 산업계에서 개발자들은 버그 추적 시스템(Bug Tracking System)을 사용하여 버그를 관리하고 있으며, 해당 버그 리포트를 사람(담당자)이 직접 확인하여 알맞은 버그 담당자에게 할당한다. 이렇게, 사람이 직접 버그를 분류하고, 담당자를 할당하는 일은 매우 비효율적이며, 알맞게 분류하는 것 또한 제대로 이루어지지 않고 있다.

이러한 문제를 해결하기 위하여, 딥러닝 모델을 적용하여 담당자를 자동으로 배정하기 위한 선행연구들[2,3]이 활발하게 진행되고 있다. 하지만 기존의 연구들은 단어 기반의 순환신경망과 합성곱 신경망에 적용하여 낮은

정확도를 보인다. 따라서 본 논문에서는 문장 기반의 사전 학습 언어 모델을 이용하여 성능을 개선하였고, 특히 버그 담당자를 예측하기 위하여 텐스레이어(Dense Layer)를 목적에 맞추어 파인 튜닝하여 기존 연구 대비 약 27%의 성능 향상을 보였다.

2. 관련 연구

버그 분류(Bug Triage)에서 파생된 버그 담당자 자동 할당은 소프트웨어의 품질 관리 및 유지 보수적 측면에서 필수적이며, 사람이 직접 할당하는 방식에서 벗어나 자동으로 버그를 할당하는 연구가 지속적으로 수행되고 있다[2,3]. Mani등[2]은 단어 기반의 순환신경망 모델을 이용하여 버그 담당자 자동 할당 기법을 제안하였으며 저명한 데이터셋[표 1]을 이용하여 기여도를 기반으로 데이터셋을 분리하여 Top-10 Accuracy는 40% 정도를 보이고 있으나, 단어 기반의 순환신경망 머신러닝 모델의 한계를 극복하지 못했다. 또한 단어 기반의 합성곱 신경망 머신러닝 모델을 활용한 선행 연구로 Lee[3]의 방식은 버그 리포트 정보를 이용하여 버그 담당자 자동 할당을 할 수 있음을 확인하였으나 본 연구의 데이터셋과 달라서 직접적인 비교를 진행하지 않았다.

문장 기반의 트랜스포머 기반 언어 모델을 활용한 선행연구[4]는 단어 기반의 모델의 단점을 개선하기 위하여

표 1. 선행 연구와 실험 결과의 최적 결과값 비교

데이터셋	적용모델	Top-10 정확도	변화정도
Google Chromium-20	DBRNN	42%	27%
Google Chromium-20	DeBERTa	69%	
Mozilla Core -20	DBRNN	38%	16%
Mozilla Core -20	DeBERTa	54%	
Mozilla Firefox-20	DBRNN	46%	10%
Mozilla Firefox-20	DeBERTa	56%	

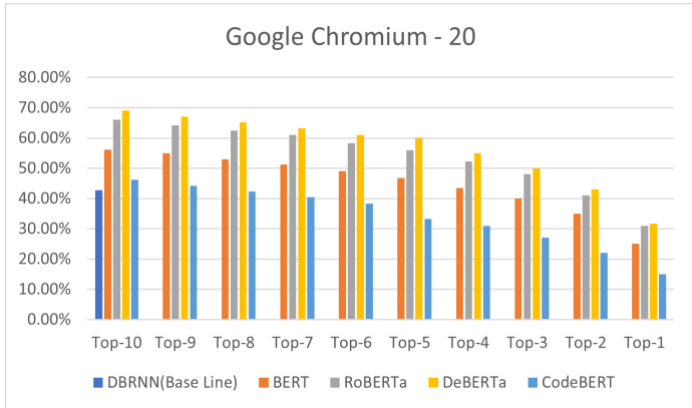


그림 1. Top-K 성능평가 결과: Google Chromium-20

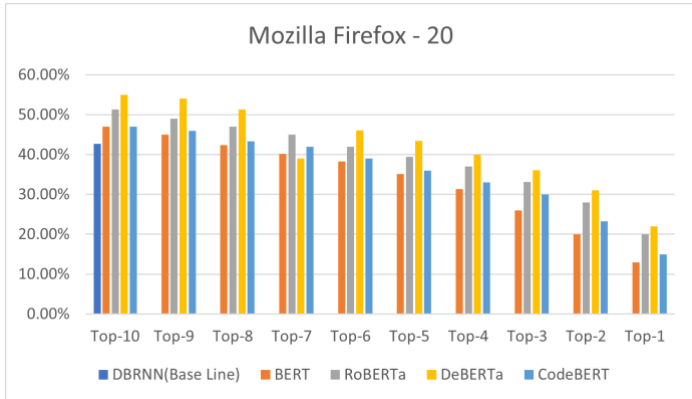


그림 2. Top-K 성능평가 결과: Mozilla Firefox-20

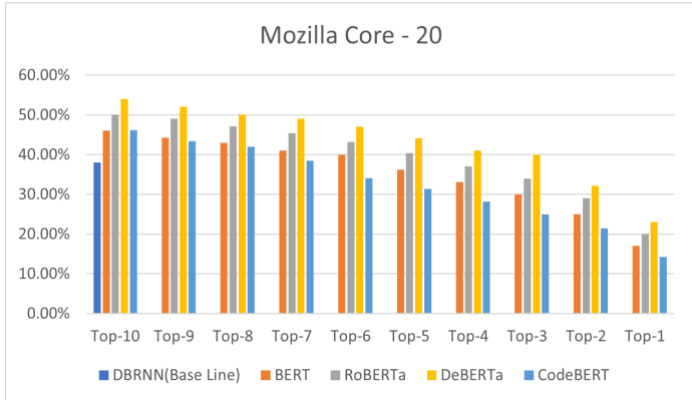


그림 3. Top-K 성능평가 결과: Mozilla Core-20

BERT 기반의 모델을 사용하였으며 실제 산업계의 한글 버그 리포트 데이터를 이용하여 학습을 진행하였다. 하지만 해당 연구에서 사용한 방법은 본 논문에서 예측하는 담당자를

찾아내는 방식이 아닌 개발팀이라는 포괄적인 예측을 하므로 직접적인 성능 비교를 진행하지 않았다.

3. 실험 및 결과

본 논문에서는 선행연구와의 직접적인 성능 비교를 위하여 선행연구의 실험을 재현하여 확인하고 비교하였다. 규격화되지 않은 자연어 데이터셋을 모델에 알맞은 형식으로 선행연구의 기여도 기반으로 분류하였으며 본 논문에서 제시하는 모델에 알맞은 형태로 전처리를 진행하였다.

본 논문에서 사용하는 BERT 기반의 모델들은 최종적으로 히든 레이어에서 분류를 진행하는 구조이며, 1,032개의 클래스를 가지는 본 데이터 셋에서 사용하기에는 부적합하여, 기존 모델을 파인 튜닝하여 본 실험에서 사용하는 데이터셋에서 활용할 수 있도록 모델을 개선하였다. 특히, 본 연구에서 BERT를 기반으로 하는 다중 라벨 분류의 경우 그림 1~3에서 확인 할 수 있듯 본 실험에서 적용하는 파인 튜닝 기법을 이용하여 선행연구 대비 높은 성능을 이루어 낸 것을 통해 본 파인 튜닝 기법이 버그 담당자 배정에 효과적임을 시사한다.

본 연구에서는 기존의 단어 기반의 담당자 자동 할당 모델을 문장 기반의 파인 튜닝된 사전 학습 언어 모델을 이용하여 낮은 정확도를 개선하는 연구를 진행하였으며, 품질 적 제약사항으로 여겨지는 버그 관리 문제를 파인 튜닝된 사전 학습 언어 모델을 이용하여 기존보다 효율적으로 해결할 수 있음을 확인할 수 있었다. 특히, Top-10 Accuracy에서 70%의 정확도를 보이는 결과를 확인할 수 있었으며, 이는 선행연구와 비교하였을 때 약 27%P 이상 향상된 성능을 보인다. 이를 통해 파인 튜닝된 사전 학습 언어 모델이 효과적으로 버그 담당자 자동 배정에 사용될 수 있음을 시사한다.

Acknowledgments

본 연구는 한국연구재단 기초연구사업 (과제 번호 2021R1F1A1059492)의 지원을 받았다.

참고 문헌

[1] Ogheneovo, E. E. “On the relationship between software complexity and maintenance costs”, Journal of Computer and Communications, 2(14), pp.1-9, 2014.
 [2] Mani, S., Sankaran, A., & Aralikatte, R. “Deeptriage: Exploring the effectiveness of deep learning for bug triaging”, in Proc. of India Joint International Conference on Data Science and Management of Data, pp. 171-179, 2019.
 [3] Lee, S. R. et al. “Applying deep learning based automatic bug triager to industrial projects”, in Proc. of Joint Meeting on Foundations of Software Engineering (FSE), pp. 926-931, 2017.
 [4] 최예람, 백종문. “BERT 기반 자연어 처리 딥러닝 모델을 활용한 효과적인 버그 배정”, KCSE 2021, 제23권, 제1호, pp. 24-31, 2021.

Actor-Critic 계열 강화학습 알고리즘의 행동 정확성 평가 방법

김 영 재⁰, 홍 장 의

충북대학교 컴퓨터과학과

youngjae@cbnu.ac.kr, jehong@cbnu.ac.kr

Evaluation of Action Correctness for Actor-Critic Series Reinforcement Learning Algorithm

Youngjae Kim, Jang-Eui Hong

Dept. of Computer Science, Chungbuk National University, South Korea

요 약

최근의 다층신경망을 이용한 인공지능은 매우 높은 성능을 선보이며 다양한 분야에서 활용되고 있다. 하지만 이러한 종류의 인공지능은 논리적 설계에 의해 개발된 것이 아니라 학습 데이터를 통하여 신경망이 스스로 학습한 것이기 때문에 인간의 입장에서는 블랙박스와의 같은 특성을 보인다. 다층신경망의 이러한 특성은 ISO/IEC 25010과 같이 소프트웨어의 품질을 보증하기 위한 기존의 방법들을 적용하기 어렵게 만든다. 따라서 인공지능을 활용한 소프트웨어의 품질 보증을 위하여 인공지능의 학습 전반에 걸친 여러 특성들을 평가하고 그 품질을 보장하는 방법에 대한 연구가 시작되고 있다. 한편, 강화학습은 에이전트가 주어진 상태에서 어떤 행동을 하였을 때, 환경으로부터 받은 보상을 통하여 학습을 진행하는 인공지능의 중요한 연구 분야 중 하나이다. 이러한 강화학습의 평가는 최대의 보상을 얻는 방법을 학습한다는 강화학습의 목표에 따라 주로 보상 값의 변화 위주로만 수행이 되었고, 따라서 강화학습이 갖는 다양한 품질 속성에 대한 평가가 이루어지지 않았다. 본 논문에서는 미리 설계된 이상적 행동 값과 학습된 강화학습의 정책망으로부터 생성된 행동 값의 차이를 비교, 분석하는 방법을 통하여 강화학습에서 가장 인기 있는 Actor-Critic 계열 강화학습 알고리즘의 품질 속성 중 하나인 행동 정확성의 평가 방법을 제시한다. 제시된 방법으로 동일한 환경에서 학습된 두 가지 종류의 강화학습 알고리즘을 평가하여 비교하였고, 이를 통해 기존의 보상 값 변화 위주의 평가 방법으로는 알 수 없었던 강화학습의 품질 속성을 확인하였다.

1. 서론

인공지능 기술의 급격한 발전으로 세상이 더욱 더 편리해지고 있다. 한 예로 인공지능 스피커는 인공지능 기반의 뛰어난 음성인식 성능을 바탕으로 사람의 명령을 알아듣고 수행한다. 딥러닝(Deep Learning)으로도 불리는 다층신경망과 인공지능의 결합은 이러한 최근의 놀라운 발전을 이끌어냈다. 이러한 발전은 인공지능이 인간 최고수를 이기는 것이 불가능하리라 여겨졌었던 바둑에서 구글 딥마인드의 바둑 인공지능인 알파고(AlphaGo)[1]가 이세돌 9단을 꺾는 성공을 거둠으로써 대중에게도 널리 알려지게 되었다.

하지만 딥러닝 이전의 고전적 인공지능이 인간에 의해 연역적, 논리적으로 설계되고 개발된 것인 반면, 딥러닝을 이용한 인공지능은 막대한 양의 학습 데이터를 통하여 신경망이 스스로 학습한 결과물이기 때문에 그 개발 과정이 귀납적이며, 인공지

능이 왜 그런 판단을 하고 그러한 출력 값을 도출하였는지 알기가 어렵기 때문에 인간의 입장에서는 블랙박스와의 같은 특성을 보인다[2]. 딥러닝의 의사결정 과정을 이해하고 추적하기 위해 설명 가능한 인공지능(Explainable AI)에 대한 연구[3-5]가 수행되고 있지만, 아직은 많은 연구를 필요로 하는 상황이다.

딥러닝을 이용한 인공지능은 이를 활용한 소프트웨어의 품질 보증에도 또 다른 어려움을 발생시킨다. 이러한 인공지능에는 ISO/IEC 25010 표준[6]과 같이 기존의 소프트웨어의 품질 보증을 위해서 개발된 표준들을 그대로 적용하기 어려운 새로운 품질 속성들이 다수 존재하기 때문[7]이다. 따라서 이러한 인공지능을 위한 새로운 품질의 평가와 보증 방법이 요구되고 있다.

한편 인공지능의 핵심 연구분야인 강화학습(Reinforcement Learning)은 에이전트(Agent)가 주어진 상태(State)에서 어떠한 행동(Action)을 하였을 때, 환경으로부터 받게 되는 보상(Reward)을 통하여 학습을 하는 것이다. 강화학습은 게임, 로

봇제어, 투자 거래 등과 같은 영역에서 점점 그 활용도를 크게 넓혀가고 있다. 전통적으로 강화학습 알고리즘의 성능을 평가하기 위해서 환경으로부터 획득한 보상 값의 변화를 관찰하는 방법이 주로 사용되어 왔다. 그러나 이와 같은 방법은 알고리즘의 학습 환경의 해결 능력과 학습 속도를 평가할 수는 있지만 딥러닝을 이용한 강화학습이 갖는 다른 품질 속성을 평가하기에는 부적합한 부분이 존재한다. 특히 학습된 정책은 딥러닝 특유의 확률적 행동 갖기 때문에 학습이 정확한 행동을 하도록 이루어졌는지를 평가하는 것은 중요한 일이다[7].

따라서 본 논문에서는 강화학습 알고리즘의 행동 정확성(Action Correctness)을 평가할 수 있는 방법에 대하여 연구하였다. 이상적인 행동(Ideal Action)을 제공할 수 있는 환경을 통해서 도출한 이상적인 행동 값과 이러한 환경에서 학습된 강화학습의 정책망(Policy Network)이 생성하는 행동 값을 서로 비교함으로써 얼마나 정책망이 이상적인 행동에 가깝게 학습이 되었는지를 평가하였다. 제시된 평가 방법은 강화학습에서 가장 인기 있는 Actor-Critic 계열의 강화학습에 적용이 가능하다. 또한 제시된 방법으로 두 가지 종류의 알고리즘을 평가하여 비교한 결과를 분석하였고, 이를 통해 기존의 보상 값 위주의 평가 방법이 확인하지 못했던 새로운 품질 속성이 존재함을 확인하였다.

본 논문의 기여는 다음과 같다.

- Actor-Critic 계열의 강화학습 알고리즘의 학습된 정책망을 통해 행동 정확성을 평가하기 위한 방법을 제시하였다.
- 제시된 방법을 적용하여 두 가지의 강화학습 알고리즘의 행동 정확성을 평가하였고, 그 결과로부터 기존의 평가 방법이 확인하지 못했던 정책망의 변화를 확인하였다.

2. 선행 연구

2.1 인공지능의 소프트웨어 품질

딥러닝을 이용한 인공지능의 연구가 활발해지고, 그 사용이 점차 확대됨에 따라 신뢰할 수 있는 인공지능 개발의 중요성이 커졌고, 이를 위해 딥러닝을 활용한 소프트웨어의 품질 보증 방법이 새로운 문제로 떠올랐다. 딥러닝이 갖는 고유의 특성은 기존의 소프트웨어 품질 보증을 위한 방법의 적용을 어렵게 만들기 때문에, 새로운 품질 기준과 보증 방법을 필요로 하게 되었다.

이러한 목적을 위해 일본에서는 QA4AI(Quality Assurance for Artificial-Intelligence-based products and services) 컨소시엄이 구성되었다. 이 컨소시엄에는 기존 소프트웨어의 품질 보증 엔지니어와 테스트 엔지니어가 참여하였으며, 딥러닝의 품질 평가를 위한 가이드라인[7]을 발표하였다. 이 가이드라인에서는 딥러닝의 품질에 대하여 <표 1>와 같이 다섯 가지의 관점

을 정의하였다. 각 관점에는 확인해야 할 체크포인트들이 존재한다. 또한 딥러닝을 활용한 소프트웨어의 품질 향상을 위하여 각 관점의 품질 균형이 잘 잡힐 것을 권장하였다.

또한 J. Siebert 등의 연구[8]에서도 딥러닝의 품질 보증과 관련된 연구가 이루어졌다. 이 연구에서는 딥러닝 시스템의 품질 관점을 모델(Model), 데이터(Data), 시스템(System), 기반(Infrastructure), 환경(Environment)의 다섯 가지로 분류하고, 각각의 분류에 따르는 품질 속성을 정의하였다. 특히 모델 관점에서 고려해야 할 품질 속성과 그에 대한 정의는 <표 2>와 같다. 하지만 이 연구는 학습 데이터와 검증 데이터를 구분하고 있고, 정답을 의미하는 데이터의 라벨링을 언급하는 등 주로 지도학습(Supervised Learning)을 고려하여 수행되었기 때문에 강화학습의 품질 평가에 적용하기 위해서는 추가적인 연구 필요로 한다.

2.2 강화학습

강화학습은 에이전트가 해결하려는 환경과의 상호작용에 의하여 이루어진다. 이때 어떤 행동의 결과로 인해 환경으로부터 양수의 보상이 주어진 경우 해당 행동은 좋은 것으로 평가되어 강화되고, 음수의 보상이 주어진 경우에는 해당 행동이 나쁜 행동으로 평가되어 약화된다. 주로 표형식 표현(Tabular Representation)에 의존하던 고전 강화학습은 대규모의 관찰공간(Observation Space)을 처리하는데 큰 어려움을 겪었으나 딥러닝 기술을 도입함으로써 이를 극복[9]하였다. 또한 오랜 기간 강화학습은 이산행동공간(Discrete Action Space)의 학습만 가능했지만 로봇제어와 같이 연속행동공간(Continuous Action Space)을 갖는 환경에도 성공적으로 응용[10]되어 더욱 더 그 활용 영역을 넓혀가고 있다.

다양한 강화학습의 알고리즘 중 현재 가장 널리 쓰이는 방식은 Actor-Critic 계열의 알고리즘[11]이다. Actor-Critic 방식은 Actor에 해당하는 주어진 상태에서 어떠한 행동을 할 것인지를 결정하는 정책망과 Critic에 해당하는 주어진 상태에서 한 행동이 얼마나 좋은 행동인지 그 가치를 평가하는 가치망(Value Network) 두 가지의 네트워크를 동시에 학습시키는 것이다. 현재 연속행동공간을 갖는 환경에서의 강화학습을 위해 널리 쓰이는 SAC(Soft Actor-Critic)[12], TD3(Twin Delayed DDPG)[13] 등의 알고리즘은 물론이고, 현재 가장 높은 성능을 나타내고 있는 TQC(Truncated Quantile Critics)[14] 알고리즘 모두 Actor-Critic 방식을 사용하고 있다.

한편, 이러한 강화학습을 학습시키기 위한 환경으로는 Gym 라이브러리[15]가 인기리에 쓰인다. 강화학습의 연구와 교육을 위해 설립된 Open AI 재단에서 개발한 Gym 라이브러리는 강화학습 에이전트를 학습시킬 수 있는 카트폴(CartPole), 마운틴

〈 표 1 〉 Five aspects of quality evaluation for ML-based systems[7]

Aspects	Description
Data Integrity	The quality assurance of samples of inputs and outputs
Model Robustness	The quality assurance of a model generated automatically
System Quality	The quality assurance of the whole system
Process Agility	The quality assurance from the viewpoint of development process
Customer Expectation	The quality assurance for various stakeholders, who may be unfamiliar with ML-based system development

〈 표 2 〉 Overview of the derived quality model for model view[8]

Measurement Object	Quality Attribute	Example quality metrics and checklists
Model type	Appropriateness: Degree to which the model type is appropriate for the current task (e.g., classification, etc.) and can deal with the current data type (e.g., numerical, categorical)	Prerequisites for model type
Trained model	Development correctness (Goodness of Fit): Ability of the model to perform the current task measured on the development dataset	Precision, Recall, F-score, etc. for training
	Runtime correctness (Goodness of Fit): Same as above measured on the runtime dataset	Precision, Recall, F-score, etc. at runtime
	Relevance (Bias-Variance tradeoff): Degree to which the model achieves a good bias-variance trade-off (neither underfitting nor overfitting the data)	Variance of cross-validation goodness of fit
	Robustness: Ability of the model to handle noise or data with missing values and still make correct predictions	Equalized Loss of Accuracy (ELA)
	Stability: Degree to which a trained model generates repeatable results when it is trained on different subsets of the training dataset	Leave-one-out cross-validation stability.
	Fairness: Ability of the model to output fair decisions	Equalized odds
	Interpretability: Degree to which the trained model can be interpreted by humans	Complexity metrics (e.g., no. of parameters, depth)
	Resource utilization: Resources used by the model when it is already trained	Required storage space

카(MountainCar) 등과 같은 고전적인 강화학습 환경은 물론, 벽돌깨기(Breakout), 팩맨(Pacman) 등과 같은 다양한 아타리(Atari) 게임 환경을 사용할 수 있다. 또한 MuJoCo 물리엔진 [16]을 통해 복잡한 물리학이 적용되는 로봇 제어(Robotics), 역진자(InvertedPendulum) 등과 같은 환경도 지원하고 있다. Gym 라이브러리와 같은 공통 강화학습 환경의 사용은 서로 다른 강화학습 연구 결과의 성능을 비교하기에 용이하게 만든다.

2.3 관련 연구

딥러닝을 활용한 인공지능 시스템의 품질 향상을 위한 연구들이 지속되고 있다. D. Schneegass 등은 강화학습의 품질 보증을 위해 불확실성이 학습 데이터를 통해 전파되었을 때 변화되는 정책망의 신뢰성을 평가[17]하였다. S. Nakajima 등은 기계학습 기반 시스템의 서비스와 품질에 대한 연구[18]를 수행하

였다. S. Chan 등도 강화학습의 신뢰성을 측정하는 방법에 대한 연구[23]를 수행하였다.

또한 S. Studer 등은 기계 학습을 사용한 소프트웨어의 생애 주기 전반을 고려한 프로세스인 CRISP-ML(Q)를 제안[19]하였다. A. Poth 등은 기존 산업에 기계학습 시스템을 도입하였을 때, 산업에서 요구하는 기계학습의 품질에 대해 연구[20]하였다.

한편 S. Jordan 등은 강화학습의 재현성이 부족한 문제[21]를 해결하기 위한 새로운 성능 평가 방법에 대한 연구[22]를 수행하였다. 이 연구에서는 여러 환경에서 나타난 알고리즘의 성능을 종합적으로 분석함으로써 알고리즘의 성능을 입체적으로 평가하였다.

그러나 강화학습을 비롯해 딥러닝을 활용한 인공지능 시스템의 품질 보증에 대한 연구는 아직 초기 수준으로, 더욱 많은 연구가 필요한 상황이다.

3. 행동 정확성 평가 방법

본 논문에서는 Actor-Critic 계열 강화학습의 세부적인 품질 속성 중 하나로서, 학습된 정책망이 주어진 환경에서 보상을 최대화하는 방향의 행동을 얼마나 정확하게 출력해 내는지에 대한 정도를 행동 정확성으로 정의하고, 이러한 행동 정확성을 평가하는 방법에 대하여 연구하였다. 이러한 행동 정확성은 <표 1>의 분류에서는 모델 강건성(Model Robustness)에 속하고, <표 2>의 분류에서는 모델 품질 관점의 정확성(Correctness)과 연관된 강화학습의 품질 속성 중 하나이다. Actor-Critic 계열의 강화학습은 학습 중에는 정책망과 가치망을 모두 사용하지만 일단 강화학습을 사용한 소프트웨어가 배포된 이후에는 정책망만을 사용하기 때문에 정책망이 높은 행동 정확성을 갖추도록 한다면 배포된 소프트웨어의 품질을 향상시킬 수 있다.

3.1 기존 평가방법의 문제점 및 연구의 필요성

강화학습의 성능을 평가하는데 일반적으로 광범위하게 사용되는 평가지표는 보상 값의 변화이다. 학습 초기에는 에이전트가 환경으로부터 낮은 보상 값을 획득하지만, 점차 학습이 진행됨에 따라 환경으로부터 받는 보상 값이 증가하는 것을 보임으로써 에이전트가 환경을 해결하고 학습이 잘 이루어지고 있음을 나타낼 수 있다. 그밖에 지표로는 각 에피소드에서의 성공 확률, 가치망의 손실 값(Loss Q-value) 변화 등이 드물게 쓰인다. 또한 에피소드가 성공하거나 실패하였을 때만 보상을 받는 희소보상환경(Sparse Reward Environment)에서는 에피소드 당 진행된 스텝의 수 등을 이용하여 학습 성과를 표현하기도 한다.

최근의 강화학습의 알고리즘을 연구한 논문들[13, 14]은 대체로 논문에서 자신들이 연구한 알고리즘이 기존의 알고리즘보다 동일한 환경에서 더 빠르게 보상 값이 증가하여 환경을 해결할 수 있음을 보여준다. 그러나 이러한 보상 값에 의한 기존의 평가 방법들은 일반적으로 강화학습의 에이전트가 환경을 잘 해결하였는지에 대한 여부와, 학습이 잘 이루어지고 있는지에 대한 상황을 보여줄 수는 있지만, 관찰 공간 전체에 대하여 얼마나 올바른 행동이 충분히 잘 학습되었는지, 이를 바탕으로 정확한 행동을 통해 환경을 해결하였는지는 알기 어렵다. 특히 위험 행동이 인명의 손실과 재산상의 피해를 가져올 수 있는 안전필수(Safety-Critical) 시스템에 강화학습을 적용하려는 경우, 학습된 에이전트가 위험한 행동을 발생시키지는 않는지에 대한 평가가 면밀히 이루어져야 한다. 따라서 강화학습의 행동 정확성에 대한 평가는 반드시 연구되어야 할 문제이다.

또한 에이전트가 환경을 잘 해결한 이후에는 높은 보상을 받을 수 있는 정답에 해당하는 상태로만 주로 전이하기 때문에 과적합(Overfitting) 문제[24]를 발생시킨다. 따라서 학습된 에이

전트를 다시 환경과 상호작용시켜 보상 값을 측정하는 방식의 기존 평가 방법은 전체 관찰 공간이 적절하게 학습되었는지 평가하는데 부적합하다. 과적합은 딥러닝 연구의 초기에는 주로 지도학습에서 발생하는 문제로, 강화학습과는 크게 관련이 없는 것으로 여겨졌지만, 다양한 연구에 의하여 현재는 강화학습에도 적용되는 문제로 확인되었다. 과적합이 일어난 강화학습의 에이전트는 최대의 보상을 얻을 수 있는 정답을 찾은 이후, 정답에 해당하는 상태로의 행동만을 지속적으로 출력하며 주어진 환경을 암기(Memorization)해버리고, 정답과 관련이 없는 상태에서의 적절한 행동은 이미 기존에 잘 학습이 되어있었다 하더라도 망각되는 현상이 발생한다. 이러한 상황에서도 보상은 일정하게 높은 값을 유지하고 있으므로 기존의 평가 방법은 명확한 한계를 갖는다. 따라서 강화학습의 품질을 향상시키기 위해 기존 평가 방법의 문제점을 극복하고 강화학습의 다양한 품질 속성을 평가할 수 있는 새로운 방법이 필요하다.

3.2 이상적 행동을 제공하는 환경

인공지능에 딥러닝을 사용하는 이유는 인간의 논리적 설계를 통해 만들어내기 복잡한 판단 과정을 기계학습에게 대신 맡기기 위함이다. 학습 데이터를 통하여 학습된 딥러닝은 스스로 문제를 해결할 수 있는 능력을 갖추게 되고, 일부 영역에서는 인간을 초월하는 성능을 달성하였다. 한 예로 알파고는 그동안 인간들이 미처 생각하지 못했던 바둑의 묘수를 여럿 발견하였고, 이러한 수들은 다시 인간 바둑 기사들에 의해 좋은 수로 평가받게 되었다. 이렇듯 일반적으로 강화학습이 해결하기를 바라는 환경은 인간이 정답을 모르고 있고, 따라서 인공지능이 그 정답을 찾아주기를 바라는 것이 대부분이다.

그러나 본 연구에서는 강화학습이 찾아낸 답이 정답과 얼마나 가까운지를 통하여 강화학습 알고리즘의 행동 정확성을 평가하고자 하므로 주어진 상태에서 제공된 보상함수에 따라 가장 큰 보상을 얻을 수 있는 이상적인 행동이 무엇인지를 알고 있어야 한다. 본 논문에서는 행동 정확성을 평가하기 위하여 필요로 하는 이러한 환경을 이상적 행동을 제공하는 환경(Environment for Providing Ideal Action, PIA 환경)이라 정의한다. 본 논문의 목적을 달성하기 위한 PIA 환경은 이상적 행동을 미리 고려하여 설계를 하거나, 기존의 강화학습 환경을 엄밀하게 분석함으로써 얻을 수 있다. 한편, 도출된 이상적인 행동은 오로지 학습이 종료된 후, 행동 정확성을 평가할 때만 사용되고, 학습을 진행하는 강화학습의 에이전트는 주어진 상태에서의 이상적 행동 값을 전혀 모른 채 기존의 방법대로 학습을 진행한다. 따라서 어떤 강화학습 알고리즘이 다양한 종류의 PIA 환경들을 잘 해결하고 그로부터 공통적인 특성을 갖는다면, 이상적인 행동이 무엇인지 알 수 없는 환경에서도 동일한 특성을 나타

내리라고 기대할 수 있다. 이러한 특성은 높은 행동 정확성을 요구하는 강화학습 시스템의 개발에 큰 도움이 될 것이다.

이러한 목적을 달성하기 위해 PIA 환경은 주어진 상태에서 최대의 미래보상총합(Total Future Reward)을 얻을 수 있는 이상적 행동을 구하는 함수 f 를 제공한다. 에이전트가 환경을 관찰하여 얻은 상태들의 집합을 S 라 하고, 에이전트의 행동들의 집합을 A 라 하자. 함수 $f: S \rightarrow A$ 는 주어진 상태 $s \in S$ 에서의 이상적인 행동 $a \in A$ 를 구하는 함수로서, $a = f(s)$ 를 만족한다.

3.3 강화학습 정책망의 행동 정확도 평가 방법

본 연구에서는 연속행동공간을 갖는 환경에서 Actor-Critic 계열 강화학습 알고리즘의 행동 정확성 평가 방법을 제안한다.

강화학습을 통해 학습된 에이전트의 상태집합을 S , 행동집합을 A 라 할 때, 파라미터 θ 에 대한 강화학습의 정책망 π_θ 의 입력은 $s \in S$ 이고, 출력은 행동 $a \in A$ 이다. 즉, $a = \pi_\theta(s)$ 를 만족한다. 이때의 a 는 정책망 π_θ 의 출력층(Output Layer)을 구성하는 n 개의 뉴런이 출력하는 실수 값 $a_{(1)}, a_{(2)}, \dots, a_{(n)}$ 을 원소로 하는 튜플로서,

$$a = (a_{(1)}, a_{(2)}, \dots, a_{(n)})$$

로 나타낼 수 있고, $n = \dim(A)$ 가 된다. 한편 이 튜플 a 로부터 a 의 각 원소를 성분으로 하는 n 차원 벡터공간의 벡터

$$\vec{a} = (a_{(1)}, a_{(2)}, \dots, a_{(n)})$$

을 생성할 수 있다. 서로 유사한 행동 값들을 원소로 갖는 두 행동 $a_p (\in A), a_q (\in A)$ 으로부터 생성한 두 벡터 \vec{a}_p, \vec{a}_q 는 유사한 길이와 방향을 갖게 된다.

두 벡터의 유사한 정도를 측정하는 다양한 방법이 존재한다. 이 논문에서는 그 중 대표적 방법인 코사인 유사도와 L2-노름을 사용한다.

■ 코사인 유사도 (Cosine Similarity) : 코사인 유사도는 두 벡터가 이루는 각의 크기가 작을수록 같은 방향을 향하는 유사한 벡터임을 이용하여 나타낸 값이다. 코사인 유사도는 특히 언어공학 분야의 인공지능에서 단어의 유사도를 측정하기 위한 방법으로 널리 쓰이고 있다[25]. n 차원 벡터공간에서의 두 벡터 \vec{p}, \vec{q} 에 대한 코사인 유사도 c 는 다음의 식 (Equ. 1)을 이용하여 구할 수 있다.

$$c = \frac{\vec{p} \cdot \vec{q}}{\|\vec{p}\| \times \|\vec{q}\|} \quad (\text{Equ. 1})$$

이때, $c \in [-1, +1]$ 이고, 두 벡터가 이루는 각의 크기가 작을수록, 즉 두 벡터가 유사할수록 c 의 값이 +1에 가까워진다.

그러나 코사인 유사도는 두 벡터의 방향만을 이용하여 계산되기 때문에 벡터의 길이는 고려되지 않는다는 단점을 갖는다.

■ L2-노름 (L2-Norm) : 유클리드 거리(Euclidean distance)로도 불리는 L2-노름은 두 벡터의 길이가 비슷할수록, 또 두 벡터가 이루는 각의 크기가 작을수록 두 벡터의 중점 사이의 거리가 짧아진다는 점을 이용하여 유사도를 측정하는 방법이다. n 차원 벡터공간의 두 벡터 $\vec{p} = (p_{(1)}, p_{(2)}, \dots, p_{(n)})$, $\vec{q} = (q_{(1)}, q_{(2)}, \dots, q_{(n)})$ 에 대한 L2-노름 d 는 다음의 식 (Equ. 2)를 이용하여 구할 수 있다.

$$d = \sqrt{\sum_{k=1}^n (p_{(k)} - q_{(k)})^2} \quad (\text{Equ. 2})$$

이때 $d \in [0, \infty)$ 이고, 두 벡터가 유사할수록 d 의 값이 0에 가까워진다. 그러나 L2-노름은 길이 차이가 많이 나는 벡터들의 유사도를 정확하게 측정하기에 불리하다는 단점이 있다.

위와 같은 내용을 바탕으로 본 연구에서 제시하는 강화학습의 행동 정확성 평가 절차는 다음과 같다.

1) 연속균등분포(Continuous Uniform Distribution)를 이용하여 상태집합으로부터 m 개의 상태

$$s_1, s_2, \dots, s_m \quad (s_k \in S, k = 1, 2, \dots, m)$$

을 수집한다.

2) 정책망 π_θ 를 이용하여 수집한 m 개의 상태 s_1, s_2, \dots, s_m 에 대한 행동

$$a_1 = \pi_\theta(s_1), a_2 = \pi_\theta(s_2), \dots, a_m = \pi_\theta(s_m)$$

을 각각 구한다.

3) 이상적인 행동을 구하는 함수 f 를 이용하여 수집한 m 개의 상태 s_1, s_2, \dots, s_m 에 대한 이상적인 행동

$$i_{a_1} = f(s_1), i_{a_2} = f(s_2), \dots, i_{a_m} = f(s_m)$$

를 각각 구한다.

4) $k = 1, 2, \dots, m$ 에 대하여 a_k 와 i_{a_k} 를 각각 벡터 \vec{a}_k 와 \vec{i}_{a_k} 로 변환한 후, 코사인 유사도와 L2-노름을 각각 계산한다.

5) 위와 같은 과정을 통해 얻은 m 개의 코사인 유사도와 L2-노름을 각각 평균 내어 강화학습의 행동 정확성을 평가한다.

학습된 정책망을 직접 조사하여 얻은 이러한 행동 정확성의 평가 방법은 기존의 보상 값을 통한 평가 방법에 비해 다음과 같은 장점을 갖는다.

■ 기존의 평가 방법은 테스트 환경을 구축하고, 테스트 환경에서의 상태에 따라 정책망으로부터 얻은 행동을 이용해 다시 환경과 상호작용함으로써 얻은 보상을 이용한다. 따라서 정책망 자체를 직접적으로 평가하는 본 논문의 행동 정확성 평가

방법은 보다 근본적인 학습 결과에 대한 평가가 될 수 있다.

- 테스트 환경에서의 상호작용을 필요로 하지 않기 때문에 테스트 환경 구축에 필요한 컴퓨팅 자원의 사용을 줄일 수 있다.
- 테스트 환경에서는 학습된 결과에 따라 일부 상태들에 대해서만 전이가 발생할 수 있으므로 전이가 발생하지 않은 상태들에 대한 평가가 불충분해진다. 그러나 본 논문의 행동 정확성 평가 방법은 전체 상태 집합을 골고루 조사하기 때문에 이러한 문제점이 발생하지 않는다.

4. 평가 설계 및 결과 분석

4.1 평가 방법 설계

현재 가장 널리 쓰이고 있는 두 가지 종류의 강화학습 알고리즘인 TD3, SAC 알고리즘의 행동 정확성을 단일 환경에서 평가하였다. 먼저 행동 정확성 평가를 위한 PIA 환경을 설계하였다. 설계한 환경에 대한 구체적인 내용은 다음과 같다.

〈그림 1〉의 (a)와 같이 2차원 공간에 각각 반지름이 0.1, 0.4, 0.7, 1.0, 1.3, 1.6인 6개의 동심원이 있다. 관찰공간은 가장 큰 원의 내부 전체이며, 이 관찰공간의 점으로 표현되는 에이전트가 위치한다. 이 에이전트는 자신의 위치를 나타내는 x 좌표와 y 좌표로 구성된 2차원의 상태 $s = (x, y)$ 를 가진다. 또한 행동공간도 2차원으로, 이 공간에서의 행동은 각각 x 축 방향과 y 축 방향으로의 움직임을 결정하는 $[-1, +1]$ 범위를 갖는 두 개의 실수 값 $a_{(1)}, a_{(2)}$ 로 구성된 $a = (a_{(1)}, a_{(2)})$ 로 표현된다. 임의의 점에서 에피소드를 시작한 에이전트는 출력된 행동 값에 따라 원 내부를 이동하며, 이동의 결과로 6개의 원 중 한 원에 접촉하였을 때 에피소드가 종료된다.

〈그림 1〉의 (b)는 이 환경의 보상함수를 그림으로 나타낸 것이다. 에이전트가 빨간색으로 표현된 원에 접촉한 경우 양수인 +100의 보상 값을 받고, 파란색으로 표현된 원에 접촉한 경우 음수인 -100의 보상 값을 받는다. 따라서 에이전트는 현재 위치에서 빨간색 원을 향해 이동하는 방법을 학습하게 된다. 또한 원에 접촉하지 않은 경우에는 -1의 음수 보상 값을 받도록 보상함수를 설계하였다. 따라서 이 환경에서 에이전트는 최대한 빠르게 빨간색 원에 도달하는 방법을 학습하게 된다.

이러한 환경에서 〈그림 1〉(c)와 〈그림 1〉(d)는 각각 행동 a 의 첫 번째 원소인 $a_{(1)}$ 와 두 번째 원소인 $a_{(2)}$ 의 각 상태에서의 이상적인 행동을 나타낸 것이다. 각 그림에서 가로축은 상태의 x 값을, 세로축은 상태의 y 값을 의미하며, 각 상태에서 이상적인 행동 값을 색상으로 변환하여 나타내었다. 이때 빨간색은 +0.1의 행동 값을, 파란색은 -0.1의 행동 값을 뜻한다. 따라서 이 그림은 학습을 진행하는 강화학습 에이전트의 정책망이

목표로 하는 출력 값을 나타내는 것이다.

이러한 환경에서 OpenAI 재단에서 개발한 강화학습 프레임워크인 SpinningUp에 구현된 SAC, TD3 알고리즘을 사용하여 학습시켰다. 또한 신경망의 활성화함수(Activation Function)에 따른 행동 정확성의 차이를 알아보기 위하여 각 알고리즘에서 ReLU 함수[26]를 활성화함수로 갖는 경우와 Tanh 함수를 활성화함수로 갖는 경우로 나누어 학습을 진행하였다.

모든 학습은 10회씩 수행되었으며, 각 학습은 1 에포크(Epoch)에 4000회의 스텝(Step)을 갖도록 하여 100 에포크씩 학습하였다. 또한 학습을 시작하기 전 초기 상태의 신경망을 저장하였고, 각 에포크가 종료될 때마다의 신경망도 저장하였다. 학습이 종료된 후, 저장된 신경망을 통하여 각 에포크에서의 행동 정확성을 평가하였고, 그 변화를 관찰하였다.

4.2 행동 정확성 평가 결과

〈그림 2〉의 (a), (b)는 학습이 진행됨에 따른 신경망의 행동 정확성의 변화를 나타낸 그래프이다. 학습이 진행됨에 따라 〈그림 2〉(a)의 평균 코사인 유사도는 점차 증가하고 있고, 〈그림 2〉(b) 평균 L2-노름은 점차 감소하고 있으므로 정책망이 점점 이상적인 행동에 가까운 값을 출력하도록 학습되고 있음을 확인하였다.

또한 이 그래프로부터 행동 정확성은

ReLU-SAC > ReLU-TD3 > Tanh-TD3 > Tanh-SAC
순으로 높게 나타난다. 이는 행동 정확성이 알고리즘의 차이보다는 활성화함수의 차이에 더 크게 영향을 받는 것을 알 수 있다. 이러한 결과는 높은 행동 정확성을 요구하는 시스템의 개발에 중요한 시사점을 제공해준다.

〈그림 3〉은 학습된 정책망이 출력하는 행동 값과 행동 정확성이 이상적인 값에 가까워지는지를 시각적으로 확인해 본 것이다. 각 실험 조건에 대하여 수행된 10회의 실험 중, 첫 번째 실험으로부터 얻은 결과를 그림으로 나타내었다. 0 에포크는 학습 전의 초기 신경망 상태를 의미하며, 그림은 각 상태에 따른 값을 색상으로 변환하여 나낸 것이다.

〈그림 3〉의 (a), (b)로부터 학습이 진행됨에 따라 정책망이 출력하는 행동 값이 이상적인 행동 값과 유사해지는 것을 시각적으로 확인할 수 있다. 또한 이 그림으로부터 활성화함수로 ReLU 함수를 사용한 실험이 이상적인 행동 값을 더 잘 모사해내는 것을 확인할 수 있다. 이는 〈그림 2〉의 (a), (b)에서 나타난 결과와 서로 일치한다. 또한 모든 알고리즘에서 이상적 행동 값의 부호가 바뀌는 경계 부근에서는 정책망의 출력 값이 이상적인 행동 값과 차이가 나는 것을 확인할 수 있다.

〈그림 3〉의 (c), (d)에서는 각각 평균 코사인 유사도와 평균 L2-노름을 동일한 방법으로 시각화하였다. 평균 코사인 유사도

는 +1에 가까운 값을 가질수록 이상적인 값과 가까워지므로 그림의 상태공간 전체가 전부 빨간색일 때 가장 행동 정확성이 높은 것이고, 평균 L2-노름은 0에 가까운 값을 가질수록 이상적인 값과 가까워지므로 그림의 상태공간 전체가 전부 파란색일 때 가장 행동 정확성이 높은 것이다. 행동 정확성 또한 학습이 진행됨에 따라 이상적인 값과 유사해짐을 시각적으로 확인할 수 있다. 다만 현재는 두 종류의 유사도가 서로 비슷한 형태를 가지며 변화하는데, 행동공간을 구성하는 각 성분의 값의 범위가 서로 다를 때에는 서로 다른 형태를 가지며 변화할 것으로 예상된다.

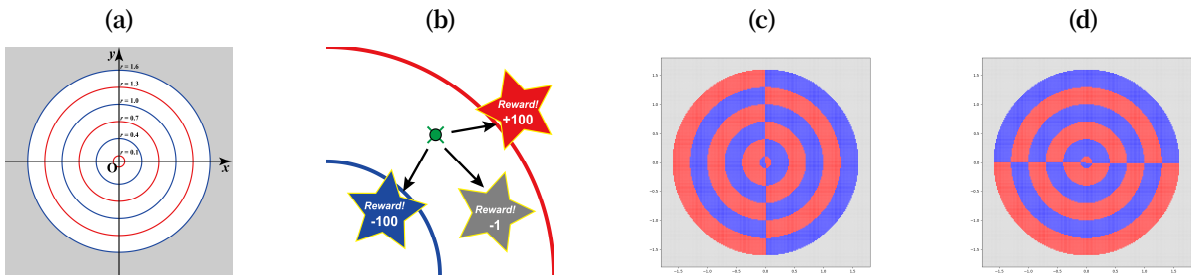
4.3 기존 평가 방법과의 비교

강화학습의 주요한 기존의 평가 지표인 보상 값의 변화, 가치망의 손실 값 변화와 본 논문에서 제시하는 행동 정확성 평가 지표인 평균 코사인 유사도, 평균 L2-노름과의 관계를 비교해보았다. <그림 2>의 (c), (d)는 기존 평가 지표인 보상 값의 변화와

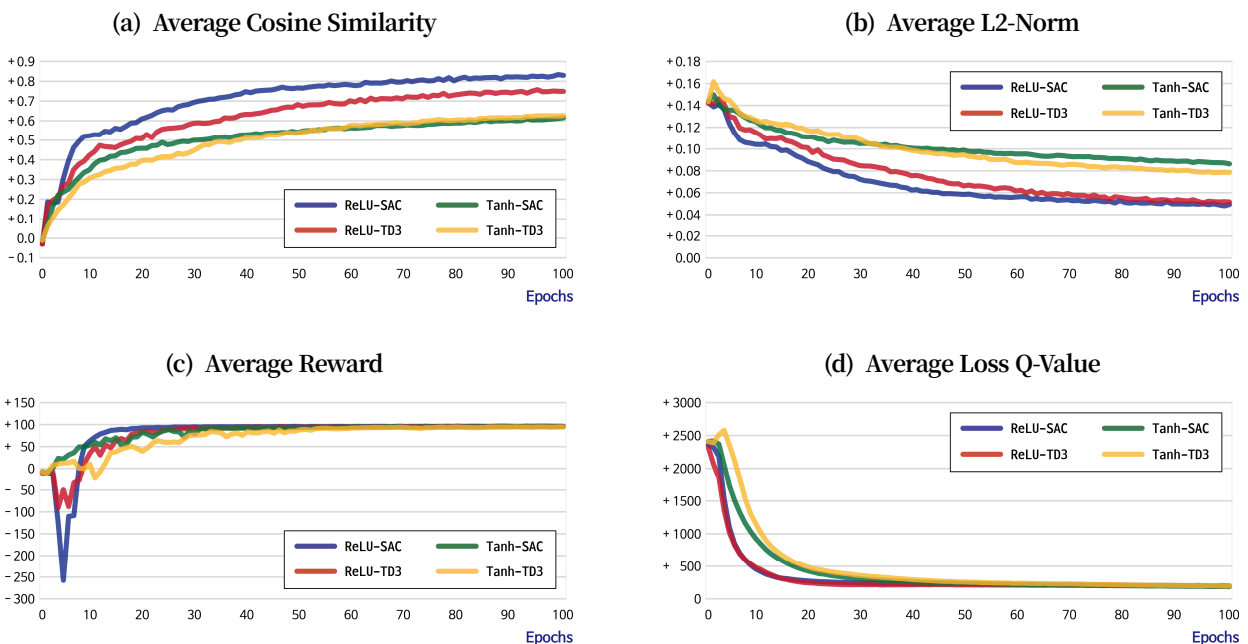
가치망의 손실 값의 변화를 나타낸 그래프이다.

강화학습 알고리즘과 사용된 활성화함수에 따라 약간의 차이가 있지만, 기존의 평가 지표인 평균 보상 값과 가치망의 손실 값은 비슷한 시점에서 수렴하는 것을 확인할 수 있다. 그런데 평균 보상 값이 일정하게 수렴하여 더 이상 증가하지 않는 시점인 약 60 에폭 이후에서도 <그림 2> (a), (b)에서 나타나는 행동 정확성은 지속적으로 증가하는 것을 확인할 수 있다. 일반적으로 강화학습에서는 평균 보상 값이 일정 수준 증가하여 수렴하게 되면 에이전트가 환경을 해결하였다고 간주하고 더 학습할 필요가 없다고 여겨지며, 계속된 학습은 과적합 문제를 발생시킨다고 알려져 있다. 그러나 본 논문에서 제시하는 행동 정확성은 보상 값이 수렴한 이후에도 계속적으로 증가하는 것을 확인할 수 있다. 이는 높은 행동 정확성을 요구하는 강화학습 시스템의 개발 과정에서 기존과는 다른 학습 중단 시점의 결정이 필요함을 의미한다.

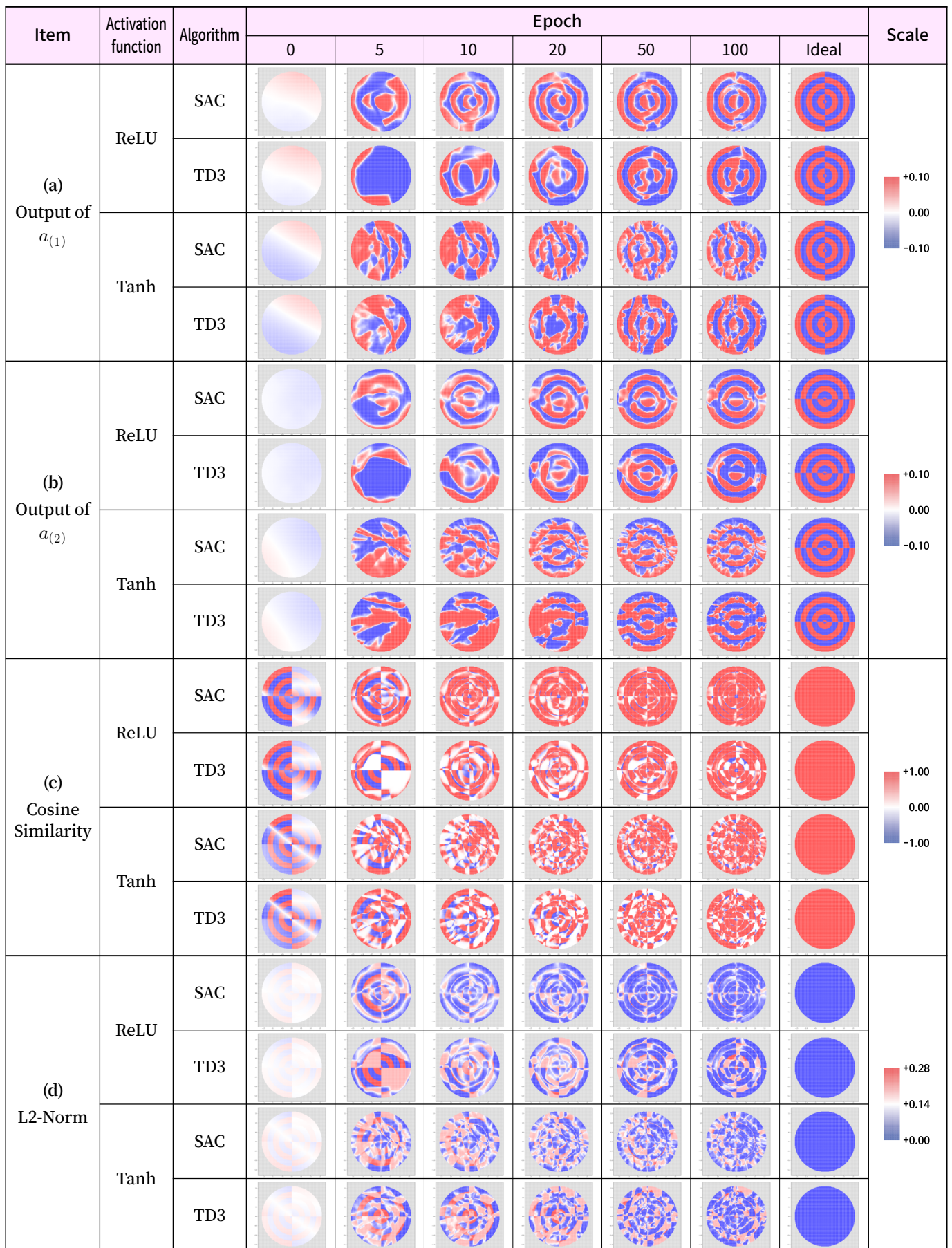
또한 학습이 진행됨에 따라 SAC 알고리즘과 TD3 알고리즘 모두 활성화함수와 관계없이 동일한 수준의 보상 값에 도달하



< 그림 1 > Evaluation environment



< 그림 2 > Results of each evaluation



< 그림 3 > Visualization of changes in each evaluation items

였지만, 행동 정확성은 알고리즘과 활성화함수의 차이에 따라 서로 다른 값을 나타낼 수 있다. 이러한 사실은 기존 보상 값 위주의 평가 방법만으로는 평가할 수 없는 강화학습의 또 다른 품질이 존재함을 의미하며, 본 연구에서 제시하는 행동 정확성 평가 방법이 강화학습의 이러한 다른 품질을 평가할 수 있는 하나의 지표가 된다는 것을 입증한다.

5. 결론 및 향후 연구

본 논문에서는 Actor-Critic 계열의 강화학습 알고리즘의 품질 중 하나인 행동 정확성을 정의하고, 이를 평가할 수 있는 방법에 대한 연구를 수행하였다. 보상함수에 따라 강화학습의 에이전트가 학습해야 할 이상적인 행동이 미리 정의된 환경에서 학습된 신경망의 출력 값을 이상적인 행동 값과 비교함으로써 행동 정확성을 평가하였다. 또한 현재 가장 널리 쓰이는 두 가지 종류의 알고리즘인 SAC, TD3을 다양한 활성화함수를 갖도록 학습시킨 후 행동 정확성을 평가함으로써 강화학습 알고리즘이 얼마나 정확한 행동을 출력하도록 학습을 수행하고 있는지를 확인하였다. 이러한 평가 결과로부터 기존의 평가 방법으로는 알 수 없었던 강화학습의 새로운 품질 속성을 확인하였고, 따라서 제시된 평가 방법이 강화학습의 품질을 평가하는데 유용하게 사용할 수 있음을 보였다.

본 논문의 연구 내용은 향후 강화학습의 품질 향상을 위하여 매우 다양하게 활용될 수 있다. 우선 여러 종류의 PIA 환경을 구성하고, 이러한 환경에서의 행동 정확성 평가를 통해 강화학습의 알고리즘, 활성화함수, 신경망의 크기 등 여러 요인에 따라 나타나는 특성을 분석하고자 한다. 또한 보상함수를 이용해 더 높은 행동 정확성을 갖는 에이전트를 학습시키는 방법에 대한 연구도 필요하다.

한편 행동 정확성은 강화학습의 다른 품질 속성인 강건성(Robustness), 안정성(Stability), 망각성(Obliviousness) 등을 평가하는 데 활용이 가능할 것으로 생각된다. 앞으로도 더 높은 성능을 갖는 강화학습의 에이전트를 만들기 위해 강화학습의 다양한 품질을 평가하고 향상시키는 연구를 지속적으로 진행하고자 한다.

Acknowledgement

본 연구는 과학기술정보통신부의 재원으로 한국연구재단의 지원을 받아 수행되었음. (NRF-2020R1A2C1007571)

참고 문헌

- [1] Silver, David, et al., "Mastering the game of go without human knowledge.", *Nature*, 550(7676), pp.354-359, 2017.
- [2] Fujii, Gaku, et al., "Guidelines for Quality Assurance of Machine Learning-Based Artificial Intelligence.", *International Journal of Software Engineering and Knowledge Engineering*, 30(11n12), pp.1589-1606, 2020.
- [3] Samek, Wojciech, et al., "Explainable AI: interpreting, explaining and visualizing deep learning.", *Springer Nature*, 11700, 2019.
- [4] Holzinger, Andreas., "From machine learning to explainable AI.", *Proceedings of the World Symposium on Digital Intelligence for Systems and Machines (DISA), IEEE*, pp.55-66, 2018.
- [5] Hoffman, Robert R., Gary Klein, and Shane T. Mueller., "Explaining explanation for "explainable ai".", *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, SAGE Publications, 62(1), pp.197-201, 2018.
- [6] Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models, *International Organization for Standardization*, 2017.
- [7] Hamada, Koichi, et al., "Guidelines for Quality Assurance of Machine Learning-based Artificial Intelligence.", *Proceedings of the International Journal of Software Engineering and Knowledge Engineering (SEKE)*, pp.335-341, 2020.
- [8] Siebert, Julien, et al., "Towards guidelines for assessing qualities of machine learning systems.", *Proceedings of the International Conference on the Quality of Information and Communications Technology*, Springer, Cham, pp.17-31, 2020.
- [9] Mnih, Volodymyr, et al., "Playing atari with deep reinforcement learning.", *arXiv preprint arXiv:1312.5602*, 2013.
- [10] Lillicrap, Timothy P., et al., "Continuous control with deep reinforcement learning.", *arXiv preprint arXiv:1509.02971*, 2015.
- [11] Mnih, Volodymyr, et al., "Asynchronous methods for deep reinforcement learning.", *Proceedings of the International conference on machine learning*, PMLR, pp.1928-1937, 2016.
- [12] Haarnoja, Tuomas, et al., "Soft actor-critic algorithms and applications.", *arXiv preprint arXiv:1812.05905*, 2018.
- [13] Fujimoto, Scott, Herke Hoof, and David Meger., "Addressing function approximation error in actor-critic methods.", *Proceedings of the International Conference on Machine Learning*, PMLR, pp.1587-1596, 2018.
- [14] Kuznetsov, Arsenii, et al., "Controlling overestimation bias with truncated mixture of continuous distributional quantile critics.", *Proceedings of the International Conference on Machine Learning*, PMLR, pp.5556-5566, 2020.
- [15] Brockman, Greg, et al., "Openai gym.", *arXiv preprint arXiv:1606.01540*, 2016.
- [16] Todorov, Emanuel, Tom Erez, and Yuval Tassa., "Mujoco: A

- physics engine for model-based control.", Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp.5026-5033, 2012.
- [17] Schneegass, Daniel, Steffen Udluft, and Thomas Martinetz., "Uncertainty propagation for quality assurance in reinforcement learning.", Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence). IEEE, pp.2588-2595, 2008.
- [18] Nakajima, Shin., "Quality assurance of machine learning software.", Proceedings of the 2018 IEEE 7th Global Conference on Consumer Electronics (GCCE), IEEE, pp.601-604, 2018.
- [19] Studer, Stefan, et al., "Towards CRISP-ML (Q): a machine learning process model with quality assurance methodology.", Machine Learning and Knowledge Extraction, 3(2), pp.392-413, 2021.
- [20] Poth, Alexander, et al., "Quality Assurance for Machine Learning -an approach to function and system safeguarding.", Proceedings of the 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS), IEEE, pp.22-29, 2020.
- [21] Henderson, Peter, et al., "Deep reinforcement learning that matters.", Proceedings of the AAAI conference on artificial intelligence, 32(1), pp., 2018.
- [22] Jordan, Scott, et al., "Evaluating the performance of reinforcement learning algorithms.", Proceedings of the International Conference on Machine Learning, PMLR, pp.4962-4973, 2020.
- [23] Chan, Stephanie CY, et al., "Measuring the reliability of reinforcement learning algorithms.", arXiv preprint arXiv:1912.05663, 2019.
- [24] Song, Xingyou, et al., "Observational overfitting in reinforcement learning.", arXiv preprint arXiv:1912.02975, 2019.
- [25] Li, Baoli, and Liping Han., "Distance weighted cosine similarity measure for text classification.", Proceedings of the International conference on intelligent data engineering and automated learning, Springer, pp.611-618, 2013.
- [26] Hahnloser, Richard HR, et al., "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit.", Nature, 405(6789), pp.947-951, 2000.

해석 가능한 시계열 이상탐지 모델을 활용한 생산 능력 계획 수립 방안

강민정^o, 이상민

광운대학교 정보융합학부

minjkang0910@gmail.com, smlee5679@kw.ac.kr

Explainable Time Series Anomaly Detection for Production Capacity Planning

Minjung Kang, Sangmin Lee
School of Information Convergence, Kwangwoon University

요 약

본 연구는 설명 가능한 이상탐지 기법을 활용하여 재공재고량의 정확한 예측을 통해 기업의 생산능력 계획 수립 방안을 제시한다. 재공재고량은 기업의 생산 운영비용에 직결되며 이에 대한 예측은 설비확장 및 생산량 증대계획에 중요한 기준이다. 본 논문은 재공재고량 데이터의 시계열적 상관성과 데이터 불균형 문제, 그리고 예측결과의 해석 문제를 동시 해결하기 위한 설명가능한 시계열 이상탐지 프레임워크를 제안한다. 기법으로는 LSTM-autoencoder 및 Shapley Additive exPlanations(SHAP)을 활용하였고 정확도 성능을 높이고자 다수의 이상탐지 알고리즘의 비교실험을 진행하였다. 실험 결과, 제안한 LSTM-autoencoder에서 가장 좋은 성능인 $f1$ score 98.18을 보였다. 또한 저평가된 재공재고량 예측 시의 중요 원인을 추적하기 위해서 대표적인 불특정 모델 해석 기법인 SHAP을 활용한 해석 모델을 구축하여 생산량 증대 계획에 차질이 되는 병목 자원을 특정 짓는데 활용하였다.

1. 연구 배경

최근에 반도체 수급 문제가 세계적인 이슈가 되고 있어서 반도체 양산에 많은 이목이 집중되고 있다. 반도체와 같이 대규모 생산시설내 최대 생산을 유지하기 위해서는 자동화된 공장시스템 가용능력의 계획 실행 방안이 필수적이다. 자동화 시스템의 가용능력에 대한 계획이 틀어질 경우 장기간의 대규모 생산성 저하가 불가피하다. 그러므로, 최대 보유 가능한 재공재고량을 의미하는 상한 재공재고량을 정확하게 예측하는 것에 대한 문제가 지속적으로 거론되고 있다. 특히 재공재고량 예측은 계획된 양산을 이행하기 위한 개략적 생산 능력 계획 (Rough Cut Capacity Planning, RCCP) 수립 시 가용능력 부족을 탐지할 수 있기 때문에 최대 생산을 위한 중요 전략으로 활용되고 있다.

상한재공량 예측에 대한 기존 연구는 크게 두 가지로 나뉘어질 수 있다. 개별 공정단위 별 재공재고의 도착시간 또는 양을 직접 예측하는 방식의 연구가 있다. [1,2] 재공재고량을 직접 추정하는 회귀모델을 사용하는 연구의 경우 정확한 수치를 추정하는 연구도 있었다 [3]. 세세한 수준의 재공재고를 예측하거나 재

공재고량 값을 직접 추정하는 것이 직관적이기는 하다. 하지만, 계획된 생산량 저하 여부를 판단하기 위한 계획의 이행 여부가 더욱 중요한 의사결정이 된다. 따라서 재공재고량 예측에 회귀(regression) 보다는 분류(classification) 측면에서 이를 구분하는 것이 정확할 수 있다 [4]. 하지만, 이미 양산 체계가 확립된 경우 최대 생산 계획을 이행하는 정상적인 상황 대비 계획을 미달하는 비정상적인 상황은 희박하다 [5]. 이러한 문제 상황에서 분류 학습모델을 구축할 경우, 상대적으로 관측치 확보가 손쉬운 정상 관측치의 클래스에 과대적합문제(overfitting) 되는 클래스 불균형 문제(class imbalance problem)가 발생한다. 또한 상한재공량의 정확한 예측을 위해서는 시계열적 자기상관성 (autocorrelation)이 높다는 측면 또한 고려되어야 한다. 이러한 문제를 해결하기 위해서, 본 연구에서는 상한재공량이 저하되는 상황을 조기 예측하고 이에 대한 원인을 추적하기 위한 프레임워크를 제안한다.

본 논문은 최대 생산 가능한 재공재고 수준을 정확하게 예측함과 동시에 최대 생산을 저하시키는 원인을 탐색하는 해석기법을 제안하였다. 첫째, LSTM-

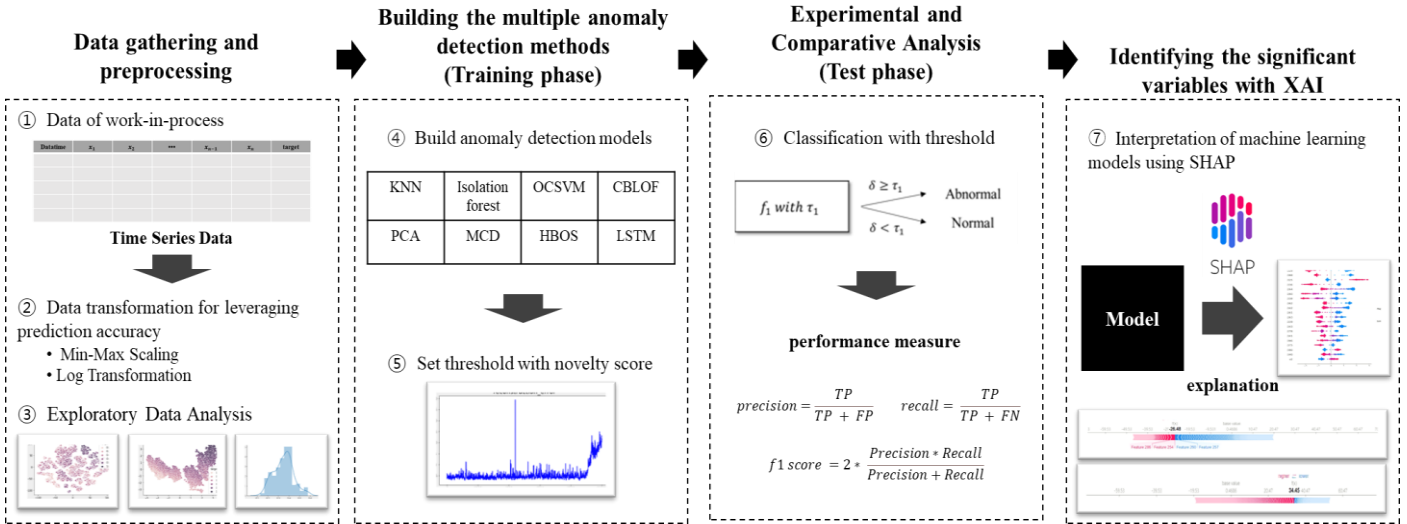


그림 1. 전체 실험 아키텍처

autoencoder(LSTM-AE)에 기반한 시계열성 이상탐지 기법을 적용하여 계획된 상한재공량의 저하를 정확하게 탐지하는 학습모델을 구축하였다. 10개 이상의 이상탐지 모델과 비교실험을 통해서 제안한 LSTM-AE 모델이 효과적임을 실험적으로 입증하였다. 둘째, 계획 대비 생산성이 저하되는 원인을 추정하기 위해 다양한 학습모델의 예측결과를 해석할 수 있는 학습모델 해석 기법을 적용하였다. 특히, 본 연구에서는 해석 가능한 인공지능의 기술로 많이 각광받는 SHAP 모델을 기반으로 상한재공량 저하의 주요 원인을 추론할 수 있었다.

본 논문 이후의 구성은 다음과 같다. 2장에서는 본 연구에서 제안한 전체 방법론 및 기법과 함께 실험 구성에 있어 단계 별로 설명한다. 3장에서는 예측 및 해석모델 적용에 따른 원인 탐지의 결과를 해석한다. 4장에서는 본 연구의 의의와 연구 한계점 그리고, 결론을 기술한다.

2. 제안 방법론

2.1 실험 아키텍처

그림 1은 본 논문에서 제안하는 전체 연구 절차 단계를 소개하고 있다. 첫째, 최대 재공재고량을 추산하고 자동화 시스템의 상태를 파악할 수 있는 반도체 공장의 센서 및 재공재고량에 대한 데이터를 수집하였다. 둘째, 학습 모델을 구축하기 전 데이터 상의 결측치를 확인하였고 기준에 따라 제거하였다. 또한, 데이터의 분포를 고려하여 스케일링의 전처리를 진행하였다. 셋째, 최대 재공재고량을 예측하기 위해 다수의 이상탐지 학습 모델을 구축하였다. 구축된 모델의 분류에 대한 예측 성능을 평가하는 여러 지표를 통해

각각의 모델 별 정확도를 확인하였다. 다수의 정확도 평가지표를 통한 비교 실험 결과 최적의 학습 모델을 선정하였다. 마지막으로, 최대 재공재고 저하를 예측한 경우의 중요 변수들을 선정하고 해석을 위해 해석 모델(surrogate model)을 구축하였다.

2.2 데이터 수집 및 전처리

본 논문에 사용된 데이터는 2019년 07월부터 2021년 04월까지 1시간 단위로 수집된 반도체 공장의 데이터이다. 데이터는 공장의 200여 개 다수의 반도체 공장내 HW/SW 자동화 시스템을 포함한 생산 및 물류 센서 데이터를 수집하였다. 이 때 각 시점의 최대 재공재고량을 종속변수(Y)로 설정하고 나머지 변수를 독립변수(X)로 설정하였다.

수집된 데이터 중 다수 변수에서 결측 값이 포함된 경우의 일부 변수 및 관측치는 대체법(imputation)의 효과를 보지 못하여 전체 제거법(complete case analysis)을 적용하였다. 개별 변수의 크기가 상이하여 리스케일링(rescaling)의 일환으로 정규화 방법으로 최소최대 스케일 변환(min-max transformation)과 함께 로그 변환(log transformation)을 적용하였다. 학습 데이터에서 예측하고자 하는 Y 는 최대 재공재고량이다. 주어진 Y 는 연속형 변수이지만 높게 또는 중간으로 예측되는 경우 큰 관심이 없고, 낮게 예측되는 경우의 상황만 관심있으므로 이를 분류하기 위한 모델을 구축하기로 했다. 또한, Y 에 대해 단순 정상/이상으로 이진 분류(binary class classification)하는 학습모델을 구축하기 에는 현장의 엔지니어가 탐지하고 싶어하는 이상의 경우가 $a\%$ 미만으로 극소량에 불과하므로 클래스 불균형 문제(class imbalance problem)가 불가피 했다. 따라서 본 연구에서는 현장의 엔지니어와 협의하여 하위

$\alpha\%$ 의 최대 재공재고량을 이상상태로 분류하였고, 대부분의 학습데이터를 구성하고 있는 최대 재공재고량이 낮지 않은 정상상태의 경우만을 학습하는 이상탐지 기법을 적용하였다. 이상탐지 학습모델을 적용할 경우 최대 재공재고량이 낮게 예측되는 이상에 해당하는 관측치는 학습 시 활용되지 않으나 테스트 시 학습모델들의 예측성능을 비교하는데 활용하였다.

2.3 탐색적 데이터 분석

분석에 활용되는 데이터의 패턴과 설명력 여부를 확인하기 위해 탐색적 데이터 분석을 진행하였다. 첫째, 본 연구에서는 선형적 관계성과 비선형적 관계성 두 가지 측면에서 데이터 내재적 특징을 살펴보고자 주성분분석(principal component analysis; PCA)과 t-분포 확률적 임베딩(t-Stochastic Neighbor Embedding; t-SNE) 기법을 적용하였다.

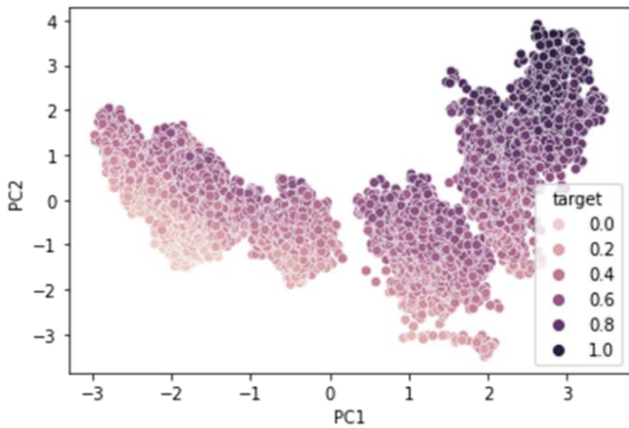


그림 2. PCA를 통해 차원 축소한 Plot

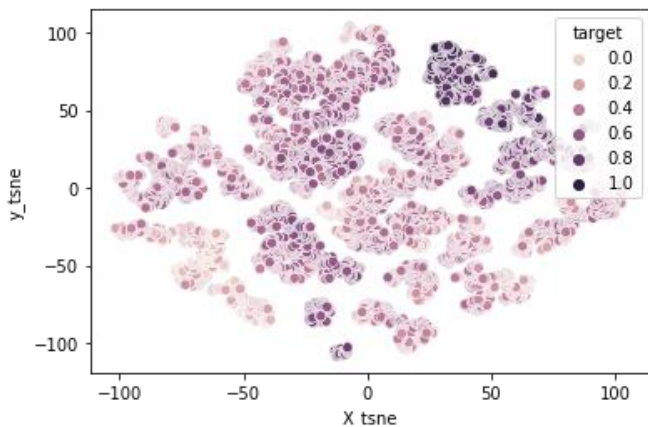


그림 3. t-SNE를 통해 차원 축소한 Plot

위의 그림 2와 그림 3은 각각 PCA와 t-SNE를 활용

해 2차원에 임베딩된 데이터 분포를 가시화하고 있다. 그림에서 개별 관측치의 색은 최대 재공재고량을 min-max scaler를 통해 스케일링한 값으로 보여주고 있고, 이 중 상대적으로 흐릿한 색일수록 관심있는 이상상황, 즉 최대 재공재고량이 낮게 예측된 경우라고 볼 수 있다.

개별 그래프에서 가로, 세로축은 차원 축소에 사용된 각 잠재 변수이며, 해석 방식은 관측치의 상대적인 위치와 인접성 정보이다. 두 그래프를 통해 최대 재공재고량이 낮은 경우와 중간 또는 높은 경우가 상대적으로 선형적 또는 비선형적으로 구분되는 구조이며 클래스 중첩(overlapping problem)은 발생하지 않는 것으로 확인했다. 특히, X 를 기준으로 최대 재공재고량이 높거나 중간 수준인 상황이 낮은 경우인 상황과 크게 겹치는 부분이 거의 존재하지 않고 임베딩된 것을 확인할 수 있다. 이는 수집된 데이터에서 X 가 Y 에 대한 학습모델 구축 시 충분한 설명이 있을 것으로 유추할 수 있다.

두번째로, 예측하고자 하는 최대 재공재고량 Y 값의 자기 상관성 여부를 확인하기로 하였다. 만약 높은 수준의 자기 상관성이 존재하는 경우 학습모델 구축 시 과거 시점의 Y 를 독립변수로서 사용할 수 있는 학습모델을 구축할 필요성을 검토해야 한다. 이를 확인하기 위해 자기상관함수(auto correlation Function: ACF)와 편자기상관함수(partial autocorrelation function: PACF)를 통해 확인하였다. ACF는 시차에 따른 일련의 자기 상관을 의미하여 시차가 커질수록 ACF는 0에 가까워진다. 이때 ACF 계산식은 일반 공분산을 구하는 식과 동일하며, 아래 수식 (1)은 y_t 와 y_{t+k} 사이의 자기 상관을 구하는 식이다.

$$ACF(\kappa) = \frac{\sum_{t=1}^{N-k} (y_t - \bar{y})(y_{t+k} - \bar{y})}{\sum_{t=1}^N (y_t - \bar{y})^2} \quad (1)$$

PACF는 ACF와 달리 y_t 와 y_{t+k} 간의 순수한 상관관계로 두 시점 사이에 포함된 모든 $y_{t-1}, y_{t-2}, \dots, y_{t-k+1}$ 와 y_{t+k} 의 영향은 제거된다. y_t 와 y_{t+k} 사이의 PACF를 구하는 식은 수식 (2)와 같다.

$$PACF(k) = Corr(e_t, e_{t-k}) \quad (2)$$

$$\text{where } e_t = y_t - (\beta_1 y_{t-1}) + \dots + \beta_{k-1} y_{t-(k-1)}$$

시계열 데이터의 ACF와 PACF를 시각화 하게 되면 lag가 각각 cut off와 die out 되는 패턴을 확인할 수 있다. Cut off는 특정 lag 이후에 급격하게 감소하거나 절단되는 경향을 보인다. 반면에 die down은 지속적으로 천천히 감소하거나 sin 함수의 형태로 소멸되는 특징이 있다. 본 실험에서 사용한 최대 재공재고량에

해당하는 Y는 ACF를 기준으로 die out, PACF를 기준으로 cut off의 형태를 보이는 것을 확인했다.

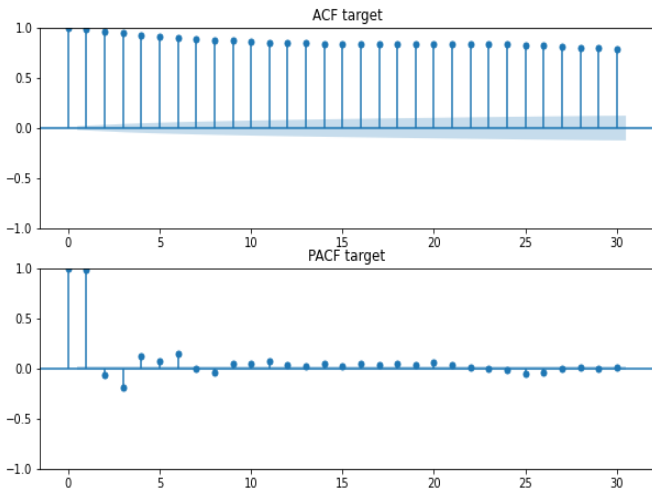


그림 4. 최대 재공재고량에 대한 ACF, PACF 그래프

따라서 ACF, PCAF 그래프를 통해서 우리는 학습 모델 구축 시 독립변수 X 및 종속변수 Y의 관계성을 결정짓는 최적의 학습모델 f^* 를 구축할 때 시계열적 상관성 정보 또한 활용할 수 있음을 판단하였다.

2.3 이상탐지 모델

본 연구에서는 최대 재공재고량을 정확하게 판정하기 위해서 다수의 이상탐지 알고리즘에 대해서 비교 실험을 진행하였다. 이상탐지 알고리즘은 정상 관측치와 상이한 패턴을 보이는 개별 관측치에 대해 이상치 점수(novelty score)를 산출하는 학습 모델로 구축하였다.

비교 실험을 위해 총 10개의 알고리즘이 사용되었다. k-Nearest Neighbors(KNN)은 각각의 데이터에 대한 이상치 점수를 k개의 근접한 이웃까지의 거리를 이용하여 계산하는 방식이다. Cluster-based local outlier factor(CBLOF)는 클러스터를 기반으로 거리를 계산하여 이상치 점수를 계산하며. Histogram based outlier detection(HBOS)은 히스토그램을 기반으로 이상치 점수를 계산하는 알고리즘이다. Minimum covariance determinant (MCD)는 소수의 이상치에 대한 영향을 최소화하는 방식으로 평균 및 공분산 행렬을 추정하는 기법이다. Principal component analysis(PCA)는 주성분 분석으로 고유 벡터 초평면에 대한 가중 투영 거리의 합을 이상치 점수로 사용하게 된다. One-class support vector machine(OCSVM)은 정상과 이상을 구분하는 결정경계(hyperplane) 구축을 위한 margin을 최대화하는 판별 함수를 기준으로 이상치를 판정하는 알고리즘이다. Isolation forest는 의사결정나무(decision tree)를 활용

하여 특정 한 개체가 분리되는 leaf 노드(terminal node)까지의 깊이를 계산하여 상대적으로 짧은 깊이로 분류할 수 있는지 여부를 이상치 점수로 산정하는 알고리즘이다.

이 중 본 연구에서는 오토인코더(autoencoder)를 활용한 이상탐지 기법의 활용을 제안하였다. Long short term memory(LSTM) 모델은 기존의 순환신경망 모델이 가진 기울기 소실 문제(vanishing gradient problem) 및 장기 의존성 문제를 해결한 기법이다. LSTM은 Recurrent Neural Network(RNN)의 hidden state에 cell state를 추가했으며, 기존의 은닉층 구조(hidden layer)를 바꾸었다. 은닉층 구조는 입력 게이트(input gate, i_t), 망각 게이트(forget gate, f_t), 출력 게이트(output gate, o_t)로 대체되며 각각 수식 (3) - (5)와 같은 식으로 계산된다. σ 는 활성화 함수(activation function)를 의미하고, x_t 는 t 시점의 입력 데이터(input data)이다. b 는 해당 시점의 바이어스 항(bias term)이며, W 는 신경망의 가중치를 나타낸다.

$$i_t = \sigma(W_{xh_f} \times x_t + W_{hh_f} \times h_{t-1} + b_{h_i}) \tag{3}$$

$$f_t = \sigma(W_{xh_i} \times x_t + W_{hh_i} \times h_{t-1} + b_{h_f}) \tag{4}$$

$$o_t = \sigma(W_{xh_o} \times x_t + W_{hh_o} \times h_{t-1} + b_{h_o}) \tag{5}$$

입력 게이트는 현재 시점에 새로 입력 받은 데이터의 영향력을 결정하고, 망각 게이트는 이전 시점의 데이터에 대해서 얼마만큼의 영향력을 반영할지를 결정한다. 출력 게이트는 현재의 셀 값을 얼마만큼 반영할지 결정하는 역할이다. LSTM의 전체적인 은닉층 구조는 그림 5와 같다.

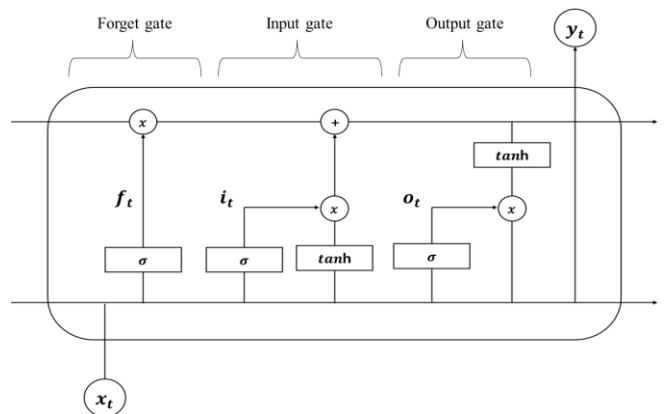


그림 5. LSTM의 은닉층 구조

오토인코더는 원본 데이터를 특징 벡터(feature vector)로 압축하고 복원하는 인코더(encoder)와 디코더(decoder) 두가지의 neural network를 활용하여 학습을 진행한다. 오토인코더는 정상 데이터만을 학습하여

출력 값과 입력 값 사이에 생기는 복원 에러를 이상치 점수로 사용하는 재구축 오차 기법(reconstruction error-based method)의 한 종류이다.

그림 6은 본 실험에 사용된 LSTM-AE의 구조이다. 두 개의 LSTM 레이어로 특징 벡터를 생성하게 되고 이 벡터를 디코더에 입력으로 전달해주어 원래 데이터의 형태로 변환하게 된다. 입력한 데이터와 재구축된 데이터의 차이를 재구축 오차(reconstruction error)라고 하며, 이를 측정하기 위해 MSE를 사용한다. 개별 관측치 별 측정된 MSE는 미리 설정한 임계치의 값을 비교해 이상 여부를 판단하게 된다. 그림 6은 본 연구에서 사용한 LSTM-AE의 구조이다.

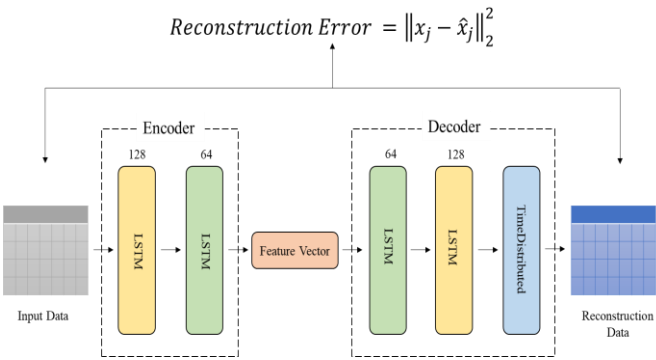


그림 6. LSTM-AE의 모델 구조

3. 실험결과

이번 장에서는 본 연구에서 제안한 예측모델들의 성능을 평가하기 위한 실험 설정과 테스트 데이터(Test data)에 기반한 예측성능 결과를 설명한다.

3.1 평가 지표

본 실험에서 사용된 이상탐지 학습모델들의 분류 예측 성능을 평가하기 위해서 정밀도(precision), 재현율(recall), true positive rate(TPR), false positive rate(FPR), f1 score를 활용했다. 정상 데이터만으로 이루어진 훈련 데이터와는 달리 테스트 결과에서는 정상데이터와 이상데이터의 비율을 거의 동일한 수준으로 구성했다.

$$precision = \frac{TP}{TP + FP} \tag{6}$$

$$recall = \frac{TP}{TP + FN} \tag{7}$$

$$TPR = \frac{TP}{TP + FN} \tag{8}$$

$$FPR = \frac{FP}{FP + TN} \tag{9}$$

$$f1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{10}$$

예측한 값의 정상(1), 이상(0)에 대한 구분은 이상치 점수로 해결했다. 그림 7은 각각의 모델에서 이상치 점수를 추출하는 과정을 나타낸다. Training phase에서 추출한 이상치 점수 값의 크기에 따라 sorting 하게 되고, 상위 5%에 해당하는 이상치 점수 값을 임계치로 설정했다. 여기서 임계치는 분석 데이터를 수취한 전 기간에서 제공재고 수준이 가장 낮은 하위 5%를 의미하며, 이 비율은 현업 엔지니어와 협의하였다.

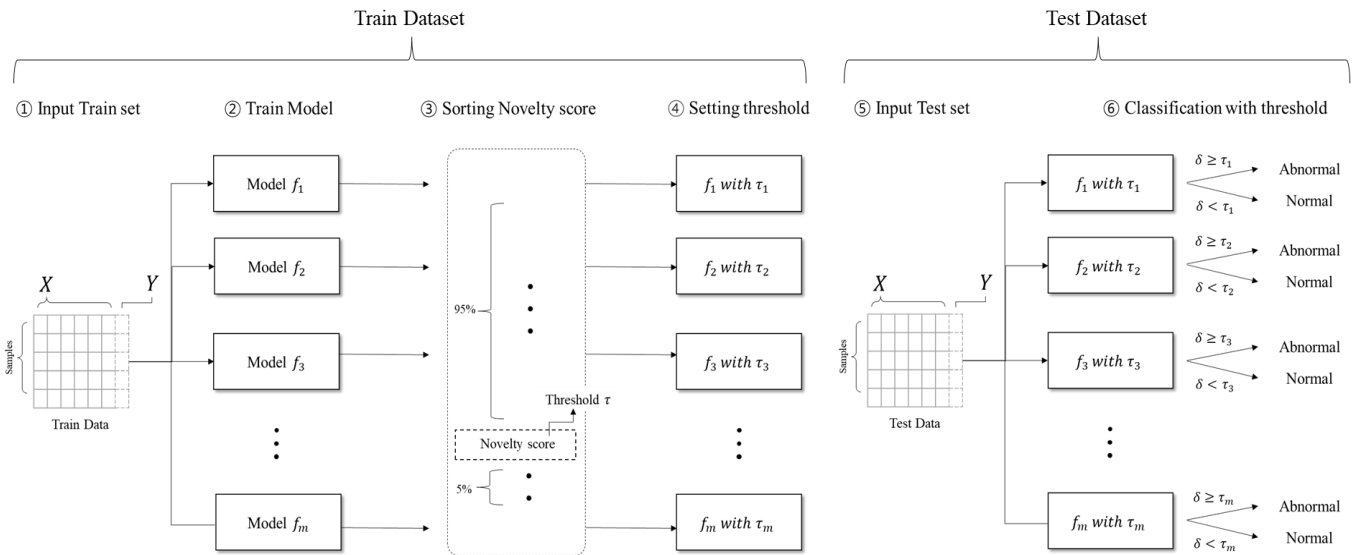


그림 7. 이상치 점수를 활용하여 이상 및 정상을 분류하는 전체 프로세스

표 1. 각 평가 지표에 따른 성능 비교

Algorithms	TPR	FPR	Precision	Recall	f1 score
k-Nearest Neighbors	0.72	0.48	0.6	0.72	0.61
Cluster-based local outlier factor	0.99	0.58	0.62	0.99	0.67
Histogram based outlier detection	0.75	0.17	0.80	0.75	0.79
Minimum covariance determinant	0.79	0.0	1.0	0.79	0.89
Principal component analysis	0.99	0.64	0.60	0.99	0.63
One-class support vector machine	0.95	0.58	0.61	0.95	0.65
Isolation Forest	0.93	0.19	0.82	0.93	0.87
Autoencoder	0.68	0.0	1.0	0.68	0.84
LSTM-autoencoder (3)	0.66	0.0	1.0	0.66	0.83
LSTM-autoencoder (5)	0.97	0.89	0.99	0.97	0.98

Test phase에서 같은 방법으로 이상치 점수를 얻은 후에 이상치 점수가 설정한 임계치보다 크면 이상(0), 작으면 정상(1)으로 분류했다. 모델에 대한 성능 평가는 데이터 전처리 시에 만들어 두었던 레이블링 한 값과 모델이 예측한 값과의 비교로 이루어졌다.

3.2 실험 결과

위의 표 1은 최대 재공재고량 예측에 대한 이상탐지 모델의 실험 결과이다. 이 중 본 연구에서 제안한 LSTM-AE 모델이 모든 지표에서 가장 좋은 예측 성능을 보였다. 특히, time step=5에서 우수한 성능을 보였으며, 이에 따라 재구축 기법의 이상탐지 방식과

시계열성 특징을 활용하는 것이 주요했음을 이해할 수 있다. 이러한 실험을 바탕으로 본 연구에서는 LSTM-AE를 최대 재공재고의 이상 여부를 판단하기 위한 이상탐지 모델로 최종 선정하였다. 이후 내용에서는 LSTM-AE 학습모델을 기준으로 해석 모델을 구축하여 개별 관측치의 예측에 대한 분석을 진행하고자 한다.

4. 해석 결과

학습을 통해 좋은 성능의 모델을 구축해도 블랙 박스 모델의 특성 상 예측 결과에 대한 명확한 해석이

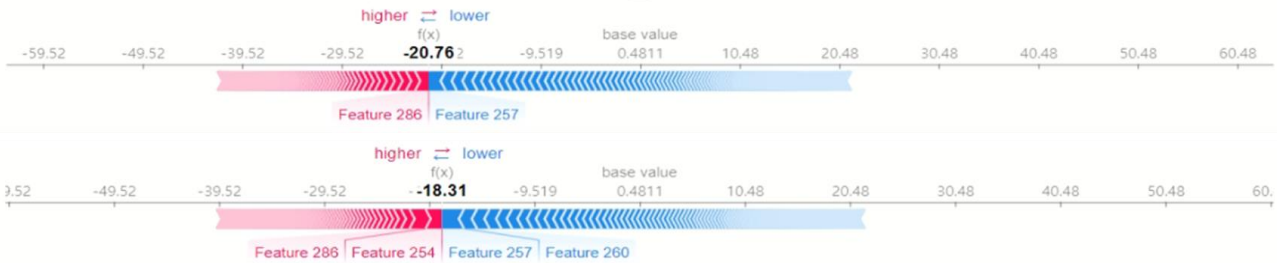


그림 8. Shapley value가 낮게 나왔던 관측치에 대한 force plot

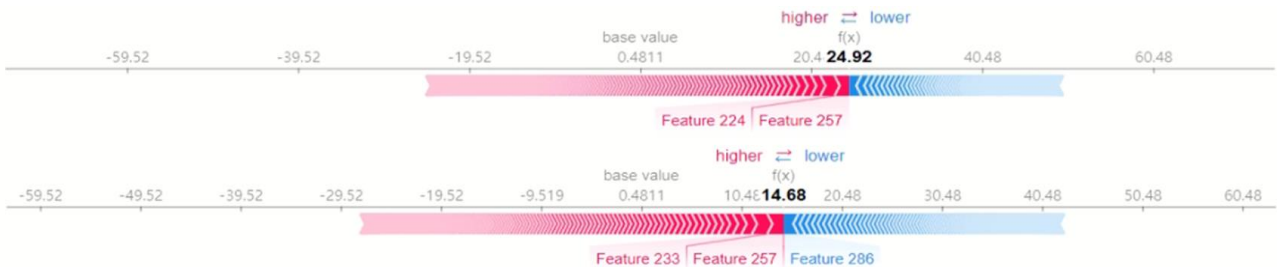


그림 9. Shapley value가 높게 나왔던 관측치에 대한 force plot

쉽지 않다. SHAP은 Shapely value를 통해 머신 러닝 모델의 해석 기법에 도움을 주는 모델이다.

아래는 특정 데이터 하나에 대해 Shapely value를 1차원 평면에 정렬해서 보여준 그래프이다. 특정한 데이터에 대해 Shapely value를 분해하고 시각화 했다고 할 수 있다. Base value는 train set에서의 모델 평균 예측 값이고 output value는 모델의 예측 값이다. 이 값에 가장 큰 영향을 준 변수의 값은 아래에 표시가 된다. 빨간색에 해당하는 부분은 예측 값을 더 높게 하는 변수의 영향도를 나타내며, 파란색에 해당하는 값은 반대로 예측 값을 더 낮게 하는 변수들의 영향도를 나타낸다. 각각의 arrows의 크기는 변수의 영향도에 대한 양을 나타낸다.

훈련 데이터(Test dataset)에서 종속변수에 해당하는 최대 재공재고량의 값이 높게 나왔던 샘플의 force plot이다. 즉, LSTM-AE 모델의 예측 값이 base value(=0.4686)보다 높게 예측한 경우에 해당한다.

Test dataset에서 종속변수에 해당하는 최대 재공재고량의 값이 낮게 나왔던 샘플의 force plot이다. 즉, LSTM-AE 모델의 예측 값이 base value(=0.4686)보다 낮게 예측한 경우에 해당한다. 모델의 예측 값을 낮추는데 가장 크게 기여한 변수는 feature 286, feature 254이다. 반대로, 예측 값을 높이는데 기여한 변수는 feature 260, feature 257 순이다.

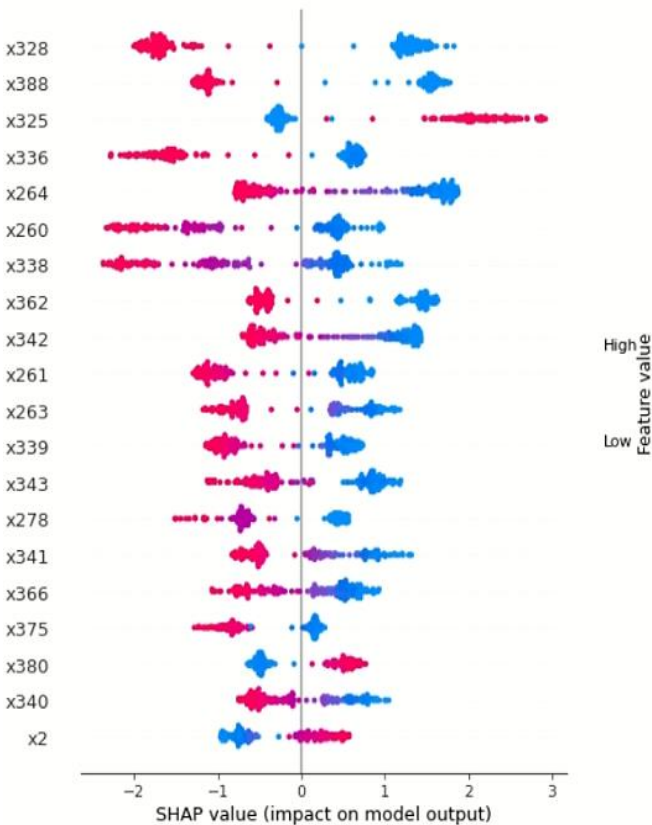


그림 10. 최대 재공재고량 해석 모델의 summary plot

그림 10은 변수 별 기여도를 확인할 수 있는 summary plot이다. Summary plot의 각 점은 변수(feature)에 대한 Shapely value와 관측치이며, x 축은 Shapely value에 의해 결정되고 y축은 변수에 의해 결정된다.

Summary plot에서 높은 기여를 했던 해당 변수들은 타워 호기 별 부하율과, 저장소인 STB 저장율에 관한 변수이다. 많은 변수들이 높은 값을 갖더라도 최대 재공재고 수준을 낮추는데 기여를 하고 있다. X328, x388, x336, x260과 같은 변수들이 최대 재공재고 저하의 주요 원인이 된다. 리프터를 기준으로 한 층간 반송에 대해 이상이 생겼거나, 층내 반송 중에서 STB의 저장률이 문제가 있어서 더 이상 증가하지 못하는 문제가 발생했다는 것을 확인하였다.

SHAP 모델로 확인한 변수들을 통해 반도체 공장의 최대 재공재고 수준의 저하를 유발하는 원인을 추론할 수 있다. 따라서, 예측 값에 기여가 많았던 변수들이 어떤 생산 및 물류 센서 데이터인지 확인하여 반도체 공장의 최대 재공재고량 예측에 정보를 제공할 수 있었다.

4. 결론

이 논문은 시계열 이상탐지 LSTM-AE를 활용하여 생산 능력 계획 수립을 위한 방법론을 제시하고 있다. 특히, 능력 계획이 부족한 상황의 주요 원인을 해석할 수 있도록 SHAP 프레임워크를 활용할 것을 제안했다. 제안 방법론에 대한 적용 결과, 현장 엔지니어가 공감할 수 있는 수준의 최대 재공재고 수준 저하의 예측 정확도 및 해석 능력을 선보였다. 본 연구에서는 시계열성 데이터를 이상탐지 모델 LSTM-AE에 적용하여 정확한 최대 재공재고량을 예측할 수 있었다. 대표적으로 사용되는 많은 이상탐지 알고리즘과의 비교를 통해 본 논문에서 제안한 LSTM-AE의 성능을 입증했다. 또한, 최대 생산을 저하시키는 원인을 추정하기 위해 해석 기법을 제안하였다. SHAP 모델을 기반으로 학습 모델의 예측 결과를 해석하여 상한 재공량 저하의 주요 원인을 추론할 수 있었다.

본 논문에서 데이터 변이 문제(dataset shift problem)에 대한 대응의 필요성을 확인하였다. LSTM-AE 모델이 가장 좋은 예측 성능을 보였지만 시계열 데이터의 구조에 따른 성능 결과의 차이가 심하다. 만약 데이터의 변이 문제가 있을 경우 성능이 급격히 저하되며 이로 인해 훈련된 모델은 재사용되기 어렵다. 이는 사용자가 매 실험마다 새롭게 훈련하는 절차를 반복해야 하는 문제점을 가져온다. LSTM-AE 모델을 적용하는데 있어 데이터 분포 변화에 따른 문제를 고려해야 하는 추가적인 연구가 필요하다.

이러한 한계점을 극복하고자 추후 데이터의 분포 문제를 해결할 수 있는 연구 및 LSTM-AE 외 최근 등장한 **generative approach** 방식의 딥러닝 기반의 이상 탐지 모델을 확장한 연구를 수행할 계획이다. 추가적으로, 생산 능력 계획 수립이 필요한 여타 분야의 상황에도 확장 연구를 수행할 계획이다.

5. 참고 문헌

1. Chien, C. F., Kuo, C. J., & Yu, C. M. (2020). Tool allocation to smooth work-in-process for cycle time reduction and an empirical study. *Annals of Operations Research*, 290(1), 1009–1033.
2. Yang, T., Fu, H. P., & Yang, K. Y. (2007). An evolutionary-simulation approach for the optimization of multi-constant work-in-process strategy—a case study. *International Journal of Production Economics*, 107(1), 104–114.
3. 방성진, 고해지, & 이상민. (2021). TabNet 기반의 반도체 재공재고 수준의 조기예측과 해석. *한국정보과학회 학술발표논문집*, 673–675.
4. Gallina, V., Lingitz, L., Breitschopf, J., Zudor, E., & Sihn, W. (2021). Work in Progress Level Prediction with Long Short-Term Memory Recurrent Neural Network. *Procedia Manufacturing*, 54, 136–141.
5. Lee, S., Kim, H. J., & Kim, S. B. (2020). Dynamic dispatching system using a deep denoising autoencoder for semiconductor manufacturing. *Applied Soft Computing*, 86, 105904.

코드 변경에 대한 검출 기법과 사람의 묘사 차이 연구

김무준, 김범철, 김진대

서울과학기술대학교

{mjkim, kkc217, jindae.kim}@seoultech.ac.kr

A Study on Change Description Differences between Human and Differencing Techniques

Moojun Kim, Beomcheol Kim, Jindae Kim

Seoul National University of Science and Technology

요 약

본 연구에서는 코드 변경 검출 기법과 사람이 코드 변경을 묘사하는 방식을 조사하고, 둘을 비교하여 그 차이를 확인하였다. 대표적인 두 가지 코드 변경 검출 기법을 수집된 다량의 코드 변경에 적용한 결과, 두 기법은 같은 코드 변경을 다른 방식으로 묘사하는 경우가 매우 많으며, 3% 미만의 코드 변경에 대해서만 완전히 일치하는 결과물을 내놓는다는 것을 확인하였다. 또한 사람의 경우 동일한 코드 변경을 다른 방식으로 묘사하는 경우의 상당 부분은 실수에 의한 것이고, 대체적으로 동일한 방식으로 코드 변경을 묘사하는 것을 확인할 수 있다. 사람과 검출 기법의 비교 결과에서는 검출 기법이 사람이 코드 변경을 묘사한 방식과 일치하는 결과를 내놓는 경우가 약 3.5%에 불과하여 많은 개선이 필요함을 알 수 있었다.

키워드 : 코드 변경 검출 기법, 코드 변경 검출 기법 평가, 변경 스크립트, 경험적 연구

1. 서 론

소프트웨어 개발에 있어서 코드의 변경된 부분을 파악하고 이해하는 것은 매우 중요한 작업 중 하나이다. Git과 GitHub 등과 같은 오픈소스 프로젝트 호스팅 서비스에서 코드 변경 검출(Source Code Differencing) 기법을 사용하고 있으며, 이런 코드 변경 검출 기법은 코드 변경과 관련된 다양한 연구에서도 활용된다. 따라서, 코드 변경 검출 기법의 신뢰도는 이를 활용한 연구 결과의 신뢰도에도 영향을 미치게 된다. 하지만 최근의 결과를 보면 연구에서 활용된 코드 변경 검출 기법이 부정확한 결과를 내놓는다는 점이 확인되었다[1-2].

본 연구에서는 이런 코드 변경 검출 기법이 활용되는 연구에서 검출 기법이 내놓는 결과에 따른 영향을 분석하고자 하였다. 첫째로 다양한 연구에서 활용되었던 코드 변경 검출 기법인 ChangeDistiller[3]와 GumTree[4]가 코드 변경을 묘사하는 방식을 확인하고, 두 기법이 어떤 차이를 보이는지 확인하였다. 둘째, 사람이 코드 변경을 어떤 식으로 묘사하는지 그 방식을 확인하였다. 마지막으로는 검출 기법과 사람이 동일한 코드 변경을 묘사하는 방식을 비교하여 실제 코드 변경 검출 기법이 사람이 생각하는 것과 유사한 결과물을 제공하고 있는지 확인하였다.

2. 배경 이론

GumTree의 변경 스크립트로부터 얻을 수 있는 변경

명령의 종류는 총 네 가지로 Update, Add, Delete, Move가 있다. 각 종류에 대한 설명은 다음과 같다.

·**Update**: 값만 바뀐 경우로 함수명이 변하거나 변수에 할당된 값만 바뀐 상황 등이 포함된다.

·**Add**: 추상 구문 트리(Abstract Syntax Tree, AST) 상에서 기존 코드에는 없던 노드가 생긴 것으로 새로 추가된 코드가 있는 경우를 의미한다.

·**Delete**: 원래 있던 노드가 사라진 것을 의미한다.

·**Move**: AST 상에서 기존에 있던 노드가 다른 부모 노드의 자식 노드로 이동한 것을 의미한다.

ChangeDistiller의 경우 이 네 가지 형태의 변경 명령의 종류로 파악된 코드 변경에서 이를 설명해 줄 수 있는 보다 많은 종류의 변경 명령을 생성하나, 본 연구에서는 GumTree와의 원활한 비교를 위해 ChangeDistiller가 구현된 코드를 수정하여 GumTree와 같은 네 가지 형태의 변경 명령을 출력하도록 수정하였다. 또한, 이 네 가지 변경 명령을 제시하는 경우 사람이 코드 변경을 묘사하는 방식을 확인하고자 하였다.

3. 데이터 수집

총 10개의 오픈소스 프로젝트에서 프로젝트별로 1만 개의 변경된 파일을 무작위로 추출하여 총 10만 개의 파일에 ChangeDistiller와 GumTree 기법을 적용하여 변경 스크립트를 추출하였다. 마찬가지로 동일 프로젝트에서 500개의 변경된 파일을 수집하여 6명의

참가자에게 직접 구현한 ChangeMarker 도구를 사용하여 변경 스크립트를 생성하도록 한 뒤 수집된 스크립트를 분석하였다. 또한 같은 500개의 파일에 ChangeDistiller와 GumTree 기법을 적용하여 뽑아낸 변경 스크립트를 비교하였다.

4. 분석 결과

10만 개의 파일로부터 변경 스크립트를 추출한 결과 ChangeDistiller의 경우 22.5%, GumTree의 경우 47.9% 정도의 파일에서만 변경 스크립트를 추출할 수 있었다. 이는 GumTree가 ChangeDistiller보다 세부적인 statement를 비교하기 때문에 더 많은 변경 사항을 감지할 수 있었던 것으로 보인다. 또한 ChangeType, EntityType, 시작 지점이 모두 일치하는 파일의 수는 2,646개로, 전체 파일의 약 2.6%에 해당하는 매우 적은 양임을 알 수 있다. 또한 다음의 표는 두 기법을 적용하였을 때 빈도수가 높은 순으로 ChangeType, EntityType 쌍을 나열한 것으로, 사용된 기법에 따라 통계적인 결과가 달라진다는 것을 볼 수 있다.

표 1 빈도수가 높은 ChangeType과 EntityType 쌍

ChangeDistiller	GumTree
ChangeType & EntityType	ChangeType & EntityType
UPD MethodInvocation	UPD SimpleName
MOV MethodInvocation	MOV SimpleName
INS MethodInvocation	MOV ExpressionStatement

다음으로, 6명의 참가자에게서 500개의 파일에 대한 변경 스크립트를 수집하고, 각각의 파일별로 동일한 스크립트를 집단으로 묶었다. 또한 서로 다른 스크립트가 작성된 경우 그 이유가 실수로 인한 것은 아닌지 확인하였다. 그 결과, 전체 500개 중 참가자의 의견이 대체로 일치한 경우는 329(64.8%)개이고, 단순한 실수로 달라진 변경 스크립트를 무시하면 참가자 의견이 일치하는 경우는 411(82.2%)개였다. 따라서 사람은 코드 변경을 대체로 동일하게 묘사한다는 것을 알 수 있다.

마지막으로, 각각의 참가자로부터 얻은 변경 스크립트를 ChangeDistiller, GumTree와 비교하여 일치하는 변경 스크립트의 비중이 전체 스크립트 중에서 얼마나 되는지 비교하였다. ChangeType이 사람의 변경 스크립트와 일치하는 경우는 ChangeDistiller는 약 33%, GumTree는 약 21% 정도였으며, ChangeType과 EntityType이 모두 일치하는지를 비교한 결과는 두 기법 모두 약 3.5%에 그쳐 매우 낮은 일치율을 보여주었다.

사람의 변경 스크립트에서 다수 집단이 있는 경우 다수 집단의 코드를 사용하여 두 기법과 변경 스크립트를 추가로 비교한 결과, ChangeType이 사람의

변경 스크립트와 일치하는 경우는 ChangeDistiller는 약 35%, GumTree는 약 44%로 GumTree에서 더욱 높은 수치의 일치율을 보여주며, ChangeType과 EntityType이 모두 일치하는지를 비교한 결과는 약 3.4%, 약 5.4%로, 여전히 매우 낮은 일치율을 보여주었다. 이는 사람의 변경 스크립트에서 6명의 참가자가 82.2%의 일치율을 보인 것에 비하면 여전히 매우 낮은 수치이다.

5. 결론

본 연구에서는 코드 변경 검출 기법과 사람이 코드 변경을 묘사하는 방식을 확인하고, 둘을 비교하여 그 차이를 알아보았다. 그 결과, 어떤 기법을 사용하느냐에 따라 매우 다른 결과를 얻을 수 있음을 알 수 있었으며, 사람은 일부 실수인 경우를 제외하면 대체로 같은 방식으로 코드 변경을 묘사하지만, 검출 기법은 사람이 묘사한 방식과는 상당히 다른 스크립트를 생성하고 있음을 확인하였다.

이런 결과는 코드 변경 검출 기법이 아직 많이 개선될 여지가 있다는 것을 알려준다. 특히, 사람이 코드 변경을 묘사하는 방식은 상당 부분 일치하는데 이런 묘사와 불일치하는 변경 스크립트를 생성하는 문제는 코드 변경 검출 기법의 활용도를 고려했을 때 반드시 해결되어야 하는 문제이다. 이를 위한 후속 연구로 현재의 연구를 더욱 발전시켜 사람이 작성한 변경 스크립트를 이용해 코드 변경 검출 기법의 성능을 검증하는 방법을 개발하는 것을 생각해 볼 수 있다.

Acknowledgement

이 연구는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2021R1G1A1094871)

참고 문헌

[1] De la Torre, Guillermo, Romain Robbes, and Alexandre Bergel. "Imprecisions diagnostic in source code deltas." Proceedings of the 15th International Conference on Mining Software Repositories. 2018.

[2] Fan, Yuanrui, et al. "A Differential Testing Approach for Evaluating Abstract Syntax Tree Mapping Algorithms." 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)

[3] Fluri, Beat, et al. "Change distilling: Tree differencing for fine-grained source code change extraction." IEEE Transactions on software engineering 33.11 (2007)

[4] Falleri, Jean-Rémy, et al. "Fine-grained and accurate source code differencing." Proceedings of the 29th ACM/IEEE international conference on Automated software engineering. 2014.

흉부 방사선 영상 학습 성능 개선을 위한 오픈 데이터셋 품질 평가 기법

김양곤^{1(○)}, 박예슬¹, 선주성², 이정원¹

아주대학교 AI융합네트워크학과¹, 아주대학교 의과대학 영상의학과²

e-mail: djwoajs96@ajou.ac.kr, yeseuly@gmail.com,

sunnahn@ajou.ac.kr, jungwony@ajou.ac.kr

Open Dataset Quality Evaluation for Performance

Improvement of Chest X-Ray Image-based Learning Model

Yang-Gon Kim^{1(○)}, Ye-Seul Park¹, Joo-Sung Sun² and Jung-Won Lee¹

Department of AI Convergence Network, Ajou University¹,

Department of Radiology, Ajou University Hospital²

요 약

최근 개발되고 있는 흉부 질병 진단을 위한 학습 모델은, 학습에 영향을 미치는 의료 데이터의 품질이 매우 중요하다. 그러나 의료 데이터는 개인정보보호 문제로 인해 획득 자체가 어려우며, 높은 품질의 데이터를 확보하는 것은 더욱 어렵다. 흉부 질병 진단 연구를 위해 일본 방사선 기술 학회(JSRT), 미국 국립보건원(NIH)과 같은 공신력 있는 의료 연구 기관에서는 오픈 데이터셋을 구축해왔다. 기관에 따라 차이는 있지만, 오픈 데이터셋은 흉부 방사선 전문의를 통해 검증되어, 인공음영, 환자자세 불량, 폐가 뿌연 상태 등의 결함 요소가 배제된 높은 품질의 데이터셋을 배포해 왔지만 영상의 수가 부족한 실정이다. 또한 영상 진단 기술이 확산되면서 품질이 보장되지 않은 오픈데이터셋이 배포되고 있으며 이를 이용한 학습 모델이 개발되고 있다. 그러나 기존 품질 자동 평가 연구에서는 단순히 품질 적합 여부(적절/부적절)만을 파악하고 있어, 가이드라인에서 제시하고 있는 세부 품질 요인에 대한 추적이 어렵다. 따라서 본 논문에서는 세부 품질 평가 요소에 대한 추적성을 확보하고, 정밀한 품질 평가를 수행할 수 있는 의료 영상 화질 평가 가이드라인 기반 품질 평가 방법을 제안한다. 제안하는 방법은 품질에 영향을 미치는 해부학적 요소(폐 영역, 흉곽 영역)를 검출하고, JSRT에서 제공하는 높은 품질의 데이터셋에 대한 통계적인 분석을 통해 품질 평가 기준을 확립하는 방식이다. 오픈 데이터 셋을 이용하여 폐 결절(Nodules)을 진단하기 위한 모델 개발에 적용한 결과, 낮은 품질 데이터셋과 높은 품질 데이터셋에 대해 각각 학습한 결과, 최대 AUC 4.5%, 정확도 4.6%의 성능 차이가 발생함을 확인하였다.

1. 서 론ⁱ

흉부 방사선 영상은 비용이 저렴하고, 간단한 촬영 과정이면서 결과가 빠르게 나오는 장점이 있어, 초기 폐 질환을 진단할 경우 널리 쓰인다[1]. 또한, 흉부 방사선 영상을 통한 초기 폐 질환 진단은 다음 치료(Following Treatment)에 대한 계속/중지 여부를 결정하는데 중요한 요인으로 작용될 수 있다[2]. 예를 들어, 향후 추가적인 검사(예: 초음파, CT, MRI 등)를 통한 불필요한 절차를 진행하지 않아 비용적인 큰 이점을 가질 수 있다. 그러나 흉부 방사선 영상을 통한 진단은 주관적인 성향이 강하여 관찰자(예: 의사, 전문의 등)마다 경험의 차이로 인한 정확도 편차가 있다. 따라서 의료 영상 해석을 보조하는 질병 진단(혹은 병변 감지) 모델이 개발되고 있다[3].

최근 개발되고 있는 질병 진단 모델은 영상 데이터의

학습을 통해 이루어지므로, 학습에 영향을 미치는 데이터의 품질이 중요해지고 있다. 구체적으로, 학습을 통한 질병 진단 모델은 모델을 구성하는 내부 파라미터가 학습에 사용된 데이터에 의해 결정된다. 또한, 통계적으로 학습 데이터의 품질은 질병 진단 모델의 학습 성능과 검증에 대한 신뢰성에 비례하여, 낮은 품질의 흉부 방사선 영상을 통한 학습 모델은 거짓 양성 오류(False Positive)와 거짓 음성 오류(False Negative)를 발생시킨다[4]. 구체적으로 거짓 양성 오류는 불필요한 추가적인 검사를 유도하고, 거짓 음성 오류는 질병의 존재를 없다고 판단하는 치명적인 문제가 생긴다. 따라서 좋은 성능의 질병 진단 모델을 개발을 위해 높은 품질의 학습 데이터가 필요하다.

그러나 의료 데이터는 개인정보보호 문제로 인해 획득 자체가 어려우며, 높은 품질의 데이터를 확보하는 것은 더욱 어렵다. 이로 인해, 일본 방사선 기술 학회(JSRT: Japanese Society of Radiological Technology)[5], NIH(National Institute of Health) 임상 센터와 같은 공신력 있는 의료 연구 기관[6]에서는

ⁱ 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음(IIITP-2021-2020-0-01461)

흉부 방사선 영상을 통한 질병 진단 모델 개발을 위해 오픈 데이터 셋을 구축해왔다. 그 중에서도 JSRT 데이터베이스(Database)는 높은 품질의 데이터가 축적되어 있는 대표적인 흉부 방사선 영상 오픈 데이터셋으로 그림 1과 같이 예시된다[5]. JSRT 데이터셋은 흉부 방사선 전문의를 통해 검증되어 그림 1(a)-(c)는 공통적으로 질병 진단을 위한 폐 영역의 흉곽, 심장, 횡격막 등의 구조물의 대비 및 밝기가 일관된 형태로 품질이 매우 우수하여 흉부 진단 데이터로 많은 논문에서 실험 데이터로 이용하고 있지만, 총 이미지의 개수가 247개로 학습에 사용하기에 적은 편이다.

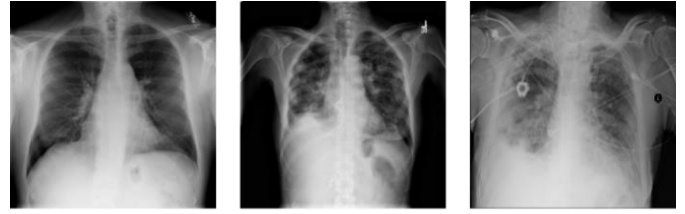


그림 2 Chest X-ray14 데이터셋 예시

위치나 분포 등을 분석해야 하기 때문에, 관련 의료 지식 없이는 쉽게 품질을 평가하기 어렵다. 따라서 학습을 위한 데이터 레이블링(Data Labeling) 작업에 있어, 비전문가인 모델 개발자가 데이터의 품질을 평가하는데 큰 한계가 존재한다. 이로 인해, [11]와 같이 현재 개발되고 있는 품질 평가 모델의 경우, 단순히 흉부 방사선 영상의 품질 적합 여부(적절/부적절)만을 파악하고 있어, 세부 품질 요인에 대한 추적이 어렵다.

따라서 본 논문에서는 세부 품질 평가 요소에 대한 추적성을 확보하고, 정밀한 품질 평가를 수행할 수 있는 통합 품질 평가 방법을 제시한다. 먼저, 국내 A병원의 영상 화질 평가 기준표(표 1)와 가이드라인[7]을 분석하여 일반 항목은 제외하고 영상의 주요 품질 평가 요소인 ‘포함 범위, 환자 자세, 흡기 수준’을 평가하기 위한 기법을 제안한다. 먼저, 폐 영역 추출과 흉곽 검출을 통한 주요 품질 요소들의 평가 기법을 설계한다. 또한, 높은 품질 오픈 데이터셋(JSRT)을 통계적으로 분석하여 세부 품질 평가 기준을 정립하고, 개발된 모델을 대규모 오픈 데이터셋(Chest X-ray14)에 적용하여 제안하는 방법의 효용성을 입증한다. 오픈 데이터 셋을 이용하여 폐 결절(Nodules)을 진단하기 위한 모델 개발에 적용하여 실험한 결과, Chest X-ray14 데이터셋에서 폐 결절과 병변 없는 정상 영상만 추출한 12,135개 영상 중 낮은 품질의 데이터셋 6,554개(54%), 높은 품질의 데이터셋 5,581개(46%)가 도출되었으며, 낮은 품질 데이터셋과 높은 품질 데이터셋에 대해 각각 학습한 결과, 최대 AUC 4.5%, 정확도 4.6%의 성능 차이가 발생함을 확인하였다.



그림 1 JSRT 데이터베이스의 영상 예시

반면, Chest X-ray14는 정면 이미지 기준으로 112,120개를 가지며, 폐 관련 질병 14가지(폐렴, 심장비대, 폐 결절 등)에 대한 정보를 갖는다[6]. 해당 데이터 셋은 타 데이터셋보다 규모가 큰 장점이 있지만, 그림 2에서와 같이 낮은 품질 수준의 데이터가 다수 포함되어 있다. 예를 들어, 그림 2(a)와 같이 품질에 문제없는 데이터셋 뿐만 아니라 그림 2(b), 그림 2(c)와 같이 폐 영역이 부족하게 포함되었거나 인공 음영, 환자자세 불량, 폐가 뿌연 상태(예: 호기 상태 촬영) 등으로 폐 질병 진단에 부적절한 영상이 존재한다.

인공 음영, 환자 자세 불량 등과 같은 영상 품질 요인은 흉부 방사선 영상에 대한 병원 간의 기준 및 가이드라인[7]에 제시되어 있다. 하지만 제시되어 있는 흉부 방사선 영상의 세부 품질 요소들은 해부학적인 구조물(예: 갈비뼈, 척추, 횡격막 등)의

표 1 흉부 방사선 영상 화질 평가 기준표(A 병원 흉부영상학과 품질 평가 기준)

정보	항목	평가내용
일반	검사표지	환자 이름, 성별, 나이, 등록번호, 촬영날짜, 촬영 기간 등의 표지 유무
	검사표지 위치	표지가 흉부포함범위를 가리는데 대한 유무
	위치표시	좌우 표시 유무
	현상조건의 적정성	제출된 사진에 노화(황화) 현상 유무
영상	인공음영	외부 및 내부에 의한 인공음영(부착물, 얼룩, 인공물 등)의 유무와 진단 방해 정도
	포함범위	상단(제 1늑골), 하단(횡격막 기준 3cm), 좌우(전체 늑골)의 포함 정도
	환자자세	견갑골의 올바른 위치와 좌우 대칭의 정도
	흡기정도	정상 흡기 상태(10늑골과 횡격막의 위치)의 정도
	투과상태 (해상도, 대조도)	폐혈관, 대동맥, 횡격막의 혈관 및 늑골, 척추 추간판 공간, 기관지 관찰 정도

2. 관련 연구

2.1 흉부 방사선 영상 품질 평가 요소 분석

흉부 방사선 영상의 품질 평가 요소는 표 1과 같이 두 가지 범주인 일반과 영상으로 구분된다[7]. 일반 정보의 항목들은 흉부 방사선 영상 촬영할 때 촬영자가 영상에 남기는 정보(예: 텍스트)에 대한 품질 평가 항목으로 폐 영상에는 직접적인 관계가 없다. 반면, 영상 정보 항목들은 영상 촬영 장비, 파라미터 설정 등 폐 영상 자체에 대한 평가 요소를 나타낸다. 본 연구에서는 영상의 품질을 평가하기 위해, 다섯가지의 영상 품질 평가 요소에 주목하였다. 영상 정보의 항목들은 총 5가지로 인공음영, 포함범위, 환자자세, 흡기정도, 투과상태로 나뉜다. 먼저, 포함범위, 환자자세, 흡기정도는 주로 보이는 폐 영역이나 구조물(갈비뼈, 척추, 횡격막 등)의 위치 등 해부학적인 특징을 통한 평가를 나타낸다. 또한, 인공음영의 경우 외부 및 내부에 존재하는 부착물(노이즈)의 유무이며, 투과상태는 방사선 촬영의 파라미터(kVm, mAs 등)에 따른 영상의 해상도와 대조도를 의미한다. 이때, 인공음영이나 투과상태의 경우, 해상도와 대조도에 의한 품질을 평가하기 때문에, 비전문가(모델 개발자)가 품질을 비교하기 쉽다. 그러나 포함범위, 환자자세, 흡기정도와 같은 품질 요소의 경우, 평가하기 위한 의료 지식이 필요하다. 예를 들어, 포함범위의 경우, 폐가 영상에 알맞게 포함되어 있는지를 판단하기 위해, 상하좌우의 늑골을 비교 분석할 수 있어야 한다[7]. 그러나 기존 연구에서는 세부적인 흉부 방사선 영상의 품질을 추적하여 평가하기 어려운 상황이다.

2.2 흉부 방사선 영상 품질 평가 방법

과거의 연구에서는 컴퓨터 비전 및 영상 처리 분야의 화질 평가 기술을 적용하여 의료 영상에도 화질(해상도 및 대조도)에 대한 영상 품질 평가를 하였다[8-9]. [8]의 연구에서는 구체적으로 인간 시각 시스템(HVS: Human Visual System)을 적용하여 색조 채도 등의 요소로 품질을 평가하였고, [9]의 연구에서는 객관적인 화질 평가 기법(Full Reference, Reduced Reference, No Reference)을 검토하여 노이즈, 해상도 등의 의료 이미지의 특성을 고려한 품질 평가 방법을 제안하였다. 하지만 [8-9]의 연구에서는 폐 영상 품질 평가 가이드라인에서 제시하고 있는 세부 품질 평가 요소(포함범위, 환자자세, 흡기정도 등)에 대해 평가를 할 수 없는 한계가 있다. 이에 비해 [10-11]의 연구는 포함범위, 환자자세, 흡기정도, 투과상태 등에 대한 품질 평가 방법을 제시하였다. [10]의 연구에서는 흉부 방사선 영상 촬영 시 발생하는 방사선량에 따라 척추를 감지하여 투과상태(penetration)에 품질 결정을 연구하였다. 하지만 해당 연구는 흉부 방사선 영상의 해상도, 대조도에 치중된 품질 결정으로, 해부학적

근거와 같은 객관적인 근거가 아닌 영상 전체에 대한 주관적인 판단에 의해 품질이 평가된다. 또한, [11]의 연구에서는 세부 품질 평가요소를 고려하여 대해 품질 적합여부(적절/부적절)를 레이블링(Labeling)하고, 흉부 방사선 영상 품질을 학습하는 방법을 제안하였다. [11]의 연구는 대표적인 CNN(ResNet, DenseNet) 모델을 사용하여 흉부 방사선 영상의 품질 평가를 자동화하였지만, 단순히 품질의 적합 여부만을 파악하고 있어 시각적인 평가가 필요한 흉부 방사선 영상의 구조물(예: 갈비뼈, 횡격막 등)의 세부적인 품질 요인(예: 흡기 수준 등)은 평가할 수 없다. 따라서 본 논문에서는 해부학적 구조물(폐 영역, 흉곽) 추출 알고리즘(폐 영역, 흉곽)을 설계하여, 세부 품질 요소에 대한 정밀한 품질 평가를 지원한다.

따라서 본 논문에서는 흉부 방사선 화질 평가 기준에 따라 오픈데이터셋의 품질 평가기법을 제안하고 고 품질의 데이터를 이용하여 학습성능을 개선시키고 낮은 품질 영상의 경우 품질 저하의 원인을 설명할 수 있는 체계를 수립한다.

3. 흉부 방사선 영상 품질 평가 기법 설계

본 논문에서는 흉부 방사선 영상의 품질 평가 요소 중 높은 수준의 의료 지식을 요구하는 포함범위, 환자자세, 흡기정도에 대한 세부 평가 방법을 설계한다. 포함되지 않은 인공음영이나 해상도는 의료 지식 없이도 평가 가능하다고 보고 제외하였다. 앞서 언급된 3가지 품질 요소에 대한 세부 설명은 표 2와 같으며, 세부 품질 평가 방법은 다음과 같이 이어진다.

표 2 품질 평가 핵심 요소 및 세부 고려사항 정의

항목	내용	세부 고려사항
포함범위	폐가 알맞게 포함되어 있는가?	폐 영역의 x, y 축 길이
환자자세	환자의 정렬이 잘되어 있는가?	폐 영역의 회전
흡기정도	적절한 흡기가 되어 있는가?	횡격막 기준 9번과 10번 갈비뼈의 위치

3.1 포함범위: 폐 영역 추출 기반 폐 포함 비율 분석

‘포함범위’는 흉부 방사선 영상의 상단, 하단, 좌우를 기준으로 늑골이 감싸는 부분인 폐 영역이 알맞게 포함되는지에 대해 평가한다. 달리 말하면, 폐 영역의 경우 너무 많은 범위를 가져서도 안 되며, 적은 범위를 가져도 안 되는 범위의 흉부 방사선 영상 데이터를 높은 품질로 평가하게 된다. 추출된 폐 영역에 대해 상(High), 하(Low), 좌(Left), 우(Right)에 대하여 정의된 비율(임계값)에 비교하여 품질 수준이 평가되며, 이는 그림 3에 표현된다.

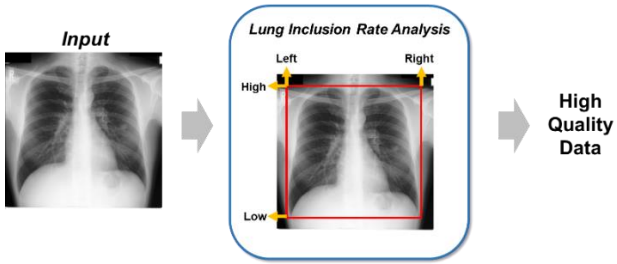


그림 3 폐 영역 추출 기반 폐 포함 비율 분석 방법

폐 영역의 추출은 HOG(Histogram of Oriented Gradients) 기반의 객체 인식 알고리즘을 사용한다[12]. HOG는 픽셀 변화량의 각도와 크기를 고려하여 히스토그램 형태로 특징을 추출하는 방법이다. 흉부 방사선 영상에서는 사람의 뼈 구조에 의해 밝기 값의 차이가 발생한다. 예를 들어, 폐 영역의 내부의 밝기는 상대적으로 낮고, 외부의 경계선은 밝다. 따라서 본 논문에서는 그림 4와 같이 폐 영상에서 픽셀 변화량의 각도와 크기에 대해 히스토그램 특징(HOG Feature)을 얻어 폐 영역을 추출한다. 먼저, 흉부 방사선 영상의 폐 영역을 자른 이미지(Setup Data)에 일정 영역 단위의 픽셀 변화량 크기와 각도를 추출한다. 그리고 폐 영역을 추출하고자 하는 흉부 방사선 영상 데이터셋(Test Data)에 동일한 크기만큼 적용하여 셋업 데이터(Setup Data)에서 정의된 HOG 특징과 가장 유사한 영역(Best HOG Feature)을 찾는다.

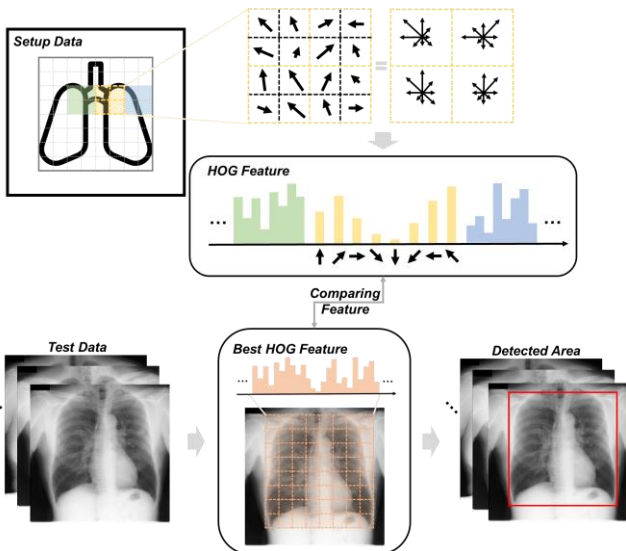


그림 4 HOG 알고리즘 적용을 통한 폐 영역 특징 추출

다음은 검출된 폐 영역을 기반으로, 높은 품질을 평가하기 위한 기준을 설정한다. 본 논문에서는 높은 품질의 데이터가 구축된 JSRT 데이터셋[5]의 폐 영역 데이터를 기반으로, 통계적으로 높은 품질의 기준을 도출한다. 도출에 사용하는 데이터는 검출된 폐 영역 영상의 X축, Y축의 길이이며, JSRT 데이터셋의 X축, Y축 길이의 사분위수를 기준으로 품질을 분류한다.

구체적으로 다음 수식 (1), (2)와 같이 사분범위(IQR, Interquartile Range) 안에 드는 것을 높은 품질에 대한 기준으로 설정하였다.

$$Q_{1,JSRT}(X_{Length}) \leq X_{Length} \leq Q_{3,JSRT}(X_{Length}) \quad (1)$$

$$Q_{1,JSRT}(Y_{Length}) \leq Y_{Length} \leq Q_{3,JSRT}(Y_{Length}) \quad (2)$$

3.2 환자자세: 폐 영역 추출 기반 회전 수준 분석

‘환자 자세’는 앞의 표 1에서 기술된 바와 같이, 견갑골의 올바른 위치와 좌우 대칭의 정도로 평가한다. 견갑골의 경우 폐야 밖에 위치해야 하며, 좌우 대칭은 척추와 쇄골 기준의 좌우 대칭을 확인한다. 따라서 ‘환자자세’ 역시, ‘포함범위’ 품질 요소와 동일하게 폐 영역 추출 결과를 기반으로 한다. ‘환자자세’ 평가에서는 환자의 정렬 상태를 확인해야 하므로, 흉부 방사선 영상에서 환자의 척추, 쇄골 등이 기울어져 폐 영역이 기울어진 정도에 따라 평가한다. 이를 위해 흉부 방사선 영상 품질 요소인 포함범위에서 적용한 객체 인식 알고리즘(HOG)을 적용하여 그림 5와 같이 제안한다. HOG 알고리즘을 통한 객체 인식 방식은 특정 영역 단위의 기하학적 정보를 유지하며 히스토그램 사용으로 지역적인 변화(Locality)에 강인한 속성을 가져 크고 작은 폐 영역을 효율적으로 추출하지만 일종의 템플릿 매칭(HOG 특징을 통한 매칭)으로 회전된 경우에 검출이 힘들다는 단점이 존재한다. 따라서 본 논문에서는 그림 5와 같이 흉부 방사선 영상이 폐 영역 추출을 할 때, 실패한 경우를 낮은 품질의 데이터로 분류하며, 성공하는 경우 환자자세 요소에는 높은 품질 데이터로 분류한다.

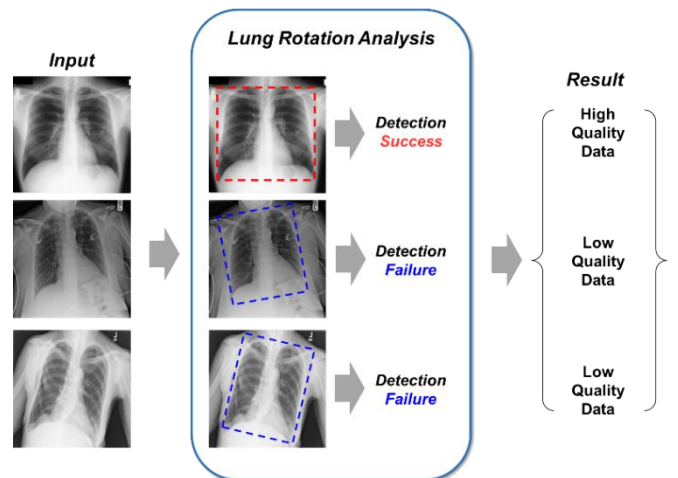


그림 5 폐 영역 추출 기반 환자자세 평가 분석 예시

3.3 흡기정도: 흉곽 검출 기반 흡기 수준 분석

‘흡기정도’는 정상 흡기 상태 유무에 따라 평가되며, 다음과 같은 세부 기준안을 갖는다[7].

- 쇄골 중심부에서 수직선을 그어 횡격막과 만나는 점이 후방 10번 갈비뼈보다 낮게 위치하는가?
- 전방 6번 갈비뼈와 후방 10번 갈비뼈가 오른쪽 폐 위에 보이는가?
- 후방 9~10번 갈비뼈가 횡격막 위로 존재하는가?
- 적어도 8번 갈비뼈가 횡격막 위로 보이는가?

세부 기준안에서 확인할 수 있듯이 정상 흡기 상태는 흉부 방사선 영상에서 보이는 횡격막과 갈비뼈를 기준으로 확인한다 따라서 본 논문에서는 그림 6과 같이 흡기 수준 평가를 위해 횡격막과 갈비뼈를 검출하여 품질을 평가하기 위한 기준을 정립한다. 먼저, 횡격막을 관찰을 위해, 영상 이진화(Binarization)를 사용한다. 또한, 갈비뼈를 추출하기 위해 학습 모델(U-Net) 기반의 영상 분할(Segmentation) 방법을 적용하여 흉곽을 검출한다. 마지막으로, 영상 이진화를 통해 얻은 영상과 흉곽 검출을 통해 9-10번 갈비뼈를 추출한 영상을 통합하여 흡기 수준을 분석(Inspiration Analysis)를 수행한다.

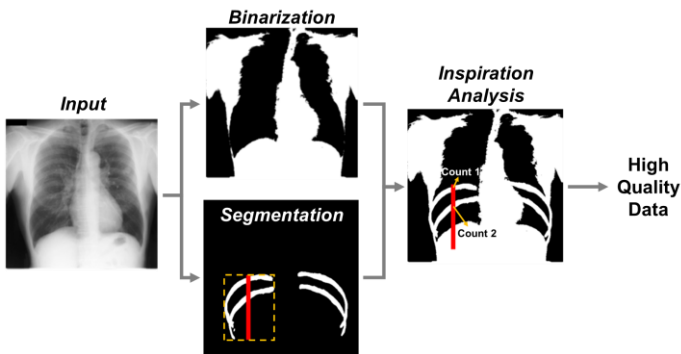


그림 6 흉곽 검출 기반 흡기 수준 분석 방법

① 영상 이진화(Binarization)

횡격막 경계선을 추출하기 위해 영상을 이진화를 사용한다. 영상 이진화는 임계값을 기준으로 영상에서 밝은 픽셀은 모두 흰색, 그렇지 않은 픽셀은 모두 검은색으로 바꾸는 것을 지칭한다. 흉부 방사선 영상은 회색조(Gray Scale)로 표현되는 특징이 있지만, 촬영 파라미터 및 장비에 따라 영상의 밝기에 대한 차이가 크다. 따라서 본 논문에서는 영상의 임계값을 자동으로 결정해주는 Otsu 알고리즘[13]을 적용하였다. 해당 알고리즘은 임계값을 기준으로 영상 픽셀들을 두 클래스(검은색, 흰색)로 분류하였을 때, 각 클래스 내부의 분산(Intra-class Variance)를 최소화하는 것과 두 클래스 간의 분산(Inter-class Variance)를 최대화하는 임계값을 찾는 이진화 방법이다. 특정 임계값에 관하여 영상 이진화를 통해 검은색 클래스의 개수(A), 흰색 클래스의 개수(B)에 대한 내부 및 외부 분산은 각 클래스의 평균(μ)와 표준편차(σ)에 의해 계산되며 Otsu의 임계값은 0-255까지 단계적으로

변화시키면서 최적의 값으로 자동 결정해준다.

$$\alpha = \frac{A}{A+B}, \quad \beta = \frac{B}{A+B}, \quad \alpha + \beta = 1$$

$$intra\ class\ variance = \alpha\sigma_A^2 + \beta\sigma_B^2$$

$$inter\ class\ variance = \alpha\beta(\mu_A - \mu_B)^2$$

② U-Net 기반의 흉곽 검출 분할(Segmentation) 모델

흉부 방사선 영상의 정상 흡기 상태를 확인하기 위해 U-Net[14] 기반의 검출 모델을 사용한다. U-Net은 바이오 의학 분야에서 영상 분할(Segmentation)을 목적으로 제안된 End-to-End 방식의 학습 모델이다. 기본적인 구조는 입력 영상(Input)에 대해 다운 샘플링(Down-Sampling)하는 합성곱층(Convolution Layer), 영상의 문맥(Context) 포착을 하는 수축 경로(Contracting Path), 포착된 문맥을 업 샘플링(Up-Sampling)하는 합성곱층, 높은 해상도의 분할(Segmentation) 결과를 얻는 확장 경로(Expanding Path)로 구성되어 있으며, 그림 7과 같이 설명할 수 있다.

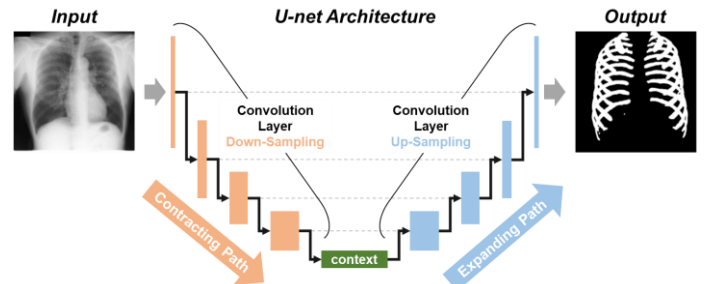


그림 7 U-Net 기반 흉곽 검출 모델 구조

U-Net의 학습은 1~10번 갈비뼈가 레이블링되어 있는 VinDr-RibCXR[15] 데이터셋을 사용하였다. 또한, 본 논문에서는 흉부 방사선 영상의 품질 요소 중 흡기 수준을 분석하기 위해 9-10번 갈비뼈만을 추출하여 품질 평가에 사용하였다.

③ 흡기 수준 분석 방법(Inspiration Analysis)

그림 8은 ①과 ②로부터 추출된 영상을 병합하여, 흡기 수준을 분석하는 방법을 나타낸다. 먼저, 흉곽 검출 모델을 통해 검출된 9-10번 갈비뼈에 대해, 그림 8(a)와 같이 Left, Right의 중간 위치(빨간색 선분)를 추출한다. 그리고 추출한 선분을 기준으로, 그림 8(b)와 같이 9-10번 갈비뼈와 이진화 영상에 적용되어 갈비뼈를 인식한다.인식하는 방법은 그림 8(c)와 같은 방식으로 추출한 선분 영역의 밝기를 분석하는 것이다. 이 때, 밝기 피크(Peak) 값의 빈도수로 갈비뼈의 개수를 추출할 수 있으며, 횡격막 구간($x > 380$ 구간)은 이진화로 인해 밝기가 255로 유지되기 때문에 피크로 추출되지 않는다. 최종적으로 9-10번 갈비뼈의

검출된다면, 해당 데이터를 흡기 수준이 충분한 높은 품질로 평가하게 된다.

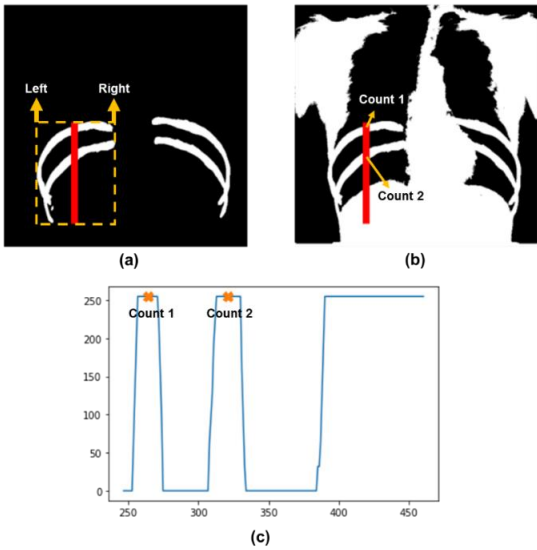


그림 8 흉부 방사선 영상 흡기정도 품질 평가 방법

4. 오픈 데이터셋 품질 평가 기반 학습 성능 비교 실험

4장에서는 3장에서 설계된 품질 평가 방법을 대규모 오픈 데이터셋(Chest X-ray14)에 적용하여 제안하는 방법의 효용성을 입증한다. 또한, 오픈 데이터 셋을 이용하여 폐 결절(Nodules)을 진단하기 위한 모델 개발에 적용하여 실험하고, 성능을 비교하여 품질의 중요성을 확인한다.

4.1 흉부 방사선 영상 오픈 데이터셋 수집

흉부 방사선 영상의 오픈 데이터셋은 JSRT[5], Chest X-ray14[6]를 수집하였다. JSRT 데이터셋은 하나 이하의 폐 결절에 대한 흉부 방사선 영상으로 총 247개로 폐 결절이 있는 영상 154개, 없는 영상 93개로 이뤄져 있다. JSRT 데이터셋을 구축하기 위해, 평균 경력 12.1년인 20명의 방사선 전문의가 참가하여 데이터에 대한 검증을 수행하였고, 폐 결절 검출에 적합한 매개변수로 흉부 방사선 영상을 촬영하였기 때문에 높은 품질의 데이터셋으로 평가되고 있다. 본 논문에서는 해당 데이터셋을 제안하는 품질 평가 기법에서 높은 품질과 낮은 품질을 나누는 기준(임계값)을 도출하기 위해 사용하였다.

Chest X-ray14 데이터셋은 의료 영상 저장 전송 시스템(PACS, Picture Archiving and Communication Systems)에서 자연어 처리 기술(NLP, Natural Language Processing)를 통한 텍스트 마이닝을 통해서 구축하여 많은 양의 데이터 수(정면 영상만 112,120개)를 확보하였지만, 비전문가가 보기에 낮은 품질의 영상이 많이 포함되어 있는 것을 볼 수 있다. 따라서 본 논문에서는 Chest X-ray14 데이터셋을

제안하는 흉부 방사선 영상의 품질 평가 기법을 통해 높은 품질, 낮은 품질의 데이터셋으로 구분하고, 질병 진단 모델의 학습에 사용하여 데이터의 품질이 성능에 끼치는 영향을 확인한다. 본 논문은 112,120개의 Chest X-ray14 데이터셋에서 폐 결절이 있는 영상 6631개, 병변이 없는 정상 영상 5804개를 수집하였다.

U-Net[14] 기반의 흉곽 검출 모델 학습을 위한 개별 갈비뼈가 레이블링된 VinDr-RibCXR 데이터셋[15]을 총 245개 수집하였으며, 오픈 데이터셋에 대한 정리는 표 3와 같이 요약된다.

표 3 오픈 데이터셋 수집 결과 정리

명칭	수집 기관	개수	용도
JSRT [5]	Japanese Society of Radiological Technology	247	높은 품질 데이터 기준 도출
Chest X-ray14 [6]	National Institutes of Health Center	12,435	품질 분류를 통한 모델 성능 실험
VinDr-RibCXR [15]	Vingroup Big Data Institute	245	흉곽 검출 모델의 학습데이터

4.2 흉부 방사선 영상 품질 평가 절차 구현 및 실험

4장에서는 3장에서 설계한 세부 품질 요소 평가 방법을 기반으로, 그림 9와 같은 흉부 방사선 영상의 통합 품질 평가 절차를 정립하였다. 해당 절차는 폐 영역 추출, 흉곽 검출의 기법을 적용하여 포함범위 환자자세, 흡기정도 3가지의 품질 요소를 평가하는 방식이며, 세부적인 평가 결과는 다음과 같이 이어진다.

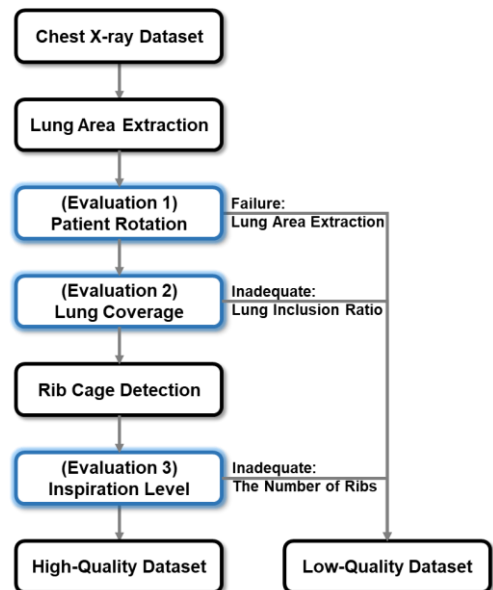


그림 9 흉부 방사선 영상 품질 평가 절차 구현

① 폐 영역 추출 기반 환자자세 및 포함범위 평가

흉부 방사선 영상 데이터셋이 입력으로 HOG 기반 객체 인식 알고리즘을 통해 폐 영역 추출하여 환자자세와 포함범위를 평가한다. JSRT[5], Chest X-ray14 데이터셋에 대해 환자자세와 포함범위를 평가한 결과는 그림 10과 같으며, 세부적인 설명은 다음과 같다.

- 그림 10(a): 포함범위 품질이 좋은 예시(JSRT)로, 폐 영역(빨간 박스) 부분이 상하좌우의 길이가 일관되고 알맞게 형성(예: 바깥으로 2~3cm)되는 것을 볼 수 있다.
- 그림 10(b), (c): 폐 영역이 미흡 및 초과하게 형성되어 상자의 길이(X-axis Length, Y-axis Length)가 높은 품질 영상보다 다양한 크기로 형성되는 것이 특징이다.
- 그림 10(d), (e): 폐 영역이 인식되지 않았으며 점선으로 추출 되어야 하는 폐 영역을 표시하였다. 폐 영역이 회전이 되는 경우 HOG 기반의 객체 인식 알고리즘이 구조적인 특징인 쇄골, 심장, 갈비뼈, 횡격막, 척추 등의 특징을 인지 못하여 폐 영역이 인식되지 않는다.

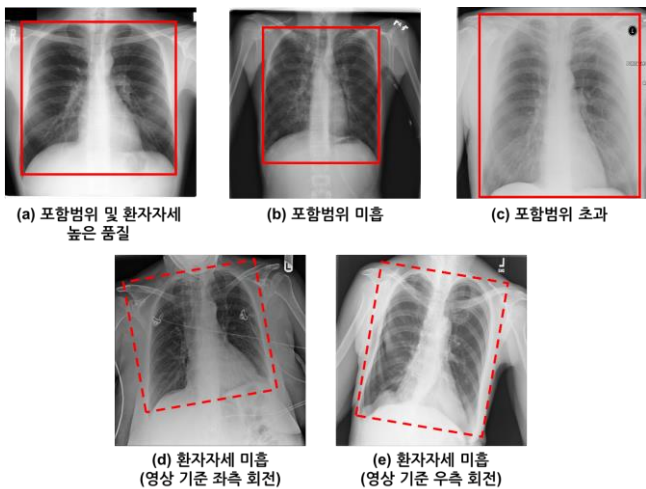


그림 10 폐 영역 추출 기반 품질 평가 방법 적용 결과 예시

이때, 포함범위에 대하여 폐 영역이 다양한 크기로 형성되는 것을 고려하여, 포함범위가 미흡 및 초과되는 것에 대한 기준을 정의하기 위해 높은 품질의 영상의 기준을 도출하였다. 표 4와 같이 JSRT 데이터셋을 이용하여 기준을 정립하였으며, 폐 영역의 상자의 길이(X-axis Length, Y-axis Length)를 데이터를 4등분으로 나누는 사분위수 값을 사용하였다.

② 흉곽 검출 기반 흡기정도 평가

본 논문에서는 U-Net의 기본 구조를 기반으로, 18개의 3x3 합성곱층(Convolution Layer), 4개의 최대

표 4 JSRT 데이터셋 포함범위 평가 결과

항목	고려 사항	제1사분위수 (Q1: 25%)	제2사분위수 (Q2: 50%)	제3사분위수 (Q3: 75%)
포함 범위	X축 길이	409	428	447
	Y축 길이	367	385	404

*흉부 방사선 영상 512x512 픽셀 기준

풀링층(Max Pooling), 4개의 2x2 업 합성곱층(Up-convolution Layer) 구성된 수축 경로(Contracting Path)와 확장 경로(Expanding Path)를 구성하였다(이때, ImageNet으로 사전 학습된 모델을 사용하였다). 또한, 1-10번 갈비뼈에 대한 레이블링 작업된 VinDr-RibCXR 데이터셋[15]을 학습 데이터로 사용하였으며, VinDr-RibCXR 데이터셋 245개 중 196개의 영상을 학습으로 사용하고, 49개의 영상을 검증 데이터셋으로 사용하였다. 학습 결과, 갈비뼈 추출에 대한 성능 지표가 F1 점수 0.823, 민감도 0.833, 특이도 0.997인 결과를 얻었다.

표 2의 흡기정도 항목의 세부 고려사항은 횡격막 기준 9-10번 갈비뼈의 위치로, 학습한 흉곽 검출 모델과 영상 이진화 알고리즘을 JSRT 데이터셋에 적용하여 3.3절의 흡기 수준 분석에 적용하였다. JSRT[4] 데이터셋의 모든 흉부 방사선 영상 247개에 대하여 제안하는 흡기정도 품질 평가에 대한 결과, 9-10번째 갈비뼈를 검출하였을 때, 3개의 영상을 제외하고 모두 2개의 갈비뼈가 인식되었다. 이때, 1-2개의 갈비뼈 인식의 경우 높은 품질이며, 인식한 갈비뼈가 없는 경우 낮은 품질로 평가될 수 있다.

4.3 품질에 따른 오픈 데이터셋 구성 결과

흉부 방사선 영상으로 구성된 Chest X-ray14[6] 데이터셋(폐 결절을 가진 영상 6,631개, 병변 없는 정상 영상 5,804개)에 대해, 총 12,135개 영상을 4.2절의 흉부 방사선 영상 품질 평가 절차를 적용하였다. 품질 평가 결과, 폐 결절을 가진 영상에 대한 품질 평가 결과는 표 5와 같으며, 병변 없는 정상 영상에 대한 품질 평가 결과는 표 6과 같다. 결과적으로, 흉부 방사선 영상 품질 평가 절차를 통해 높은 품질 데이터셋은 폐 결절 영상 2,852개, 병변 없는 정상 영상 2,729개로 총 5,581개(전체의 영상의 46%)의 데이터셋을 구성할 수 있었으며, 낮은 품질 데이터셋은 폐 결절 영상 3,479개, 정상 영상 3,075개로 총 6,554개(전체 영상의 54%)의 데이터셋을 구성하였다. 또한, 구성된 높은 품질과 낮은 품질 영상을 5:5 비율로 샘플링하여, 중간 품질 데이터셋을 구성할 수 있으며, 각 품질에 따른 모델 성능을 확인하였다.

표 5 Chest X-ray14 폐 결절 영상 품질 평가 결과

절차	입력 개수	높은 품질 분류 개수	낮은 품질 분류 개수
폐 영역 추출	6,331	-	-
환자자세 평가	6,331	6,020	311
포함범위 평가	6,020	3,434	2,586
흉곽 검출	3,434	-	-
흡기정도 평가	3,434	2852	582
결과	-	2,852 (45%)	3,479 (55%)

표 6 Chest X-ray14 정상 영상 품질 평가 결과

절차	입력 개수	높은 품질 분류 개수	낮은 품질 분류 개수
폐 영역 추출	5,804	-	-
환자자세 평가	5,804	5,642	262
포함범위 평가	5,642	3,240	2,402
흉곽 검출	3,240	-	-
흡기정도 평가	3,240	2,729	511
결과	-	2,729 (47%)	3,075 (53%)

4.4 품질에 따른 진단 모델 성능 분석

학습 데이터의 품질에 따른 진단 모델의 성능 분석을 위해 본 논문에서는 대표적인 CNN(Convolution Neural Network) 모델을 사용하였다. 이때, 폐 결절 질병 진단에 대한 학습을 8:2의 훈련데이터와 검증데이터로 나눠 수행하였으며, 규모가 작은 18층의 합성곱층(Convolution Layer)로 구성된 ResNet-18 모델과 161층의 합성곱층으로 이루어진 큰 규모의 DenseNet-161에 대해 실험을 하였다. 또한, 학습 데이터의 수가 적을 때 효과적으로 학습 속도와 정확도를 높일 수 있는 방식인 사전학습 영향에 대한 실험을 수행하였다. 실험 결과, ImageNet을 통한 사전학습 유무에 따라 실험하여 표 7과 표8과 같이, AUC(Area Under Curve) 점수와 정확도(Accuracy)를 지표로 평가하였다.

먼저, 사전 학습된 모델에서 낮은 품질 데이터셋(L)의 진단 성능보다 높은 품질 데이터셋(H)로 학습한 모델의 성능이 AUC의 경우 약 1.9~2.2%, 정확도의 경우 약 3.2~3.8%의 성능 향상을 볼 수 있었으며, 두 데이터셋(L, H)을 5:5로 샘플링하여 구성된 중간 품질 데이터셋(M)의 학습 성능은 두 데이터셋(L, H) 학습 성능의 중간에 위치하는 것을 확인할 수 있다. 또한 규모가 큰 모델(DenseNet-161)이 더 높은 성능 향상을 볼 수 있었다. 그리고 사전학습을 수행하지 않은 모델에 대한 성능에서도 동일한 양상을 확인할 수 있다. 하지만

AUC의 경우 약 2.7~4.5%, 정확도의 경우 2.9~4.6% 정도 성능이 개선되었으며, 사전학습 모델보다 더 큰 폭으로 성능이 개선되는 것을 확인할 수 있다.

표 7 사전학습 O: 품질에 따른 진단 모델 성능 결과

		사전학습 O	
		ResNet-18	DenseNet-161
L	AUC	0.9250	0.9284
	Accuracy	0.8513	0.8543
M	AUC	0.9335	0.9340
	Accuracy	0.8584	0.8691
H	AUC	0.9442	0.9500
	Accuracy	0.8836	0.8926

표 8 사전학습 X: 품질에 따른 진단 모델 성능 결과

		사전학습 X	
		ResNet-18	DenseNet-161
L	AUC	0.8547	0.8610
	Accuracy	0.7780	0.7857
M	AUC	0.8787	0.8983
	Accuracy	0.7967	0.8140
H	AUC	0.8811	0.9059
	Accuracy	0.8075	0.8317

5. 결론

본 논문에서는 흉부 방사선 영상의 질병 진단 모델 성능 개선을 위한 영상 데이터의 통합 품질 평가 방법을 제안하였다. 특히, 흉부 방사선 영상 오픈 데이터셋(Chest X-ray14)을 제안하는 방법에 적용하여, 폐 결절에 대한 질병 진단 모델의 성능이 높은 품질의 학습 데이터에 의해 개선되는 것을 확인하였다. 구체적으로는 총 12,135개의 폐 결절 데이터셋에서 높은 품질 데이터 5,581개, 낮은 품질 데이터 6,554개의 데이터셋을 구성한 것에 대해 높은 품질 데이터셋이 진단 모델에 따라 AUC 약 1.9~4.5%, 정확도 약 2.9~4.6%의 성능이 향상되는 것을 확인하였다. 이처럼, 제안하는 방법은 단순히 품질 적합 여부(적절/부적절)만을 판단하는 것이 아니라, 가이드라인에 제시된 세부 품질 평가 요소에 대한 추적성을 확보하고, 정밀한 품질 평가를 수행할 수 있도록 하여 학습 모델의 성능 개선을 지원한다.

참고 문헌

[1] CHASSAGNON, Guillaume, et al. Artificial intelligence applications for thoracic imaging. European journal of radiology, 123, 108774, 2020
 [2] YOU, Seulgi, et al. Visual assessment of calcification in solitary pulmonary nodules on

- chest radiography: correlation with volumetric quantification of calcification. *European radiology*, 29.8: 4324–4332, 2019
- [3] QIN, Chunli, et al. Computer-aided detection in chest radiography based on artificial intelligence: a survey. *Biomedical engineering online*, 17.1, 1–23, 2018
- [4] RANASINGHE, Suneth, PICHLER, Horst, EDER, Johann. Report on Data Quality in Biobanks: Problems, Issues, State-of-the-Art. arXiv preprint arXiv, 1812.10423, 2018.
- [5] SHIRAIISHI, Junji, et al. Development of a digital image database for chest radiographs with and without a lung nodule: receiver operating characteristic analysis of radiologists' detection of pulmonary nodules. *American Journal of Roentgenology*, 174.1, 71–74, 2000
- [6] WANG, Xiaosong, et al. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 2097–2106, 2017
- [7] J.H.E. Carmichael, C. Maccia, B.M. Moores, J.W. Oestmann, H. Schibilla, D. Teunen, et al. Commission of the European Communities European Guidelines On Quality Criteria For Diagnostic Radiographic Images Office of Official Publications of the European Communities, Luxembourg, 1996
- [8] CHOW, Li Sze; PARAMESRAN, Raveendran. Review of medical image quality assessment. *Biomedical signal processing and control*, 27: 145–154, 2016
- [9] WEI, Wu. An image quality assessment method based on HVS. In: *2007 41st Annual IEEE International Carnahan Conference on Security Technology*. IEEE, p. 320–324, 2008
- [10] DOVGANICH, A. A., et al. Automatic Quality Control in Lung X-Ray Imaging with Deep Learning. *Computational Mathematics and Modeling*, 1–10, 2021
- [11] NOUSIAINEN, Katri, et al. Automating chest radiograph imaging quality control. *Physica Medica*, 83: 138–145, 2021
- [12] DALAL, Navneet; TRIGGS, Bill. Histograms of oriented gradients for human detection. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. IEEE, p. 886–893, 2005
- [13] OTSU, Nobuyuki. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9.1: 62–66, 1979
- [14] RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-net: Convolutional networks for biomedical image segmentation. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, p. 234–241, 2015
- [15] NGUYEN, Hoang Canh, et al. VinDr-RibCXR: A Benchmark Dataset for Automatic Segmentation and Labeling of Individual Ribs on Chest X-rays. 2021

오픈 소스 기계학습 어플리케이션에 대한 결함 사례 조사

최윤호^{o+}, 이창공*, 방제형*, 남재창^{h*}

한동대학교 ⁺정보통신공학과, *전산전자공학부
{yhchoi, 21701042, 21700328, jcnam}@handong.edu

An Empirical Study on Defects in Open Source Artificial Intelligence Applications

Yoon-ho Choi^o, Chang-gong Lee, Jehyeong Bang, Jaechang Nam
Department of Information Technology, Handong Global University

요 약

기계학습 기법을 기반으로 한 프로그래밍 패러다임은 전통적인 방식의 프로그래밍 패러다임과 큰 차이가 있다. 이와 같은 프로그래밍 패러다임의 차이는 전통적인 방식의 어플리케이션의 결함을 검출하고 이해, 분석, 해결하는 것에도 서로 다른 양상을 나타낼 수 있다. 이와 같은 상황에서, 본 연구는 기계학습 기반 시스템이 가진 결함을 이해하고 분석하기 위해, 오픈소스 기계학습 어플리케이션에서 발생했던 결함의 사례들을 수집하고 기계학습 기반 시스템에서 빈번하게 발생하는 결함의 원인을 파악하고자 하였다. 이를 위해, GitHub에 공개된 10개의 오픈소스 기계학습 어플리케이션을 대상으로 GitHub 이슈 게시판에 있는 내용과 결함 해결 코드 수정 이력을 직접 분석하여 보고/발견/수정되었던 결함에 대해 분석하였다. 조사 대상으로 선정한 10개의 어플리케이션에서 발견한 1,205개의 결함 이슈 보고와 결함 수정 코드를 직접 분석해 10개 중 5개 이상의 프로젝트에서 공통적으로 발견된 결함의 근본적인 원인 기준으로 범주화하였으며, 20개의 결함 원인 범주를 설정하였다. 본 연구의 결과는 결함 보고 분류, 결함 위치 추적, 가능한 결함 해결 코드 수정 제안, 테스트/디버깅 비용 측정, 개발 과정 관리 등에 유용하게 활용될 수 있다.

1. 서론

기계학습 기법이 등장한 이후, 여러 가지 문제에 대해 전통적으로 사용했던 알고리즘보다 뛰어난 성능을 보이며 기계학습 기반 어플리케이션이 여러 분야에 적용되고 있다. 기계학습 기법을 기반으로 한 프로그래밍 패러다임은 전통적인 방식의 프로그래밍 패러다임과 큰 차이가 있다. 이러한 차이는 기계학습 어플리케이션에서 발생 가능한 결함을 기존의 방식으로 검출하고 이해, 분석, 해결하는 것과 다른 양상을 나타낼 수 있다. 따라서, 기존의 프로그래밍 패러다임과 다른, 기계학습 모델이 가진 결함에 대한 분석과 이해가 요구된다.

본 연구에서는 기계학습 기반 어플리케이션이 가진 결함을 분석하기 위해, 오픈소스 기계학습 어플리케이션에서 발견 및 발생했던 결함의 사례들을 수집하고 기계학습 기반 어플리케이션에서 빈번하게 발생하는 결함의 원인을 파악하였다. 이를 위해, GitHub에 공개되어 있는 10개의 오픈소스 기계학습 어플리케이션을 대상으로 GitHub 이슈 게시판에 있는 내용과 결함 해결 코드 수정 이력을 직접 분석하여 보고/발견/수정되었던 결함에 대해 분석하였다.

조사 대상에서 발견한 1,205개의 결함 이슈 보고와 결함 수정 코드를 직접 분석해 10개 중 5개 이상의 프로젝트에서 공통적으로 발견된 결함의 근본적인 원인 기준으로 범주화하였으며, 20개의 결함 원인 범주를 설정하였다.

2. 연구 방법

사례 수집에 앞서, GitHub에 공개된 오픈 소스 프로젝트 중,

머신러닝 혹은 딥러닝 기술을 사용하였고 해당 레포지토리의 닫힌 이슈 (Closed-Issue)의 개수가 500개 이상인 프로젝트를 조사 대상으로 선정하였다.

조사 대상 기준에 적합한 기계학습 오픈소스 프로젝트를 찾기 위해, GitHub에서 제공하는 프로젝트 검색 태그에 Machine learning 또는 Deep learning 태그를 붙여놓은 프로젝트를 선별 후, 해당 프로젝트가 머신러닝 혹은 딥러닝 기술을 사용하는 프로젝트인지 프로젝트 설명 문서와 코드를 참고하여 2차 선별하였다. 또한, 닫힌 이슈의 개수가 500개 이상인지 확인하기 위해 2차 분류된 프로젝트에 GitHub API 버전 4를 NodeJS와 함께 사용해 닫힌 이슈 개수와 이슈 제목, 이슈 라벨, 이슈 내용, 이슈 답글, 커밋 아이디, 커밋 메시지를 마이닝 하였다.

최종적으로 조사 대상으로 선정한 프로젝트의 이름은 DeepSpeech, OpenPose, spaCy, Real-Time-Voice-Cloning, tesseract, yolov5, mediapipe, ChatterBot, phtoprism, streamlit 으로, 총 10개이다. 다양한 분야에 응용되고 있는 기계학습 코드를 분석하고, 일반성을 높이기 위해 응용 분야가 상이한 프로젝트를 선정했다.

그림 1은 본 연구에서 사용한 사례의 선정 및 분류 방식을 간략하게 도식화하여 보여준다. GitHub 이슈에서 결함이 보고/해결된 이슈를 찾아 분석함으로써 빈번하게 발생하는 결함의 원인에 대해 범주화하기 위해 총 두 단계의 필터링 과정을 거쳤으며, 필터링 이후 단계에서 남아있는 결함 이슈에 대해 일반적인 원인을 기반으로 분류하였다. 분류 단계에서는 결함과 관련된 이슈 내용과 해당 결함을 해결한 코드 수정을 직접 확인하였다.

세 단계 중 첫째로, 이슈 라벨을 활용해 결함과 관련이 있는 이슈와 그렇지 않은 이슈를 분류하고 결함과 무관한 라벨이 붙어

있는 이슈를 제거하였다. 첫 번째 단계에 이어서 두 번째 단계로, 이슈 내용을 활용해 개발자가 이슈로 제기된 결함에 내용에 동의하고 수정하였는지 확인해, 결함과 관련된 이슈 라벨이 붙어 있는 이슈가 실제로 결함과 관련된 이슈인지 분류하고 실제 결함이 아닌 경우는 제거하였다. 세 번째 단계로, 결함을 유발한 원인이 무엇인지 파악하고 어떻게 수정하였는지에 따라 결함의 원인을 분류하고 범주화하기 위해 결함 유발 코드를 직접 확인하고 분류하였다.

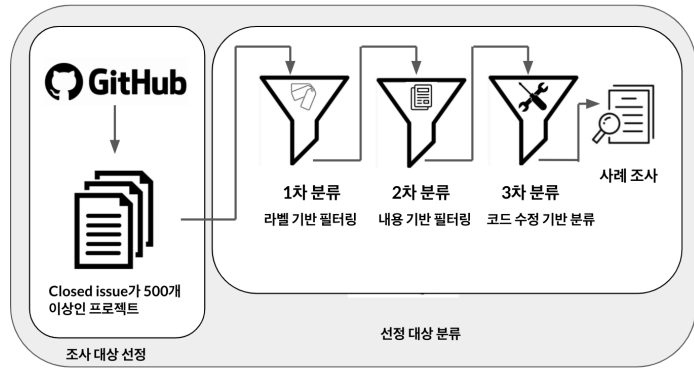


그림 1. 본 연구의 조사 대상 분류

선정한 10개의 오픈소스 프로젝트에 존재한 닫힌 이슈의 개수는 13,189개였으며, 그중 1차 분류 이후 2,447개를 선정하였다. 1차 분류를 통해 선정된 이슈 가운데 2차 분류를 진행한 결과 총 1,205개의 이슈가 선별되었으며, 이를 바탕으로 결함의 근본적인 원인 분석을 진행하였다.

3. 연구 결과

조사 대상으로 선정한 10개의 오픈 소스 프로젝트 중 5개 이상의 프로젝트에서 발견 가능한 이슈를 대표적 결함 원인으로 선정하였다. 표 3은 2.2.에서 명시한 세 단계에 걸쳐 수집된 대표적인 결함의 원인 목록이다. 표3의 빈도 열은 1,205개의 이슈 중 각 원인이 발견된 이슈의 개수를 의미한다. 순번은 빈도를 기준으로 내림차순으로 정렬한 결과이다.

표 1. 결함 유발 코드 사례 목록

순번	결함 유발 원인	빈도	프로젝트 개수
1	예외적인 데이터에 대한 처리 누락	214	8
2	특정 하드웨어/ 장치/운영 체제에 대한 잘못된 구현	84	8
3	데이터 처리 과정 중 이전 데이터 누락	81	8
4	필요한 데이터 속성 (attributes) 누락	60	7
5	오타	52	9
6	잘못된 오류 핸들링	45	7
7	처리에 알맞지 않은 데이터/데이터 타입 /연산자 사용	44	6
8	모듈 간의 일관성 유지 실패	39	7

9	기본값/옵션값의 잘못된 설정 및 누락	35	9
10	문서/매뉴얼 오류	35	8
11	데이터의 잘못된 비교, 정렬, 배치	32	6
12	외부 라이브러리 의존성 문제	32	8
13	사용자 옵션/설정에 대한 업데이트 불가	28	7
14	예외적인 실행 환경에 대한 처리 누락	28	6
15	라이브러리 임포트/설치 문제	27	9
16	잘못 저장된 모델/캐싱 혹은 잘못된 출력값	15	5
17	데이터 처리를 위한 경로/웹 주소의 잘못된 설정	14	7
18	메모리 누수/ 잘못된 메모리 관리	12	5
19	잘못된 병렬화 구현	9	6
20	잘못된 인코딩/디코딩	9	5

4. 결론 및 향후 연구

본 연구에서는, 기계학습 기법을 사용하는 다양한 오픈소스 어플리케이션에서 발견한 결함을 기반으로 기계학습 기반 어플리케이션에서 나타날 수 있는 결함의 원인과 증상을 분석하였다. 결함을 분석하기 위해 GitHub 어플리케이션 10개에 보고된 1,205개의 결함 보고를 직접 분석하였으며, 연구 결과 20개의 결함 원인을 정리하였다. 분석한 결함을 자동으로 수정할 수 있는 방법을 고안하는 방향으로 연구를 확장할 계획이다.

※ 본 연구는 한국과학기술정보연구원(KISTI) ‘Data/AI 기반 문제해결 체계 구축(K-22-L04-C04)’ 사업과 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원(No.2021R1F1A1063049)으로 수행된 연구임.

참고 문헌

[1] Li, Z., Tan, L., Wang, X., Lu, S., Zhou, Y., & Zhai, C. (2006). Have things changed now? An empirical study of bug characteristics in modern open source software. In *Proceedings of the 1st workshop on Architectural and system support for improving software dependability* (pp. 25-33).
 [2] Lu, S., Park, S., Seo, E., & Zhou, Y. (2008). Learning from mistakes: a comprehensive study on real world concurrency bug characteristics. In *Proceedings of the 13th international conference on Architectural support for programming languages and operating systems* (pp. 329-339).
 [3] Pan, K., Kim, S., & Whitehead, E. J. (2009). Toward an understanding of bug fix patterns. *Empirical Software Engineering*, 14(3), 286-315.

효율적인 아키텍처 패턴 적용을 위한 소프트웨어 요구사항 분류체계에 관한 연구

최종우^o 민상윤

한국과학기술원

jongwoo.choi@kaist.ac.kr, symin@kaist.ac.kr

A Study on the Taxonomy of Software Requirements for Efficient Application of Architectural Patterns

Jong-Woo Choi^o Sang-Yoon Min

Korea Advanced Institute of Science and Technology

요약

소프트웨어 아키텍처 패턴은 경험이 많은 아키텍트들의 지식을 집약한 솔루션이다. 아키텍처 설계 방법론이 많이 연구되고 있지만, 시스템에 여러 개의 패턴을 적용하는 세부적인 방법이나 적용 순서에 따른 효과성에 대해서는 다루고 있지 않다. 본 연구에서는 먼저, 같은 세트(Set)의 패턴들을 동일 시스템에 적용할 때, 적용 순서의 차이에 따른 아키텍처 설계 결과의 상이성에 대해 설명한다. 이러한 현상적 논리에 기반하여 패턴 적용이 필요한 요구사항들을 사전에 식별하고, 적용할 패턴을 분류하는 방안을 제시하고 분류한 패턴들의 효율적 적용을 위해 적용 순서를 정할 수 있도록 요구사항을 우선순위화하는 절차를 제안한다. 또한, 사례연구를 통해 아키텍처 패턴 유형에 근거한 요구사항의 우선순위화가 품질속성 측면에서 효율적인 소프트웨어 아키텍처 설계에 도움이 됨을 보인다.

1. 서론

소프트웨어 아키텍처는 소프트웨어 요구사항을 기반으로 설계되고, 요구사항은 중요도를 설정하여 우선순위화 할 수 있다[1]. 아키텍트가 소프트웨어 아키텍처를 설계하는 데 보통 경험이 많은 아키텍트의 직관에 의지하는 경우가 많다[2]. 그럼에도 아키텍처 설계 방법론이 많이 연구되고 있는데, 기존의 패턴 연구나 관련 서적에서는 각 패턴에 대해 설명만 할 뿐 여러 패턴을 적용하는 세부적인 방법이나 적용 순서에 따른 효과에 대해서는 다루고 있지 않다[1,3,4]. 동일 시스템에 같은 세트(Set)의 패턴들을 적용할 때 적용 순서에 따라 아키텍처 설계 결과가 달라질 수 있는데, 예를 들면 실무에서 미디어 스트리밍 시스템 설계에 자주 사용하는 Pipe-and-Filter 패턴과 Broker 패턴이 적용되는 순서에 따라 최종 아키텍처가 상이해지는 경우들이 있다. 이러한 현상적 논리를 바탕으로 요구사항 별로 적용할 패턴을 분류하는 방안을 제시하고, 분류한 패턴들을 적용하는 순서를 정하기 위해 요구사항을 체계적으로 우선순위화하는 방법을 제안한다. 또한 사례연구를 통해 제안한 방법론이 품질속성 측면에서 효율적인 아키텍처 설계에 도움이 됨을 보인다.

2. 접근 방법: 패턴 기반 요구사항 분류체계

본 논문에서는 요구사항 별로 적용할 패턴을 분류하고 이를 기반으로 요구사항을 우선순위화하여 순서대로 적용하는 패턴 기반 요구사항 분류체계 (Pattern-Based Requirements Management, PRM)를 제안한다. PRM 방법을 설명하기에 앞서 패턴 분류에 활용할 수 있는 주요 아키텍처 패턴 카탈로그를 표 1과 같이 새롭게 정리한다.

표 1. 주요 아키텍처 패턴 카탈로그

패턴명	적용 속성 키워드
Layered 패턴	독립 개발, 분리, 호환성, 모듈성, 재사용, 추상화, 계층, 테스트 용이, 조직간 개발 용이, 밀집성, 공개 인터페이스, 의존적, 격리, 라이브러리 등
Broker 패턴	분산, 가용성, 다수의 서버, 리다이렉션, 브로커, 명령 전달, 요청, 프록시 등
Client-Server 패턴	상호작용, 다수의 클라이언트, 서버, 통신, 분리, 포트, 요청, 응답, 동시접속, DB, 보안, 데이터 이중화, 명령 전달, 프로토콜 등
Publish-Subscribe 패턴	출판, 구독, 이벤트, 주기적, 비동기 이벤트, 이벤트 버스, 커넥터 등
Peer-to-Peer 패턴	분산, 동시성, 가용성, 확장성, 상호작용, 대칭 요청/응답, 1대1, 유동적, 탈중앙, 프로토콜, 라우팅, 인덱싱, 피어 검색 등
Model-View-Controller 패턴	사용자 인터페이스, 작은 뷰 변경, 통지, 이벤트, 콜백
Pipe-and-Filter 패턴	스트림 입출력, 데이터의 연속, 필터

그림 1에서처럼 PRM 방법론에서는 체계적으로 패턴을 분류하고 적용하는 절차를 보인다. 그림 1에서 둥근 사각형과 마름모는 행위를 나타내는 절차이고, 직사각형은 입력물 또는 산출물을 나타낸다. 패턴 적용이 필요한 요구사항을 식별할 때 요구사항 문장에서 표 1에 기술되어 있는 적용 속성 키워드와 동일하거나 유사한 단어가 있는 지 분석하고, 존재하면 해당 패턴명을 후보자 패턴으로 작성해둔다. 패턴 중 물리적인 단위와 관련된 패턴을 1st Priority, 개별 노드와 관련된 패턴을 2nd Priority, 그 외 패턴 적용이 어려운 요구사항은 3rd Priority로 설정한다. 같은 Priority일 때 요구사항에서 등장하는 패턴의 빈도수를 추가 조건으로 사용하여 순서를 정하고 정리된 패턴을 적용한다. 또한 특정 상황에서 직관적으로 판단할 수 있는 수정필요 클래스(Class) 갯수를 통해 변경 용이성 측면에서 아키텍처를 평가해본다.

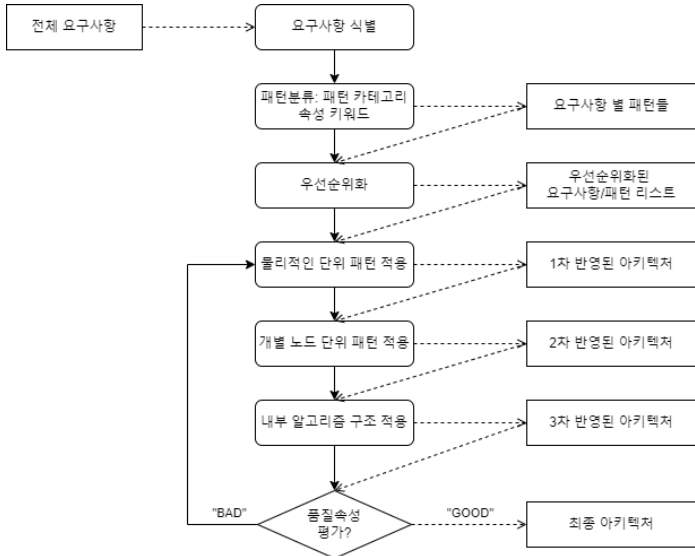


그림 1. PRM 방법 절차

3. 사례 연구

사례연구를 통해 제안한 방법론의 실효성을 확인한다. Github에 공개되어 있는 open webOS의 uMediaServer(uMS) 모듈을 실험대상 소프트웨어로 선정한다. uMediaServer는 미디어 프레임워크의 중심을 담당하는 서비스이다[5].

3.1. PRM 미적용 아키텍처 및 평가

실험대상 소프트웨어는 아키텍처의 직관에 의해 설계된 것으로 보이며 Broker 패턴이나 Client-Server 패턴을 사용한 것으로 분석된다. 이벤트 전달을 위해 Publish-Subscribe 패턴이 적용될 수 있으나 콜백함수(Callback Function) 형태로 구현되어 있다. 변경 용이성 관점에서 평가해보면 Client와 Server가 분리되어 있어 업무 분리가 용이하다. 다만 인터페이스를 추가하거나 수정할 때 uMediaClient, uMediaServer, pipelinemanager, pipeline 4개의 클래스 변경이 일괄적으로 필요하고, Resource 정책 변경 시에는 ResourceManager, resource_request_t, AcquireQueue 3개의 클래스 수정이 필요해 이러한 부분들은 코드 변경이 용이하지 않을 것으로 분석된다

3.2. PRM 적용 아키텍처 및 평가

실험대상 소프트웨어의 주요 요구사항 별로 표 1의 패턴 카탈로그에서 유사 단어를 찾아 후보자 패턴을 분류한다. 분류한 패턴을 기반으로 요구사항들을 물리적 단위 패턴, 개별 노드 패턴, 기타의 순서로 우선순위화하고 등장한 패턴의 빈도수에 따라 같은 우선순위 내에서 추가 조건으로 사용한다. PRM 방법을 적용한 우선순위화는 표 2와 같고, 패턴들을 적용한 최종 아키텍처는 그림 2와 같다. 이는 인터페이스를 추가하거나 수정할 때 uMediaClient 클래스만 수정하면 되고, Resource 정책 변경 시 PipelinePolicy 클래스만 변경되어 변경량이 기존 아키텍처에 비해 적어지게 된다.

표 2. 실험대상 소프트웨어 주요 요구사항 우선순위화

순번	요구사항	후보자 패턴	PRM 기준
1	앱에서 지정한 비디오 재생 파이프라인에 명령을 전달할 수 있어야 한다.	Client-Server Broker Peer-to-Peer	Priority 1
2	앱에서 지정한 카메라 파이프라인에 명령을 전달할 수 있어야 한다.	Client-Server Broker Peer-to-Peer	Priority 1
3	파이프라인은 분리되어 앱에서 직접 호출하면 안 된다.	Layered	Priority 2
4	uMS는 재생과 관련된 알림 이벤트와 에러를 전달해 주어야 한다.	Publish-Subscribe	Priority 1
5	다수의 앱이 재생 명령을 동시에 전달할 수 있어야 한다.	Client-Server Peer-to-Peer	Priority 1
6	uMS는 재생할 수 있는 앱의 우선순위를 지정해야 한다.	NO	Priority 3
7	통신체계는 추상화되어야 한다.	Layered	Priority 2
8	uMS는 항상 실행되어 있어야 한다.	NO	Priority 3
9	uMS는 앱에게 프록시 형태의 분리된 통신 라이브러리를 제공한다.	Broker Layered Peer-to-Peer	Priority 2
10	명령 요청은 10ms 내에 리턴 된다.	NO	Priority 3

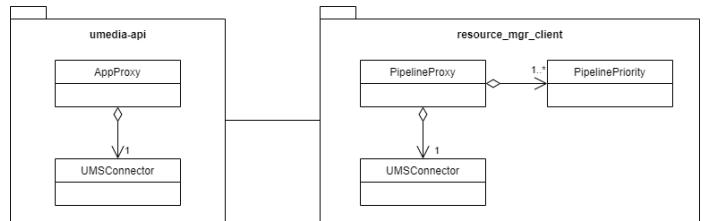


그림 2. PRM 적용 아키텍처

4. 연구 결과

인터페이스 추가 시 기존 아키텍처는 4개의 클래스가 변경되어야 하지만 PRM을 적용한 아키텍처는 1개의 클래스가 변경되어 변경 용이성 측면에서 더 효율적인 것으로 분석된다. 이러한 연구 결과를 바탕으로 효율적인 설계를 체계적으로 할 수 있는 PRM을 제안한다.

5. 결론 및 향후 연구

본 연구에서는 PRM을 제안하고, 사례연구를 통해 실효성을 확인하였다. 향후 연구에서는 우선순위화에 대해 더 효율적인 방안을 연구하고, 대규모 요구사항 시스템에서 의존성이 있는 요구사항들을 그룹핑하면서 PRM을 적용하고 다른 품질속성을 평가항목으로 아키텍처를 평가해 볼 수 있을 것이다.

참고문헌

[1] Bass, Len, Paul Clements, and Rick Kazman. Software architecture in practice.(3rd ed.). Addison-Wesley Professional, 2013.
 [2] Kruchten, Philippe. "Mommy, where do software architectures come from." Proceedings of the 1st Intl. Workshop on Architectures for Software Systems. 1995.
 [3] Richards, Mark. Software architecture patterns. Vol. 4. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Incorporated, 2015.
 [4] Kim, Dae-Kyoo, and Charbel El Khawand. "An approach to precisely specifying the problem domain of design patterns." Journal of Visual Languages & Computing 18.6 (2007): 560-591, 2007
 [5] Github URL, <https://github.com/webosose/umediaserver>

소프트웨어 요구사항 분류 모델의 성능 향상을 위한 불균형 데이터 처리에 관한 연구

최종우[○] 이영준 임채균 최호진

한국과학기술원

{jongwoo.choi, yj2961, rayote, hojinc}@kaist.ac.kr

A Study on Handling Imbalanced Data to Improve Performance of Software Requirements Classification Models

Jong-Woo Choi[○], Young-Jun Lee, Chae-Gyun Lim, Ho-Jin Choi

Korea Advanced Institute of Science and Technology

요 약

정확한 소프트웨어 요구사항 카테고리 분류는 품질속성 기반 아키텍처 설계에 많은 영향을 준다. 고비용 작업인 요구사항 분류에 관한 자연어처리 모델이 많이 연구되고 있지만, 데이터 증강 기법을 이용한 품질속성 요구사항 데이터 셋(dataset)의 불균형 개선에 대해서는 많이 다루고 있지 않다. 본 연구에서는 먼저, 실험을 통해 분류 모델들이 한국어 요구사항 데이터 셋을 자동으로 분류할 수 있음을 보인다. 이 결과를 바탕으로 EDA(Easy Data Augmentation) 기법을 통한 데이터 증강과 언더샘플링(undersampling) 전략으로 데이터 셋의 불균형을 개선할 수 있음을 설명하고, 실험을 통하여 분류 성능 향상에 효과가 있음을 보인다. 실제 실험 결과 F1 점수 기준으로 최대 5.24 향상되어 데이터 증강과 언더샘플링 기법이 분류 모델의 한국어 요구사항 분류에 도움이 됨을 확인 할 수 있다.

1. 서 론

주로 자연어로 작성되는 소프트웨어 요구사항은 기능 요구사항과 비기능 요구사항으로 구분되고, 비기능 요구사항은 품질속성과 제약사항으로 분류된다. 실무에서는 품질속성 기반 설계를 위해 아키텍트나 개발자들이 품질속성 요구사항을 수작업으로 카테고리 분류를 하는데, 이는 많은 시간과 노력이 필요한 고비용의 업무이다. 이로 인해 자동으로 소프트웨어 요구사항을 분류하는 자연어처리 방법들이 연구되고 있다[1,2]. 요구사항 분류 연구에 보통 사용되는 PROMISE NFR[3] 데이터 셋은 데이터 불균형을 이루고 있으나 데이터 증강 기법을 이용한 연구가 적고, 더욱이 한국어 소프트웨어 요구사항 분류에 대해서는 다루고 있지 않다. 본 연구에서는 자동 분류 모델들의 소프트웨어 요구사항 분류 성능이 향상되도록 데이터 셋을 처리한다. 먼저 실험을 통해 분류 모델들이 한국어 요구사항 데이터 셋을 자동으로 분류할 수 있음을 보인다. 이 결과를 바탕으로 EDA(Easy Data Augmentation) 기법[4]을 통한 데이터 증강과 언더샘플링(undersampling) 전략을 적용해 요구사항 데이터 셋의 불균형을 개선할 수 있음을 설명하고, 실험을 통하여 이 기법들이 분류 성능 향상에 효과가 있음을 보인다.

2. 관련 연구

요구사항 분류 모델에 관한 연구들과 불균형 요구사항 데이터에 관한 연구들이 있다. Navarro-Almanza et al.[1]은 합성곱 신경망(Convolutional Neural Network, CNN) 모델을 이용하여 다중 분류에서 F1 점수 기준 77%를 달성 하였으나 불균형 데이터 처리에 대한 방법은 기술하지 않고 있다. Hey

et al.[2]은 BERT 기반의 NoRBERT 모델을 제시하고, 불균형 데이터 처리를 위해 오버샘플링(oversampling)과 언더샘플링 기법을 활용해 다중 분류에서 F1 점수 기준 90%를 기록하였다. Lima et al.[5]은 PROMISE NFR 데이터 셋에 데이터를 추가한 PROMISE_exp 데이터 셋을 제시하였고, 이는 F1 점수 기준으로 0.01 향상 시켰다. 본 논문에서는 알려진 불균형 데이터 개선 방법 중에 상대적으로 구현 복잡도가 낮은 언더샘플링 기법과 문맥을 유지하면서 데이터 증강을 하는 EDA 기법을 이용해 요구사항 데이터 셋의 불균형을 개선하여 분류 모델의 한국어 요구사항 분류 성능을 개선 하고자 하였다.

3. 연구 방법

3.1. 한국어 요구사항 데이터 셋

한국어 요구사항 데이터 셋은 영문 PROMISE NFR[3] 데이터 셋을 구글 번역을 통해 한글로 번역하여 생성한다. 이 데이터 셋은 255개의 기능 요구사항과 370개의 비기능 요구사항으로 구성되어 있고, 비기능 요구사항은 각각 11개의 품질속성 클래스로 분류되어 있으며 데이터 셋은 클래스 별로 최대 255개, 최소 1개 데이터 가지고 있어 불균형을 이루고 있다. 데이터 셋을 8:2 비율로 학습 데이터와 실험 데이터로 분리하여 사용한다.

3.2. 소프트웨어 요구사항 분류 모델

구글에서 발표한 다국어 버전의 언어모델인 Multilingual BERT[6]와 뉴스와 위키피디아 등의 한국어 코퍼스를 사전에 학습한 BERT 계열의 모델인 bert-kor-base[7], 한국 뉴스

기사의 댓글들을 수집한 데이터를 학습한 kcbert[8]를 이용한다.

3.3. 데이터 불균형 개선 방법 1: 데이터 증강

EDA(Easy Data Augmentation)[4]는 유의어 교체, 랜덤 삽입, 랜덤 교체, 랜덤 삭제와 같은 개별 연산으로 이루어진다. 한국어 유의어를 위해 KAIST(한국과학기술원)에서 공개한 KWN(Korean WordNet)[9]을 이용하고, 생성된 문장의 클래스는 원본의 클래스를 그대로 적용한다.

3.4. 데이터 불균형 개선 방법 2: 언더샘플링

데이터 수가 1개인 PO(Portability) 클래스의 데이터는 제외하고 그 다음으로 적은 10개 데이터로 구성된 FT(Fault Tolerance) 클래스를 기준으로 다른 클래스의 데이터를 임의로 언더샘플링하여 데이터 10개로 균형을 맞춘다.

4. 실험 구성 및 결과

분류 모델의 하이퍼파라미터는 Learning Rate 2e-5, Batch Size 32, Epoch 32로 설정하고 EDA 연산 파라미터는 각각 0.1로 설정하며 평가는 F1 점수(F1-score)로 비교한다. 실험 결과는 각 연구질문(RQ) 결과를 확인한다.

4.1. [RQ1] 한국어 소프트웨어 요구사항도 분류가 잘 되는가?

한국어 요구사항을 분류한 bert-kor-base가 영문 요구사항을 분류한 합성곱 신경망(CNN)보다 F1 점수 기준 3.15 높은 성능을 보였지만 BERT Base보다는 3.77 낮은 성능을 보였는데, 이는 데이터 셋의 번역 오류로 인한 차이로 생각된다. 약간의 성능 차이는 있지만 분류 모델이 한국어 요구사항도 분류에도 적합함을 알 수 있다.

4.2. [RQ2] 불균형 데이터 개선이 요구사항 분류 모델의 성능을 향상 시키는가?

이진 분류에서는 bert-kor-base가 F1 점수 기준 91.91, kcbert-base가 89.55, kcbert-large가 92.73, Multilingual BERT가 91.60의 성능을 보였고, EDA 기법 적용 시에 bert-kor-base와 Multilingual BERT가 각각 0.67, 0.33 소폭 상승을 보였지만 kcbert-base와 kcbert-large는 하락하는 결과를 보였다. 이는 이진 분류를 위해 비가능 요구사항들을 하나로 통합하면서 데이터 불균형이 일부 개선되어 EDA 기법이 큰 도움이 되지 않은 것으로 보인다. 다중 분류 실험에서는 표 1에서 보는 것처럼 EDA 기법 적용 시 모든 모델의 다중 분류 성능이 향상되어 불균형 데이터가 있을 때 EDA 기법이 분류 모델의 성능 향상에 도움이 됨을 확인할 수 있다. 언더샘플링만 적용할 때 kcbert-large만 3.21 상승하고 나머지 모델은 하락하였지만, 언더샘플링과 EDA를 조합하면 kcbert-large가 5.24, kcbert-base가 4.16, Multilingual BERT가 0.19의 성능 향상을 보이고 bert-kor-base는 6.92 하락을 하였다. 이를 통해 불균형 데이터에 문제가 더 큰 kcbert-base와 kcbert-large에 언더샘플링과 데이터 증강을 조합한 기법이 큰 도움이 될 수 있음을 확인할 수 있다.

표 1. 적용 기법 별 다중 분류 결과

분류 모델	정밀도(%)	재현율(%)	F1 점수(%)	차이
bert-kor-base	81.45	80.41	80.15	0
+EDA	84.36	83.97	83.75	+3.6
+UNDER	81.82	77.27	76.34	-3.81
+UNDER+EDA	82.06	75.00	73.23	-6.92
kcbert-base	72.46	73.60	72.16	0
+EDA	74.50	75.60	74.50	+2.46
+UNDER	80.30	68.18	69.09	-3.07
+UNDER+EDA	80.30	77.27	76.32	+4.16
kbert-large	74.30	75.00	73.76	0
+EDA	74.95	75.00	73.99	+0.23
+UNDER	83.33	77.27	76.97	+3.21
+UNDER+EDA	82.92	79.55	79.00	+5.24
Multilingual BERT	74.19	74.36	73.04	0
+EDA	74.91	75.32	74.48	+1.44
+UNDER	67.27	68.18	62.77	-10.27
+UNDER+EDA	82.06	75.00	73.23	+0.19

5. 결론 및 향후 연구

언더샘플링과 데이터 증강 기법을 활용하면 F1 점수 기준 최대 5.24 향상되어 데이터 불균형 개선이 분류 모델의 한국어 요구사항 분류에 도움이 될 수 있을 것으로 판단된다. 향후 연구에서 한국어 요구사항을 수집할 수 있고, 또한 Stack overflow와 같은 플랫폼에서 소프트웨어 용어 수집을 통해 워드넷을 보완하면 EDA의 유의어 관련 연산 효과를 더 높여볼 수 있을 것으로 보인다. 추가로 오버샘플링 기법을 이용하면 다각도로 분류 성능을 비교해볼 수 있을 것이다

감사의 글

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2013-2-00131, (엑소브레인-총괄/1세부) 휴먼 지식증강 서비스를 위한 지능진화형 WiseQA 플랫폼 기술 개발)

참고문헌

- [1] Navarro-Almanza, Raul, Reyes Juarez-Ramirez, and Guillermo Licea. Towards supporting software engineering using deep learning: A case of software requirements classification. In: 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT). IEEE, p. 116-120, 2017
- [2] Hey, Tobias, et al. NoRBERT: Transfer learning for requirements classification. In: 2020 IEEE 28th International Requirements Engineering Conference (RE). IEEE, p. 169-179, 2020.
- [3] <http://promise.site.uottawa.ca/SERepository/>
- [4] Wei, Jason, and Kai Zou. "Eda: Easy data augmentation techniques for boosting performance on text classification tasks." arXiv preprint arXiv:1901.11196 (2019).
- [5] Lima, M.; Valle, V.; Costa, E.; Lira, F.; Gadelha, B. Software Engineering Repositories: Expanding the PROMISE Database; XXXIII Brazilian Symposium on Software Engineering; SBC: Porto Alegre, Brasil; pp. 427-436, 2020
- [6] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805, 2018.
- [7] <https://github.com/kiyoungkim1/LMkor>
- [8] <https://github.com/Beomi/KcBERT>
- [9] <http://wordnet.kaist.ac.kr/>

협동 로봇의 센싱 데이터 특징 기반 테스트 데이터셋 구성을 통한 학습 모델 신뢰성 분석

김진세^{1(O)}, 배수빈¹, 박예슬¹, 이정원^{1,2}

아주대학교 시융합네트워크학과¹, 아주대학교 전자공학과²

jinsae913@gmail.com, bpgs5316@ajou.ac.kr,

yeseuly@gmail.com, jungwony@ajou.ac.kr

Reliability Analysis of Learning Model by Sensing Data Feature-based Test-Dataset Construction for Collaborative Robots

Jinse Kim^{1(O)}, Subin Bea¹, Ye-Seul Park¹, and Jung-Won Lee^{1,2}

Department of AI Convergence Network, Ajou University¹

Department of Electrical and Computer Engineering, Ajou University²

요 약

최근 스마트팩토리의 대표 설비 기기인 협동 로봇의 건전성 관리 기술(PHM)에 인공지능 모델을 접목하여 상태를 진단하고 고장 징후를 포착하는 시도가 이어지고 있다. 하지만 현재 개발되고 있는 진단 모델은 일부 테스트 프로그램에 대해 한정적으로 데이터를 수집하거나, 수집된 데이터에 대한 별도의 체계적인 전처리나 특징에 대한 분석 없이 휴리스틱한 방식으로 개발이 이루어지고 있다. 따라서 본 논문에서는 협동 로봇에서 생성되는 시계열 센싱 데이터의 특징을 분석하여 테스트 데이터셋을 구성하고, 학습 모델의 성능 평가 결과에 따라 개발 과정에 피드백할 수 있는 특징 클러스터 기반 학습 모델 신뢰성 검증 방법을 제안한다. 또한, 이를 실제 분류 모델에 적용하여 효용성을 검증한다.

1. 서 론¹

IIoT(Industrial Internet of Things) 기술의 발전으로 건전성 관리 기술(PHM: Prognostics and Health Management)이 필수적으로 적용되는 스마트팩토리 산업이 활성화되고 있으며[1], 대표적인 스마트 기기인 협동 로봇(Cobot, Collaborative Robot)의 상태 모니터링 및 고장 징후 포착을 위해 인공지능 기술이 활용되고 있다. 협동 로봇의 진단 데이터는 수행 가능한 작업의 다양성으로 인해 다른 설비 기기보다 높은 복잡도를 지닌다. 현재 대부분의 연구에서는 정밀한 진단을 위해 고정된 테스트 환경에서 진단 모델을 개발하고 있다. 또한, 일부 테스트 프로그램에 한정적으로 데이터를 수집하였거나, 실험으로부터 획득한 데이터에 대한 체계적인 전처리나 특징에 대한 분석 없이 시행착오, 즉 휴리스틱(Heuristic)한 방식으로 모델을 개발하고 있다.

전체 데이터 셋에서 학습 데이터를 일정 비율로 무작위 추출(Random Sampling)하여 훈련 및 검증 데이터로 활용하는 보편적인 모델 개발 방식[2-4]의 경우, 데이터의 특정 측면에 편향되어 학습될 수 있으며, 모델의 성능 평가도 데이터의 특징 분포에 따라 체계적으로 분석되어 이루어지기 어렵다.

따라서 본 논문에서는 협동 로봇 건전성 관리 기술에 적용되는 시계열 센싱 데이터의 특징을 분석하여, 데이터 특성에 따라 학습 모델의 성능을 평가하고 분석된 결과를 다시 개발 과정에 피드백할 수 있는 특징 클러스터 기반 학습 모델 신뢰성 검증 방법을 제안한다. 제안하는 방식은 모델 개발자에게 모델 및 훈련 데이터의 취약점을 분석할 수 있게 하며, 체계적인 모델 개발 과정을 구축하도록 한다. 이에 대한 적용 사례로 협동 로봇의 모션 결함 검출 네트워크의 프로그램 분류 모델의 취약성을 확인한다.

2. 협동 로봇 데이터 특성 분석

본 장에서는 실제 협동 로봇에서 수집된 데이터 샘플을 통해 데이터의 특징을 시간, 공간, 운행 관점에서 분석한다.

2.1 시계열 특성 분석

협동 로봇의 센싱 데이터는 대표적인 시계열 데이터로, 시간에 따른 상태 변화를 반영하여 데이터가 수집된다. 시계열 데이터의 분석은 고정된 시간 구간의 관측치를 기준으로 이루어지며, 관측 범위에 따라

¹ 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2020R1A2C1007400).

패턴을 발견할 수 있는지 여부가 달라진다. 창크기와 샘플링 위치에 따라 시계열 데이터의 구간별 패턴이 다르게 나타나며, 샘플링된 데이터의 주요 특징을 추출하기 위해서는 분할된 데이터의 유사도 측정이 필요하다.

2.2 파라미터 별 특성 분석

데이터 파라미터(Data Parameter)는 센서의 종류, 센서가 장착된 위치에 의해 구별되어, 하나의 물리적 특징을 설명하는 데이터 타입(Type)을 나타낸다. 협동 로봇 센싱 데이터를 기반으로 학습 모델을 개발하는 경우, 모델의 성능 개선을 위해 입력 데이터 파라미터에 따른 특성을 파악해야 한다.

2.3 동작 특성 분석

협동 로봇의 동작에 따라 생성되는 데이터는 다양한 작업 조건(Task Condition)의 영향을 받는다. 따라서 협동 로봇 센싱 데이터는 프로그램, 모션의 조합에 의해 다양한 패턴의 데이터를 생성하게 되며, 이러한 특성을 고려하여 모델의 성능을 분석해야 한다.

3. 특징 클러스터 기반 학습 모델 신뢰성 검증

본 장에서는 특징 클러스터 기반의 테스트 데이터셋 구성을 통한 학습 모델 신뢰성 검증 방법을 소개한다.

3.1 특징 클러스터 생성

특징 클러스터 생성 단계에서는 학습 데이터의 특성을 분류한다. 이에 대한 예시로 그림 1은 분류 모델(Classification Model)에서 사용되는 학습 데이터를 클러스터링하여 특징 클러스터를 생성하는 과정을 나타낸다.

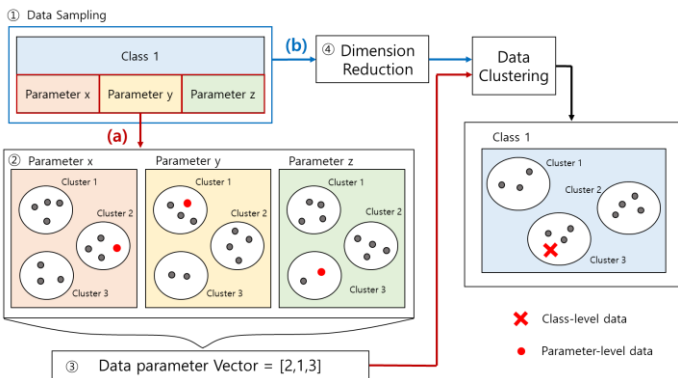


그림 1 학습 방식에 따른 특징 클러스터 생성 기법

3.2 특징 클러스터 기반 테스트 데이터셋 구성

특징 클러스터의 생성 이후에는 각 클러스터가 가지는 특징을 고려한 테스트 데이터셋을 구성한다. 세부적으로 클래스 별 특징 클러스터 분석, 클래스 별 데이터 분할, 클러스터 별 데이터 분석을 통해 테스트 데이터셋을 구성하며, 그림 2는 이에 대한 예시를 나타낸다.

(a)

Class 1			Class 2		
FC 1-1	FC 1-2	FC 1-3	FC 2-1	FC 2-2	FC 2-3
102	84	114	128	104	68

(b)

Train						Test					
FC 1-1	FC 1-2	FC 1-3	FC 2-1	FC 2-2	FC 2-3	FC 1-1	FC 1-2	FC 1-3	FC 2-1	FC 2-2	FC 2-3
90	50	100	106	92	42	14	34	14	22	12	26

(c)

Class 1			Class 2			Class 1			Class 2		
FC 1-1	FC 1-2	FC 1-3	FC 2-1	FC 2-2	FC 2-3	FC 1-1	FC 1-2	FC 1-3	FC 2-1	FC 2-2	FC 2-3
Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
90	14	50	34	100	14	106	22	92	12	42	26

TC 1 TC 2 TC 3 TC 4 TC 5 TC 6

그림 2 클러스터 기반 테스트 데이터 셋 구성

3.3 테스트 데이터셋 기반 학습 모델 취약성 분석

본 절에서는 테스트 데이터셋 기반의 학습 모델의 신뢰성 분석 방법을 소개한다. 데이터셋의 클래스 내부에 존재하는 테스트 클러스터에 따라 성능을 평가하여 취약한 클러스터를 확인할 수 있으며, 이를 보완하는 방향으로 모델의 개선 방안을 확립할 수 있다.

4. 구현 및 실험

본 장에서는 제안하는 방법의 적용 가능성을 확인한다. 대상은 협동 로봇 결함 검출 네트워크의 프로그램 분류 모델(Program Classification Model)이다.

5. 결론

본 논문은 협동 로봇의 건전성 진단에 사용되는 센싱 데이터의 특징 분석 기반 테스트 데이터셋 구성을 통해 학습 모델의 신뢰성을 평가한다. 제안하는 방법은 일반적으로 사용되는 학습 모델의 평가 방법으로 확인이 어려운 클래스 내부의 특징에 대해 분석할 수 있도록 하여, 학습 모델의 편향성과 취약성에 대한 세부적인 분석이 가능하게 한다. 또한, 구체적인 모델 개선 방향을 제시함으로써 체계적인 모델 개발 과정의 구축을 지원한다.

참고 문헌

[1] Chekired, Djabir, Lyes, et al. Mouftah. "Industrial IoT data scheduling based on hierarchical fog computing: A key for enabling smart factory." IEEE Transactions on Industrial Informatics 14.10 (2018): 4590-4602.

[2] Yadav, Sanjay, and Sanyam Shukla. "Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification." 2016 IEEE 6th International conference on advanced computing (IACC). IEEE, 2016.

[3] Kim, Ji-Hyun. "Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap." Computational statistics & data analysis 53.11 (2009): 3735-3745.

[4] Kadilar, Cem, and Hulya Cingi. "A new ratio estimator in stratified random sampling." Communications in Statistics—Theory and Methods 34.3 (2005): 597-602.

AIAG-VDA 방법론 기반 SFMEA : 효과적인 자동차 SW 안전분석

김소연^o 권기현

경기대학교

se.ksy0513@kyonggi.ac.kr, khkwon@kyonggi.ac.kr

SFMEA based on AIAG-VDA FMEA methodology : Effective automotive SW safety analysis

So-Yeon Kim^o Gi-Hwon Kwon

Kyonggi University

요 약

자율주행에서 가장 기본이 되는 센서는 기능이 잘못 작동하는 경우 큰 사고로 이어지므로 안전성 확보가 요구된다. 잘못된 요구사항 분석과 설계는 소프트웨어 에러의 주요 원인으로 초기에 결함을 제거할 시 낮은 비용으로 안전성 확보가 가능하므로 이를 초기에 확인하는 소프트웨어 안전 분석의 중요성이 강조된다. 자동차 기능 안전 표준에서는 안전 분석 시 FMEA 기법을 사용하는 것을 권장하며, 이를 위하여 AIAG-VDA FMEA Handbook이 만들어 사용되고 있다. 하지만 이는 소프트웨어 수준에서의 안전 분석을 수행하는데 어려움이 있다. 소프트웨어 안전 분석을 위하여 SFMEA가 제시되었지만 이는 자동차 소프트웨어 개발 프로세스를 반영하기 어렵다. 본 논문에서는 AIAG-VDA FMEA 수행 절차를 소프트웨어 수준의 분석에 적합하게 개선하고 소프트웨어 특성에 따른 분석 수행을 위하여 SFMEA를 적용한 SW-FMEA-DEV를 제안하고 이를 ADAS AK-2 센서 소프트웨어 안전 분석에 적용한다. 적용 결과 소프트웨어 수준에서의 FMEA 수행이 가능하였으며, 기존의 SFMEA로는 식별이 어려웠던 자동차 소프트웨어의 특징을 반영한 분석이 가능하였다.

1. 서 론

자율 주행 자동차의 센서는 인지, 탐지, 이탈 등을 판단하기 위한 사람의 눈과 같은 존재이며 자율주행에서 가장 기본이 되는 시스템이다. 최근 자율 주행 자동차 시스템 고도화 및 시장 발전으로 차량에 탑재되는 센서의 수와 요구되는 기술력이 향상하고 있으며 다수 개의 시스템이 상호작용하여 하나의 기능을 제공하므로 시스템의 복잡도와 상호작용이 함께 증가한다[1]. 시스템 제어 역할이 하드웨어에서 소프트웨어로 점차 이동함에 따라 시스템이 고도화될수록 이를 제어하는 소프트웨어의 복잡도 및 상호작용이 함께 증가한다. 센서의 기능이 올바르게 작동하지 않는 경우 인명 및 환경의 큰 피해로 이어질 수 있으며 재산 손실이 발생할 가능성이 있으므로 실 사용 전 시스템 및 제어 소프트웨어의 안전성을 확보하는 것이 중요하다.

소프트웨어 안전 분석은 개발하는 소프트웨어가 안전 요구사항을 만족하는 지를 확인하는 것이다. ISO26262에서는 요구사항 분석 및 설계 단계의 안전 분석 수행을 위하여 FMEA(Failure Mode and Effect Analysis) 기법 사용을 권장한다[2]. 자동차 시장에서 일관된 방법으로 안전 분석을 수행하기 위하여 자동차 산업 행동 그룹(AIAG)과 자동차 산업 협회(VDA)가 협력하여

AIAG-VDA FMEA Handbook을 발간하여 FMEA 수행 표준을 수립하였으나 이는 시스템 수준에서의 FMEA 수행을 목적으로 소프트웨어 수준에서의 FMEA를 수행하는데 어려움이 있다[3]. SFMEA(Software-FMEA) 기법은 소프트웨어 수준에서 FMEA를 수행하기 위하여 약 20만여개의 소프트웨어에 대하여 FMEA를 수행한 결과를 정리하여 제안된 기법이다[4]. 이 기법은 소프트웨어의 특성을 반영한 안전 분석을 수행하고자 소프트웨어를 분석하는 관점(Viewpoints)을 선택하고 이에 따라 발생 가능한 고장 유형을 분석한다. 하지만 제안된 관점들이 전통적인 폭포수 모델을 기반으로 추론 되어 소프트웨어 복잡도 및 상호작용 증가와 이에 따른 요구사항 분석 시 표준 모델링 언어를 사용하는 등의 프로세스가 도입된 자동차 소프트웨어 분석을 고려하기 어렵다.

본 연구에서는 안전한 자동차 소프트웨어 개발을 위하여 주로 시스템 수준에서 수행되는 FMEA를 소프트웨어 수준의 분석에 적합하도록 개선하는 것을 목적으로 한다. 이를 위하여 시스템 수준 안전 분석을 위하여 AIAG-VDA FMEA 프로세스를 소프트웨어 수준에 적합하게 개선하고, 소프트웨어의 특징을 고려한 안전 분석을 수행하고자 SFMEA에서 제안한 관점에 따른 안전 분석을 수행한다. 안전 분석 수행 시 자동차 소프트웨어의 특성을 반영하고자 새로운 분석 관점을 정의한다.

2. AIAG-VDA 방법론 기반 SFMEA

AIAG-VDA FMEA Handbook에서 제안된 시스템 수준 안전 분석 수행 프로세스를 기반으로 SFMEA의 안전 분석 수행 방법을 적용하여 자동차 소프트웨어 안전 분석에 적합하게 개선한 FMEA 프로세스를 SW-FMEA-DEV라 칭하며 전체 프로세스는 그림 1과 같다.

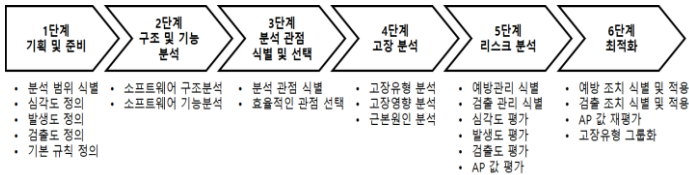


그림 1. SW-FMEA-DEV 프로세스

AK-2 센서 시스템은 초음파를 출력하고 반사된 초음파를 읽어 외부 사물과의 거리 값을 계산한다. 이 중 분석하고자 하는 영역인 제어 소프트웨어 구조는 그림 2와 같다. 이 중 제어 소프트웨어에서는 초기화 및 거리 값 측정이 이루어진다.

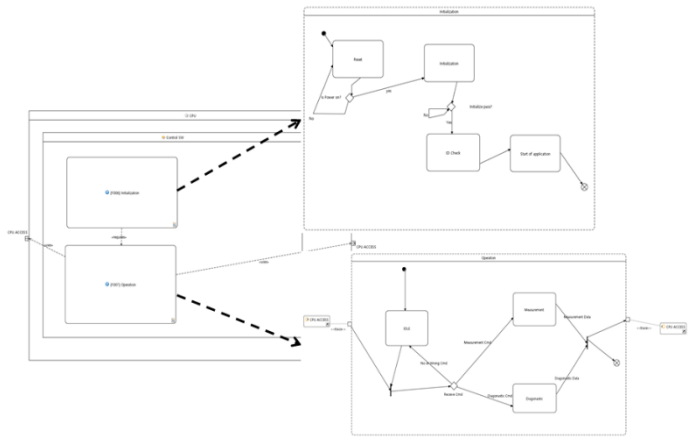


그림 2. AK-2제어 소프트웨어 구조

기존 시스템 수준에서만 수행하던 구조 및 기능 분석을 소프트웨어 수준까지 진행한 뒤 고장 분석, 리스크 분석, 최적화를 차례로 시행한다. 고장 분석 시 SFMEA에서 제안한 방법에 따라 소프트웨어를 분석할 관점을 선택하고 이에 따라 발생 가능한 고장을 분석한다. 자동차 시스템 고도화에 따라 변화된 자동차 소프트웨어 개발 프로세스로 인하여 SFMEA에서 제시한 관점으로는 분석이 어려운 부분에 대한 추가 분석을 수행하고자 새로운 분석 관점을 정의한다. 기존의 관점으로 분석이 어려운 부분은 소프트웨어 간 상호작용에 따른 고장이다. 안전 요구사항 분석 시 이러한 점을 고려하기 위하여 유스케이스 다이어그램을 사용하므로, 유스케이스 다이어그램을 분석하여 새로운 분석 관점을 정의한다. 소프트웨어 간 상호작용을 고려하여 새롭게 정의한 분석 관점은 표 1과 같다.

표 1. 새롭게 식별된 소프트웨어 분석 관점

관점	대상	프로세스
Pre-condition	소프트웨어 안전 요구사항 소프트웨어 간 인터페이스	소프트웨어 안전 요구사항 식별
Post-condition	소프트웨어 안전 요구사항 소프트웨어 간 인터페이스	소프트웨어 안전 요구사항 식별

SFMEA에서 제시한 8개의 분석 관점 및 새롭게 정의한 관점 중에서 분석하고자 하는 관점을 선택하고 고장 분석 및 리스크 분석을 수행한다.

3. 평가 및 정리

실험을 통해 확인한 내용은 다음과 같다. 우선 제안한 방법을 실제 자동차에 탑재되는 시스템에 적용하여 소프트웨어 수준에서의 FMEA를 통한 안전 분석 수행이 가능함을 확인하였고, 다수 개의 상호작용을 가진 소프트웨어에도 적용 가능함을 확인하였다. 적용 결과 소프트웨어 구성요소 개별에서는 문제가 발생하지 않더라도 이를 수행하기 위한 사전 및 사후 조건으로 인해 고장 발생이 가능하며, 소프트웨어 복잡도 및 상호작용이 증가함에 따라 사전 조건 및 사후 조건이 상호작용하는 다른 소프트웨어 구성요소에 의해 결정되는 경우가 빈번하고 이 경우 관련 문제 발생 가능성이 증가한다. 또한 특정 소프트웨어 구성요소에서는 고장이 발생하지 않거나 고장 유형의 심각도가 높지 않아도 해당 소프트웨어 구성요소의 출력 값이 다른 소프트웨어 구성요소의 사전 조건으로 작용하는 경우 영향을 받는 소프트웨어에서는 치명적인 경우가 존재하는 것을 확인하였다. 향후 연구에서는 다수 개의 사례를 적용하여 제안 방법의 효과성을 추가적으로 확인하고 새롭게 정의한 분석 관점에 대하여 추가적으로 발생 가능한 고장 유형의 여부를 확인하고자 한다.

참고문헌

[1] J. Giacalone, L. Bourgeois and A. Ancora, "Challenges in aggregation of heterogeneous sensors for Autonomous Driving Systems," 2019 IEEE Sensors Applications Symposium (SAS), 1-5, 2019

[2] ISO, "ISO 26262:2018, Road Vehicles - Functional Safety - Part 4: Product development at the system level", International Organization for Standardization, 2018.

[3] AIAG and VDA, "AIAG&VDA FMEA Handbook", Automotive Industry Action Group and Verband Der Automobilindustrie, 2019.

[4] Ann Marie Neufelder, "Effective Application of Software Failure Modes Effects Analysis - 2nd Edition", Quanterion Solutions Incorporated, 2017

엣지 클라우드 환경에서 다목적 최적화 알고리즘을 활용하여 비용 및 서비스 지연을 최소화하는 마이크로서비스 배포 방법

김민협[○], 고인영

한국과학기술원

Minhyeop.kim@kaist.ac.kr, iko@kaist.ac.kr

Cost-effective and Latency-efficient deployment of Microservices Using Multi-objective Optimization Algorithms in Edge Cloud Environments

MinHyeop Kim[○], In-Young Ko

Korea Advanced Institute of Science and Technology

요 약

본 연구에서는 엣지 클라우드 환경에서 다수의 마이크로서비스로 구성된 마이크로서비스 아키텍처 기반 애플리케이션을 제공하는 과정에서 배포 비용 및 서비스 지연을 최소화하기 위한 마이크로서비스 배포 방법을 제안하였다. 본 연구는 같은 환경에서의 기존 연구에서 배포 비용 최소화라는 최적화 목표를 추가하여 확장한 연구로, 2개의 최적화 목표를 만족시키기 위하여 다목적 최적화 기법을 활용하였다. 다목적 최적화 기법으로 NSGA-II, ϵ -MOEA, PESA2를 활용하여 각 알고리즘의 성능과 최적해를 비교 분석하였다. 세 기법 중 ϵ -MOEA, PESA2는 유사한 분포의 해가 출력되었으나, NSGA-II와 비교했을 때 상대적으로 더 최적화된 해를 생성했다. NSGA-II의 경우 다른 두 기법 보다 넓은 분포의 해를 생성하였으나, 비교적 최적화되지 않은 해를 생성하였다. 세 기법은 애플리케이션 제공자의 총 배치 예산의 변화에도 불구하고 결과를 출력하기까지 큰 차이가 없는 실행시간이 걸리는 것을 확인하였다.

1. 서 론

엣지 클라우드 환경(Edge cloud environment)에서는 서비스를 클라우드 서버뿐만 아니라, 사용자가 직접적으로 접근할 수 있는 엣지 서버(Edge server)에 배포할 수 있다 [1]. 해당 엣지 서버를 활용할 수 있는 범위 안에 있는 사용자는 코어 네트워크(Core network)를 통해 서버와 통신하는 대신, 엣지 서버와 무선 접속 네트워크(Radio access network)를 통해 직접적으로 통신함으로써 상대적으로 적은 지연시간 내에 서비스를 이용할 수 있다 [1]. 엣지 클라우드 환경의 이러한 장점을 제대로 활용하기 위해서는 사용자가 직접적으로 접근할 수 있도록 하는 서비스의 배포가 중요하다.

최근 개발자들이 마이크로서비스 아키텍처(MicroService Architecture, MSA)에 기반하여 애플리케이션을 개발하면서 이러한 엣지 클라우드 환경의 특성을 활용하는 것이 중요해졌다. MSA 애플리케이션은 애플리케이션을 구성하는 여러 개의 마이크로서비스의 조합으로 되어있으며, 각 마이크로서비스는 하나의 서버에 배치될 필요 없이, 분할하여

여러 서버에 배치되어 사용자에게 제공될 수 있다 [2].

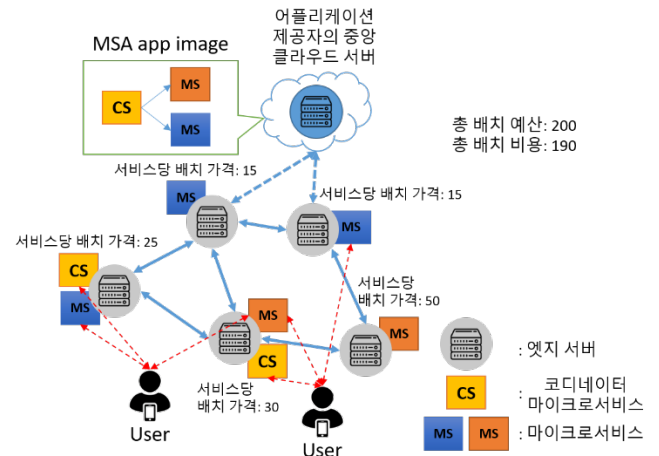


그림 1. 엣지 클라우드 환경에서의 MSA 애플리케이션 배포 예시

엣지 클라우드 환경에서 애플리케이션 제공자는 사용자의 평균 지연시간을 고려하여 애플리케이션을 구성하는 마이크로서비스를 배포하는 것이 중요하다. 그러나 사용자의 응답시간만을 고려하여 엣지 서버에 마이크로서비스를 배치할 경우 최대한 많은 수의 엣지

서버에 MSA 어플리케이션을 구성하는 모든 마이크로서비스를 배치하게 되는데, 이 경우 어플리케이션 제공자가 각 엣지 서버의 사용료로 지불해야하는 배포 비용이 너무 높아진다. 이 때문에 서비스 사용 과정에서의 평균 지연시간과 배포 비용을 함께 고려하여 어플리케이션을 구성하는 마이크로서비스를 배포하는 것이 중요하다.

그림 1은 위에서 설명한 엣지 클라우드 환경에서의 MSA 어플리케이션 배포 예시이다. 사용자가 직접 접근할 수 있는 엣지 서버 위주로 마이크로서비스를 배치하고, 불가피한 경우 간접적으로 접근할 수 있는 엣지 서버에 마이크로서비스를 배치한 상황이다.

본 연구는 기존 MSA 어플리케이션 배포 방법[3]의 후속 연구로서 기존의 평균 지연시간 최소화를 단일 최적화 목적으로 하는 모의 담금질 (simulated annealing) 기법과 달리 배포비용 최소화 역시 최적화 목적으로 하는 다목적 최적화 알고리즘 (Multi-objective Optimization algorithm)을 활용 및 분석한 연구이며, 기존 연구에서 가정한 문제 정의 및 환경 설정들을 활용하였다.

본 연구에서는 평균 지연시간 최소화 및 배포비용 최소화를 서로 상충하는 목적으로 설정하여 다목적 최적화 알고리즘을 활용하였으며, 어플리케이션 제공자는 해당 목적에 대해 파레토 최적(pareto-optimality)을 만족하는 다양한 최적해를 찾을 수 있다. 어플리케이션 제공자는 알고리즘의 결과로 산출된 다양한 최적해 중에 본인의 원하는 기준에 가장 알맞은 해를 골라 사용할 수 있다. 본 연구에서는 Non-dominated Sorting Genetic Algorithm-II (NSGA-II), ϵ -Multi-Objective Evolutionary Algorithm (ϵ -MOEA), Pareto Envelope-based Selection Algorithm (PESA2), 총 3가지의 다목적 최적화 알고리즘을 비교 실험하였다. 각 알고리즘이 출력하는 비지배 해의 최적도와 분포에 대한 분석과 각 알고리즘의 수행 시간에 대한 분석을 수행하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하고 3장에서는 본 논문에서 다루는 문제를 정의한다. 4장에서는 문제 해결을 위한 접근 방법에 대해 설명하며, 5장에서는 본 연구에서 진행한 실험 결과에 대해 설명한다. 마지막으로 6장에서는 결론 및 향후 연구에 대해 논의한다.

2. 관련 연구

최근 엣지 클라우드 환경과 유사한 포그 컴퓨팅 환경(fog computing environment)에서, 해당 환경의 장점인 짧은 지연 시간을 활용한 서비스 및 어플리케이션 제공에 관한 연구가 활발하게 진행되고 있다. 그와 함께 이러한 환경에서 사용자에게 원활한 서비스 제공을 위해 서비스를 어떻게 배치해야

하는지에 대한 연구도 수행되고 있다.

Mahmud et al. [4]은 환경에 배치된 포그 노드(fog node)의 기능을 활용하여 사용자에게 더욱 짧은 지연 시간을 가지는 어플리케이션의 제공을 위해 어플리케이션을 모듈 단위로 나누어 배포하는 방법을 제안하였다. 이 때 모듈 단위로 배포한 이유는 포그 노드의 성능이 기존 서버에 비해서 좋지 않아, 포그 노드를 활용하기 위해서는 어플리케이션 보다 더 작은 단위의 상호의존적인 모듈로 나누어야 했기 때문이다. 이 연구에서는 서비스 제공 데드라인을 만족시켜 어플리케이션의 Quality of Service (QoS)를 보장하고 포그 노드의 자원 활용을 최적화하는 모듈 배포 방안을 제안하였는데, 본 연구와는 달리 어플리케이션 제공자의 관점에서 배포 비용을 고려하진 않았다.

Hong et al. [5]은 [4]와 유사하게 포그 환경에서 모듈로 분할된 어플리케이션을 활용 가능한 장치에 배치하는 방법에 대한 연구이다. 이 연구에서는 성공적으로 수행된 사용자의 요청의 수를 최대화하는 것을 목표로 하였으며, 각 요청에 필요한 모듈이 적절한 장비에 배치되었을 경우 해당 요청을 성공적으로 수행했다고 평가하였다. 이 연구에서는 이러한 최적화 문제를 풀기 위하여 휴리스틱 알고리즘을 제안하였는데, 해당 알고리즘은 상용 최적화 소프트웨어와 비교하였을 때 같은 최적 배치 결과를 상대적으로 짧은 시간에 출력할 수 있었다. 그러나 이 연구에서는 사용자가 모든 장비에 직접적으로 접근할 수 있다는 가정을 하여 네트워크상의 복잡도를 완화한 접근 방법을 제시하였다는 차이가 있었으며, 서비스 활용시에 발생하는 지연시간보다 모듈 배치에 걸리는 시간에 더 집중하여 분석하였다는 차이점이 있다.

Velasquez et al. [6]은 서비스 이미지와 사용자의 위치간 hop 수와 통신 노드간의 hop 수를 최소화하고, 서비스 배치 과정에서의 서비스 마이그레이션(service migration)의 수를 최소화함으로써 지연 시간을 최소화하는 서비스 배치 방법을 제시하였다. [4]와 유사하게 서비스의 배포 비용을 고려하진 않았으나, 사용자와 서비스 이미지 간의 통신경로에 따라 발생하는 서비스 제공 지연 시간을 최소화한다는 점에서 본 연구와 유사하다고 할 수 있다.

Arkian et al. [7]은 포그 환경에서 클라우드센싱으로 수집된 데이터 분석을 위한 가상머신의 배치 방법에 관한 연구이다. 이 연구에서는 사용자로부터 데이터를 수집하고 수집한 데이터를 분석하기 위한 가상머신을 배치할 때 발생하는 비용을 최소화하기 위하여 mixed-integer linear programming 문제로 정의하였다. 해당 연구에서도 서비스 지연시간을 서비스 배치 과정에서 제약사항으로서 고려하긴 하나, 본 연구에서는 서비스 지연 시간 최소화를 최적화 목표로 한다는 점이 가장 큰 차이라고 할 수 있다.

3. 문제 정의

3장에서는 문제 정의에 필요한 엣지 클라우드 환경 및 마이크로서비스 관련 정의와 최적화 목표 및 제약 사항에 관하여 설명한다. 아래 설명은 [3]의 문제 정의 내용을 보충하여 설명한 내용이다.

3.1. 엣지 클라우드 환경 및 마이크로서비스 명세 요소

표 1. 엣지 클라우드 환경 명세 요소

표기	설명
E	엣지 서버 집합
U	사용자 집합
$sr_i, i \in E$	기준 엣지 서버 대비 컴퓨팅 성능 비율
$rc_i, i \in E$	컴퓨팅 자원 용량
$ct_i, i \in E$	단위 컴퓨팅 자원당 비용
$tr_{ii}, i, i' \in E$	엣지 서버 사이의 데이터 전송률
$tr_{ji}, j \in U, i \in E$	사용자와 엣지 서버 사이의 데이터 전송률
$y_{ji}, j \in U, i \in E$	사용자와 엣지 서버 사이의 통신 유형, 0 또는 1

첫번째로, 엣지 클라우드 환경에 대한 정의가 필요하다. 표 1은 엣지 클라우드 환경에서 엣지 서버가 갖는 특성들을 정리한 표이다. 엣지 클라우드 환경에서 엣지 서버 i 는 (sr_i, rc_i, ct_i) 로 표현할 수 있다. 이 때 단위 컴퓨팅 자원당 비용인 ct_i 는 서버의 성능 sr_i 에 비례한다고 가정한다. 서버의 성능은 기준 엣지 서버 대비 컴퓨팅 성능의 비율로, 해당 값이 1일 경우 각 마이크로서비스가 정해진 시간만큼 동작한다는 것으로 가정하였다. 또한 엣지 서버의 연산력 등을 컴퓨팅 자원 용량 수치 $(rc_i, i \in E)$ 로 추상화하여 표현하였다. 하나의 엣지 서버에는 한정된 양의 컴퓨팅 자원이 있어, 마이크로서비스가 요구하는 컴퓨팅 자원의 합이 엣지 서버의 컴퓨팅 자원 용량을 초과해서는 안된다고 가정한다.

엣지 서버 간의 연결상태와 엣지 서버와 사용자 간의 연결상태는 추상화하여 표현하였다. 엣지 서버간 연결상태는 해당 연결에서 평균적으로 측정되는 데이터 전송률 $(tr_{ii}, i, i' \in E)$ 로 나타낼 수 있으며, 해당 수치는 서비스 제공 과정에서 변화하지 않는다고 가정하였다.

엣지 컴퓨팅 환경에서 사용자는 엣지 서버와 코어 네트워크를 통해 간접적으로 통신하거나, 무선 접속 네트워크를 통해 직접적으로 통신할 수 있다. 이 사용자와 엣지 서버 사이의 통신 유형을 추상화하여 $y_{ji}, j \in U, i \in E$ 라 정의하였으며, 이 값은 직접적으로 연결되었을 때 1, 간접적으로 연결되었을 때 0으로

표현하였다. 엣지 서버간 연결 상태와 유사하게, 엣지 서버와 사용자 간의 통신 상태도 해당 연결에서 평균적으로 측정되는 데이터 전송률 $(tr_{ij}, j \in U, i \in E)$ 로 정의하였으며, 해당 수치가 서비스 제공 과정에서 변화하지 않는다고 가정하였다.

표 2. 마이크로서비스 명세 요소

표기	설명
S	마이크로서비스 이미지 집합
$st_k, k \in S$	기준 엣지 서버에서의 서비스 시간
$Rq_k, k \in S$	컴퓨팅 자원 요구사항
$req_k, k \in S$	요청 메시지의 데이터 크기
$res_k, k \in S$	응답 메시지의 데이터 크기
$x_{ki}, k \in S, i \in E$	마이크로서비스 배포 상태, 0 또는 1

두번째로, 마이크로서비스 관련 정의가 필요하다. 표 2는 엣지 클라우드 환경에서의 마이크로서비스가 갖는 특성들을 정리한 표이다.

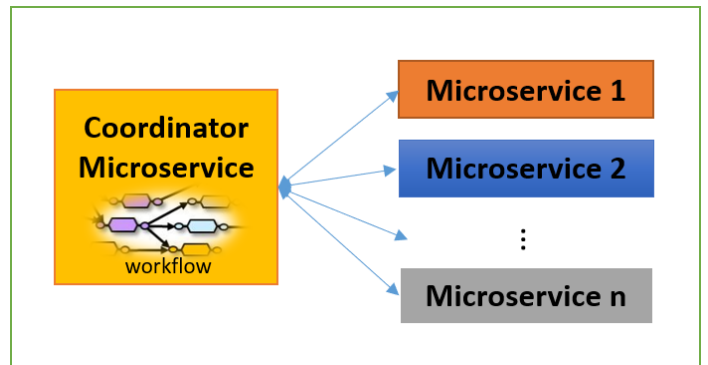


그림 2. MSA 어플리케이션의 구조

본 연구에서는 MSA 어플리케이션을 그림 2와 같이 가정하였다. 사용자와 직접 상호작용을 하는 코디네이터 마이크로서비스가 존재하며, 코디네이터 마이크로서비스는 해당 어플리케이션의 워크플로우에 따라 어플리케이션 내의 다른 마이크로서비스의 수행을 관리한다. 코디네이터 마이크로서비스는 다른 서비스가 필요로 하는 데이터를 전송하며, 해당 서비스의 출력을 받아와 가공 후 사용자에게 전달한다.

또한 마이크로서비스는 기준 엣지 서버에서의 서비스 시간 $(st_k, k \in S)$ 이 정해져있다고 정의하며, 이는 엣지 서버의 성능에 따라 실제 서비스 시간이 달라질 수 있다. 또한 컴퓨팅 자원 요구사항 $(Rq_k, k \in S)$ 은 마이크로서비스가 엣지 서버에 배치되었을 때 점유하는 자원의 양을 추상화한 것으로 서버의 컴퓨팅 자원 용량의 수치를 넘어설 경우 서버에 배치할 수 없다.

마이크로서비스는 요청 메시지를 받은 후에 작동하며, 그 결과를 사용자나 다른 마이크로서비스에 보내게 되는데 이 때 주고받는 요청 메시지, 응답 메시지의

크기 ($req_k, res_k, k \in S$)는 지연 시간 계산에 꼭 필요한 수치이기 때문에 추상화하여 정의하였다.

마지막으로 마이크로서비스의 배포 상태는 해당 마이크로서비스가 특정 엷지 서버에 배치되었는지를 나타내는 지표이다. 마이크로서비스 k 가 엷지 서버 i 에 배포되었을 때 해당 배포 상태 x_{ki} 는 1이 되며, 그렇지 않을 경우는 0이 된다.

위에 주어진 명세만으로 마이크로서비스 배포 및 실행이 되려면 다음과 같은 가정이 추가적으로 필요하다. 우선 사용자는 마이크로서비스를 따로 발견할 필요 없이 항상 필요한 서비스 중 최소 응답 시간을 가지는 마이크로서비스를 선택하여 사용할 것이라고 가정한다. 또 다른 가정은 다수의 사용자가 특정 서버에 배치된 하나의 서비스 인스턴스를 사용할 때 컴퓨팅 자원 요구사항에 이미 그러한 로드가 고려 되어있어, 추가적인 서버 부하가 발생하지 않는다는 것이다. 이러한 가정 하에서는 하나의 서버 안에 같은 마이크로서비스 인스턴스 생성으로 인한 배포 비용 상승을 고려하지 않아도 된다. 마지막으로 본 연구에서 가정하는 환경에는 문제 단순화를 위하여 하나의 MSA 어플리케이션과 해당 어플리케이션을 구성하는 마이크로서비스만 고려한다는 점이다.

위와 같은 명세 요소를 정의하였을 때 사용자 j 의 MSA 어플리케이션에 대한 평균 응답 시간은 다음과 같다.

$$R_j = \frac{1}{|S|} \sum_{k \in S} \min_{i \in E_g, i^* \in E_k} \left(\frac{req_k}{tr_{ji}} + \frac{req_k}{tr_{i^*i}} + \frac{res_k}{tr_{i^*i}} + \frac{res_k}{tr_{ji}} \right) + \left(\frac{st_g}{sr_i} + \frac{st_k}{sr_{i^*}} \right) \quad (1)$$

such that $j \in U, E_k = \{i | i \in E \wedge x_{ki} = 1\}$

평균 응답시간 R_j 는 사용자와 코디네이터 마이크로서비스, 그리고 나머지 마이크로서비스 간의 데이터 전송에 소모된 시간과 서비스 수행에 걸리는 시간의 합이다. 데이터 전송에 소모되는 시간은 전송되는 메시지 크기를 데이터 전송률로 나눈 값으로 계산하며, 서비스 수행에 걸리는 시간은 해당 마이크로서비스의 기준 서비스 시간을 엷지 서버의 컴퓨팅 성능 비율로 나누어서 계산한다.

그림 3은 식 (1)의 각 항이 실제 마이크로서비스 처리 과정에서 어떤 부분에서 발생하는지에 대하여 설명한 그림이다. 실제 환경에서는 MSA 어플리케이션을 구성하는 마이크로서비스가 1개 이상 배포될 수 있으므로 MSA 어플리케이션 작동 과정의 경로는 여러 개가 생길 수 있다. 이때 이미 언급한 가정대로, 가장 지연시간이 덜 걸리는 경로를 선택해서 서비스를 받는다고 가정한다.

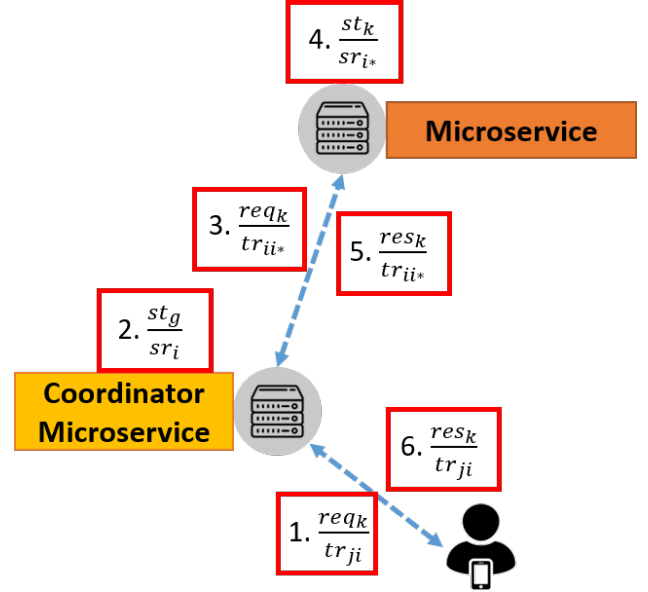


그림 3. 사용자 j 의 MSA 어플리케이션에 대한 평균 응답 시간 계산 과정의 도식화

마이크로서비스의 배치가 구성되었을 때, 배치 비용의 총합은 식 (2)와 같이 계산할 수 있다.

$$Cost_{msa} = \sum_{k \in S} \sum_{i \in E} x_{ki} \times Rq_k \times ct_i \quad (2)$$

마이크로서비스 k 가 엷지 서버 i 에 배치되었을 때 엷지 서버의 단위 컴퓨팅 자원당 비용과 마이크로서비스의 자원 요구량의 곱을 모든 k, i 의 페어에 대해 계산하여 합한 식이다.

3.2. 배포 비용 및 지연시간을 최소화하는 배포 상태 최적화 문제

본 연구에서는 배포 비용 최소화와 사용자의 평균 지연시간을 최소화를 목표로 삼는 마이크로서비스 배치 상태 x_{ki} 를 찾는 최적화 문제를 정의한다.

배포 비용 최소화 목표는 다음과 같이 정의할 수 있다.

$$\text{Minimize Cost}_{msa} = \sum_{k \in S} \sum_{i \in E} x_{ki} \times r q_k \times ct_i \quad (3)$$

또한 각 사용자의 평균 지연시간 최소화는 다음과 같이 정의한다.

$$\text{Minimize } \frac{1}{|U|} \sum_{j \in U} R_j \quad (4)$$

이 최적화 문제에서는 위의 두 목표와 함께 만족시켜야 하는 제약사항들이 3개 존재한다. 첫째로, MSA 어플리케이션을 구성하는 모든 마이크로서비스들은 최소 1개 이상 배포 되어야만 한다.

$$\sum_{i \in E} x_{ki} \geq 1, \quad \forall k \in S \quad (5)$$

둘째로, 각 엷지 서버에 배포된 마이크로서비스들의 자원 요구량의 합이 해당 엷지 서버의 컴퓨팅 자원 용량을 초과해서는 안된다.

$$\sum_{i \in E} x_{ki} \times rq_k \leq rc_i, \quad \forall i \in E \quad (6)$$

마지막으로, MSA 어플리케이션의 전체 배포 비용은 어플리케이션 제공자의 총 배치 예산보다 적어야 한다. 총 배치 예산을 **CostLimit** 이라고 했을 때, 위 제약조건은 다음과 같다.

$$Cost_{msa} \leq CostLimit \quad (7)$$

다시 정리하자면, 본 연구에서 다루는 문제는 3개의 제약사항(식 (5), (6), (7))을 만족시키는 동시에 2개의 최적화 목표(식 (3), (4))에 맞춰 최적화된 서비스 배치 방법을 찾는 문제이다.

4. 접근 방법

본 연구에서는 총 3가지의 다목적 최적화 방법을 활용한 MSA 어플리케이션을 구성하는 마이크로서비스를 배포하는 방법을 비교 분석하였다. 본 연구에서 활용한 3가지의 다목적 최적화 방법은 다양한 분야에서 쓰이는 방법이며, 본 연구에서 이진 스트링(binary string)으로 표현되는 마이크로서비스 배치 상황을 나타내는 해를 직접적으로 사용할 수 있는 알고리즘이기 때문에 선택하였다. 마이크로서비스 배치 상황을 나타내는 해에 대해서는 4.4절에서 자세히 설명하였다.

4.1. Non-dominated Sorting Genetic Algorithm-II (NSGA-II)

NSGA-II 알고리즘은 기존의 NSGA 알고리즘에서 비지배 프론트 정렬방법(nondominated front sorting)을 수정하여 상대적으로 더 적은 연산 과정을 필요로 하고, 우수해(elitism)를 더 강조하며 공유 변수를 미리 선택할 필요가 없다는 장점을 가진다 [8]. NSGA-II는 자손 세대를 생성하는 과정에서 개체의 비지배 프론트의 차수가 높은 해를 다음 세대의 부모 개체로 선택하며, 차수가 같을 경우 개체간 군집 거리(Crowding Distance)를 계산하여 군집거리가 큰 개체를 다음 세대의 재생산을 위한 부모 개체로 선택한다 [9].

4.2. ϵ -Multi-Objective Evolutionary Algorithm (ϵ -MOEA)

ϵ -MOEA는 steady-state genetic algorithm에 기반한 MOEA로, 고정된 크기의 비지배 해들을 보관하면서 지배 관계에 있는 해와 ϵ -dominance의 개념으로

관리되는 아카이브에 있는 해로 새로운 해를 만드는 방식의 알고리즘이다 [10]. 이 때 기존 아카이브의 모든 해가 새로운 해를 ϵ -dominate 하지 못하는 경우 새로운 해에 의해 ϵ -dominate 된 해들을 아카이브에서 제거한다. 또한 기존 population의 해 중에서 새로운 해에 의해 지배당하는 해는 그 중에서 하나를 새로운 해로 바꾸고 다음 세대로 넘어가게 된다 [11].

4.3. Pareto Envelope-based Selection Algorithm (PESA2)

PESA2는 각각의 해에 대해 적합도를 계산하는 대신 최소 하나 이상의 해가 들어있는 하이퍼박스(hyperbox)에 대해 선택적으로 적합도를 계산하고, 적합도가 높은 하이퍼박스 안에 있는 해 중 무작위로 선택된 해를 선택하는 독특한 과정을 기본으로 한다. 이러한 해 선택 방법은 각각의 해에 대한 적합도를 계산하여 선택하는 것보다 파레토 프론트(Pareto front)에서 더 퍼져 있는 해들을 발견할 수 있다 [12].

4.4. 다목적 최적화 알고리즘 적용을 위한 해 설계 및 구현

본 연구에서는 3장에서 정의한 서비스 배치 최적화 문제를 MOEA framework [13]를 활용하여 구현하였으며, 3개의 다목적 최적화 알고리즘(NSGA-II, ϵ -MOEA, PESA2)은 MOEA framework에 구현되어 있는 버전을 수정 없이 사용하였다.

MOEA framework에서는 하나의 문제가 여러 개의 변수와 제약사항을 가질 수 있다고 정의되어 있으며, 각 변수들의 타입은 사용하는 알고리즘에 따라 다르나, 본 연구에서 사용하는 세 알고리즘은 같은 타입의 변수를 활용할 수 있다.

위의 세 알고리즘을 사용하기 위해서는 마이크로서비스의 배치 상태를 나타내는 해를 정의해야 하며, 본 연구에서는 3장에서 정의한 문제의 변수를 활용하여 해를 정의하였다. 본 연구에서는 마이크로서비스 k 가 엷지 서버 i 에 배포 되었는지의 여부를 x_{ki} 로 정의하였으며, 해당 변수는 마이크로서비스 k 가 엷지 서버 i 에 배포되었을 때 1, 그렇지 않은 경우에 0으로 정의하였다. 이 변수를 길이가 $|E|$ (엷지 서버의 수)인 $|S|$ (마이크로서비스 이미지의 수)개의 이진 스트링으로 나누어 MOEA framework의 변수로 사용하였다. 즉 각 마이크로서비스 이미지마다 특정 엷지 서버에 배치되었는지를 나타내는 이진 스트링으로 정의하였다.

마이크로서비스의 배치 상태를 나타내는 변수에 따라 NSGA-II, ϵ -MOEA, PESA2, 이 세 개의 알고리즘들이 생성된 해가 3개의 제약사항을 위반하였는지의 여부와, 2개의 최적화 목표를 얼마나 만족시키는지 평가할 수 있도록 평가 함수를 구현하여 실험하였다.

5. 실험 및 평가

4장에서 논의한 세 알고리즘을 비교실험 하기 위해 3장의 명세 요소들에 기반한 실험 변수들을 설정하였다.

표 3. 실험 변수 기본 설정값

실험 변수	설정값
$ E , U , S $	100, 500, 4
$sr_i, i \in E$	[0.5, 2]
$rc_i, i \in E$	[1, 6]
$ct_i, i \in E$	$\approx sr_i$
$tr_{ii}, i, i' \in E, i = i'$	$\approx \infty$ mbps
$tr_{ii}, i, i' \in E, i \neq i'$	[50mbps, 200mbps]
$tr_{ji}, j \in U, i \in E, y_{ji} = 1$	[1000mbps, 1500mbps]
$tr_{ji}, j \in U, i \in E, y_{ji} = 0$	[47mbps, 176mbps]
$p(y_{ji} = 1)$	0.6
$st_k, k \in S$	[10ms, 20ms]
$Rq_k, k \in S$	[1, 2]
$(req_k, res_k, k \in S)$	[256kb, 1024kb]
CostLimit	50, 70, 90

본 연구에서 정의하는 환경과 정확히 일치하는 실제 환경 데이터를 구할 수 없었기 때문에 위에 정의된 실험 변수들은 정의된 범위 안에서 무작위적으로 정해진 환경을 가정하였다. 사용자와 엣지 서버 사이의 직접, 간접 데이터 전송률은 각각 KT에서 제공하는 5G와 일반 인터넷 속도 범위를 기준으로 설정하였다.

본 실험은 Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz (8 CPUs), ~3.6GHz CPU와 16GB의 RAM을 장착한 데스크탑 PC에서 수행하였다.

5.1 총 배치 예산 변화에 따른 각 다목적 최적화 알고리즘의 출력 해 분포 비교

그림 4, 5, 6은 어플리케이션 제공자의 총 배치 예산을 각 50, 70, 90으로 증가시켰을 때, 각 다목적 최적화 알고리즘이 출력한 해를 총 비용을 y축, 평균 지연 시간을 x축으로 이루어진 2차원 평면에 나타낸 그래프이다.

다목적 최적화 알고리즘은 서로 트레이드오프(trade-off) 관계에 있는 최적화 목표들이 존재할 경우 단 하나의 최적해를 생성하는 것이 아니라, 파레토 프론트에 퍼져 있는 여러 개의 비지배 해들을 생성한다. 이러한 비지배 해들이 얼마나 실제 파레토 프론트에 가까운지가 비지배 해의 최적도를 나타낼 수 있다.

또한 이렇게 생성된 비지배 해들은 트레이드오프 관계에 있는 최적화 목표의 우선순위가 정해져 있지 않는 한 하나의 해가 다른 해보다 더 우월하다고 볼 수 없다. 그러므로 다목적 최적화 알고리즘이 더 넓은 분포의 비지배 해들을 출력할 경우 사용자에게 더

다양한 선택지의 해를 제공하게 된다. 따라 다목적 최적화 알고리즘의 성능을 비교할 때, 비지배 해의 최적도뿐만이 아니라 각 알고리즘이 생성한 해가 얼마나 넓게 분포되어 있는지도 중요한 기준이 된다.

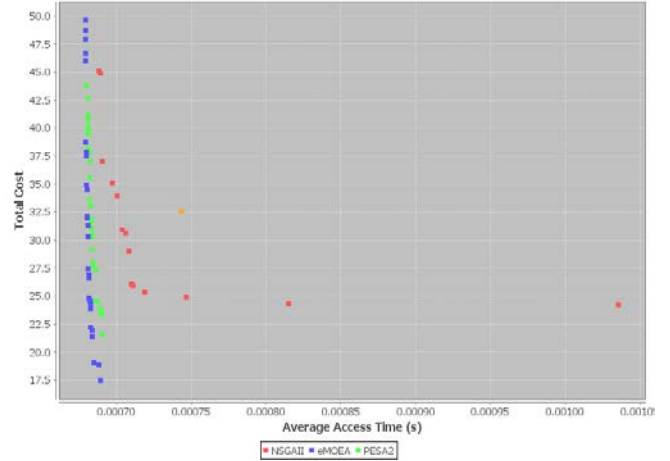


그림 7. 총 배치 예산이 50일 때 각 알고리즘의 출력 해의 분포

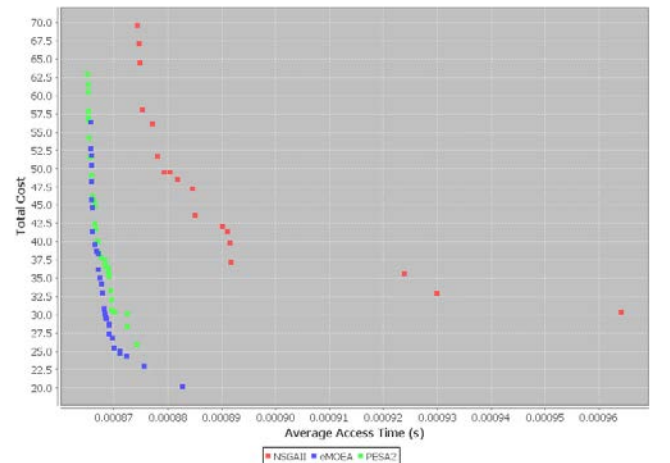


그림 8. 총 배치 예산이 70일 때 각 알고리즘의 출력 해의 분포

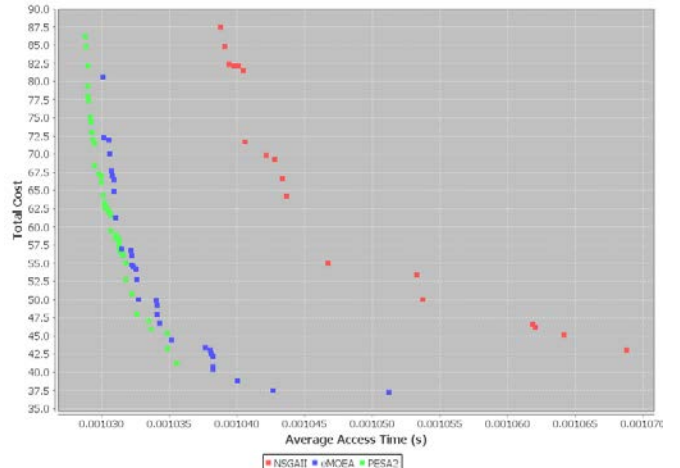


그림 9. 총 배치 예산이 90일 때 각 알고리즘의 출력 해의 분포

위 3개의 그래프를 비교해 보았을 때, ϵ -MOEA와 PESA2는 상대적으로 비슷한 분포의 해를 찾아냈으나, NSGA-II는 보다 넓은 분포의 해를 찾아냈다.

그러나 이러한 결과에서도 NSGA-II가 더 넓은 선택지를 제공한다고 보기는 어렵다. 그 이유는 NSGA-II의 해의 분포는 ϵ -MOEA, PESA2와 비교했을 때, 상대적으로 평균 지연 시간이 높은 분포의 해를 출력했으며, 총 배치 비용은 유사하게 나왔기 때문이다. 그러므로 평균 지연 시간을 최소화하는 최적화 목표를 고려했을 때, ϵ -MOEA, PESA2는 NSGA-II에 비해 상대적으로 더 최적해에 가까운 결과를 보여줬다고 할 수 있다. 각 알고리즘이 출력한 비지배 해 간의 격차는 배치 예산이 50에서 90으로 늘어났을 때 더욱 커졌다.

그에 비해 ϵ -MOEA, PESA2가 생성하는 비지배 해의 분포는 평균 지연 시간 최소화의 관점에서 봤을 때는 크지 않았다. 평균 지연 시간 최소화를 총 배치 비용 최소화의 관점보다 더 높은 우선순위로 둘 경우에는 두 알고리즘이 유사한 해를 출력한다고 볼 수 있다. 그러나 총 배치 비용 최소화를 최우선적인 목표로 한다고 봤을 때, ϵ -MOEA는 다른 알고리즘과 비교했을 때, 가장 적은 예산을 사용한 해를 출력하였다. 이러한 경향성은 총 사용 예산을 변화하였을 때도 관찰할 수 있었다.

5.2 총 배치 예산 변화에 따른 각 다목적 최적화 알고리즘의 실행 시간 분석

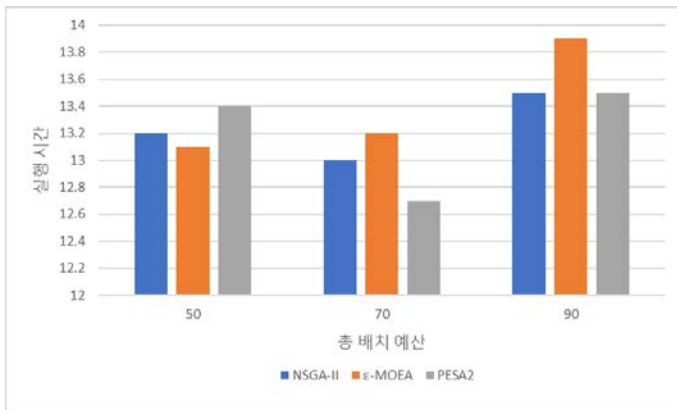


그림 10. 각 다목적 최적화 기법의 실행 시간

그림 11은 총 배치 예산을 50, 70, 90으로 변화하였을 때 각 알고리즘의 실행 시간을 측정하여 나타낸 그래프이다. NSGA-II, ϵ -MOEA, PESA2은 최대 30000번의 해 evaluation을 하도록 설정하였을 때, 평균적으로 약 13초 정도의 실행 시간이 나왔다. 이러한 실행 시간은 총 배치 예산의 증감에 따라 큰 차이가 발생하지 않았다. 예산을 50에서 90으로 증가시켰을 때, 실행시간이 평균적으로 0.3초 더 길어졌으나, 예산을 50에서 70으로 증가시켰을 때에는 오히려 0.2초 감소하는 결과를 보였다. 이러한 결과는 해를 계산하는 데에 있어 총 배치 예산의 증감으로 인한 Solution space의 변화보다도 무작위적으로 생성된

실험 변수의 영향이 더 크기 때문인 것으로 예상된다.

또한 각 알고리즘이 소모하는 시간의 차이도 크지 않았다. 각 알고리즘은 최대 약 0.5초의 차이가 나는 경우도 있었으나, 그러한 차이가 각 알고리즘간 차이가 무작위적으로 생성된 실험 변수에 따라 뒤바뀌는 경우도 발생하였으며 여러 차례 수행된 실험 과정에서 경향성을 보이지는 않았다.

각 알고리즘의 수행 시간이 유사한 이유는 각 알고리즘들이 수행 과정에서 생성된 해를 평가하기 위해 사용하는 평가 함수가 공통적이기 때문인 것으로 예상된다. 각 알고리즘 수행 과정에서의 세부적인 해 선택 및 진화 방법은 다르나, 그 과정에서 생성되는 해를 평가하는 과정에서 해당 해의 3개의 제약사항 위반 여부와 2개의 최적화 값을 계산하는 함수를 공통적으로 사용한다. 이 때문에 각 알고리즘의 수행 과정에서 공통적인 평가 함수를 자주 호출하게 되며 이 함수들의 수행 시간이 알고리즘 수행 시간의 대부분을 차지하게 되었고, 각 알고리즘의 수행 시간이 유사하게 나왔을 것으로 예상된다.

6. 결론

본 연구에서는 엣지 클라우드 환경에서 다수의 마이크로서비스로 구성된 마이크로서비스 아키텍처 기반 애플리케이션을 제공하는 과정에서 배포 비용 및 평균 지연시간을 최소화하기 위한 마이크로서비스 배포 방법을 제안하였다. 본 연구가 기반한 기존 연구[3]에서 평균 지연시간을 최소화하는 것을 최적화 목적으로 삼았던 것에 비해, 본 연구에서는 이를 확장하여 배포 비용을 최소화하는 것 역시 최적화 목적으로 삼아 문제를 확장하였다.

본 연구에서는 다목적 최적화 기법인 NSGA-II, ϵ -MOEA, PESA2 기법들을 활용하여 애플리케이션 제공자의 총 배치 예산의 변화에 따른 각 다목적 최적화 기법의 성능을 측정하였다. NSGA-II의 경우는 다른 두 기법에 비해 더 넓은 분포의 해를 생성하였으나, 상대적으로 최적화가 덜 된 해를 생성하였다. 세 기법 모두 실행 시간 상으로 큰 차이가 없었으며, 총 배치 예산의 변화에도 실행시간의 차이가 유의미한 변화가 있지는 않은 것으로 측정되었다.

향후에는 엣지 서버 네트워크 시뮬레이션 등을 활용하여 추상화된 실험 설정을 보다 현실적으로 만들어 실험하고, 다종의 애플리케이션을 제공하는 다수의 제공자를 가정하여 보다 복잡한 상황에서의 마이크로서비스 배치 실험을 할 계획이다.

Acknowledgement

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음(IIIP-2021-2020-0-01795). 또한 이 성과는 정부(과학기술정보통신부)의 재원으로

한국연구재단의 지원을 받아 수행된 연구임(No. 2019R1A2C1087430).

참고 문헌

- [1] P. G. Lopez et al., "Edge-centric computing: Vision and challenges", ACM SIGCOMM Comput. Commun. Rev., vol. 45, no. 5, pp. 37-42, Oct. 2015.
- [2] A. Balalaie, A. Heydarnoori and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," IEEE Software, vol. 33, no. 3, pp. 42-52, May-June 2016.
- [3] 손정연, 고인영. 엣지 컴퓨팅 환경에서 비용을 고려한 마이크로서비스 기반 어플리케이션 배포 방법. 2020 한국컴퓨터종합학술대회 논문집, Online, 2020년 7월 2일~4일
- [4] Mahmud, Redowan, Kotagiri Ramamohanarao, and Rajkumar Buyya. "Latency-aware application module management for fog computing environments." ACM Transactions on Internet Technology (TOIT) 19.1 (2018): 1-21.
- [5] Hong, Hua-Jun, Pei-Hsuan Tsai, and Cheng-Hsin Hsu. "Dynamic module deployment in a fog computing platform." 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, 2016.
- [6] Velasquez, Karima, et al. "Service placement for latency reduction in the internet of things." Annals of Telecommunications 72.1-2 (2017): 105-115.
- [7] Arkian, Hamid Reza, Abolfazl Diyanat, and Atefe Pourkhalili. "MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications." Journal of Network and Computer Applications 82 (2017): 152-165.
- [8] 박순규, 이수복, 이원철. "다목적 최적화를 위한 Goal-Pareto 기반의 NSGA-II 알고리즘." 한국통신학회논문지 32.11 (2007): 1079-1085.
- [9] Deb, Kalyanmoy, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." IEEE transactions on evolutionary computation 6.2 (2002): 182-197.
- [10] Mishra, Sudhansu Kumar, et al. "A fast multiobjective evolutionary algorithm for finding wellspread pareto-optimal solutions." In KanGAL Report No. 2003002, Indian Institute of Technology Kanpur. 2002.
- [11] 이인희, 신수용, 장병탁. "입실론-다중 목적함수 진화 알고리즘에 대한 비교 연구." 한국정보과학회 학술발표논문집 31.2 I (2004): 241-243.
- [12] Corne, D. W., N. R. Jerram, J. D. Knowles, and M. J. Oates (2001). PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. Proceedings

of the Genetic and Evolutionary Computation Conference (GECCO 2001), pp. 283-290.

[13] Hadka, Dave. "MOEA framework." Available at: <http://moeaframework.org> (2017).

소프트웨어 결함 예측을 위한 설명 가능한 인공지능 적용

최지원[○], 이재욱, 류덕산*, 김순태

전북대학교 소프트웨어공학과

{jiwon.choi, aewexpress01, duksan.ryu, stkim}@jbnu.ac.kr

The application of Explainable AI for Software Defect Prediction

Jiwon Choi[○], Jaewook Lee, Duksan Ryu*, Suntae Kim

Department of Software Engineering, Jeonbuk National University

요 약

소프트웨어 결함 예측(SDP)은 결함 모듈을 식별해 한정된 품질 보증 자원을 효과적으로 배분하는 것을 목표로 한다. SDP 성능을 향상시키기 위해 점점 더 복잡한 기계 학습 및 딥러닝 모델이 개발됐다. 이 모델들은 기존의 기법 대비 높은 성능을 보이지만, 블랙박스 형태의 모델로 해석 가능성이 저하된다는 한계를 가지고 있다. XAI(eXplainable Artificial Intelligence) 기법은 모델이 결과를 도출한 이유에 대해 설명함으로써 모델의 신뢰와 투명성을 향상시킬 수 있다. 본 연구는 SDP 연구에서 많이 사용되는 랜덤 포레스트 모델에 XAI 기법을 적용하고, 개별 특징과 결함과의 관계성을 연결시켜 설명하는 것을 목표로 한다. 이를 위해, 본 연구에서는 SHapley Additive exPlanations(SHAP)과 moDel Agnostic Language for Exploration and eXplanation(DALEX) 기법을 사용한다. 실험 결과, 모델이 중요하다고 판단한 특징이 두 XAI 기법에 따라 다소 차이가 있음을 보였다. XAI가 도출한 설명력이 높은 특징만을 사용해 모델을 학습하고 성능을 확인한 결과, 모든 특징들을 사용했을 때와 특징 선택 기법을 사용했을 때 우수한 성능을 보였다. XAI 기법을 활용해 SDP 모델의 결과를 설명함으로써 현업에서 소프트웨어 결함 예측 모델이 보다 많이 활용되길 기대한다.

1. 서 론

소프트웨어 결함 예측(Software Defect Prediction, SDP)은 소프트웨어 시스템에서 결함도가 높은 모듈을 보다 많이 식별해 한정된 품질 보증 자원을 효과적으로 배분할 수 있도록 돕는다[1]. SDP 성능을 향상시키기 위해 다양한 인공지능(Artificial Intelligence, AI) 기법이 활용되었다[12,13]. 특히 Tree 기반의 앙상블 모델 중 랜덤 포레스트(Random Forest, RF)는 우수한 성능을 보였다[2].

기계 학습 모델과 같은 AI 기법은 우수한 예측 성능을 보이지만, 모델이 발전함에 따라 모델이 복잡해지고 있다. 이는 입력 데이터와 모델 결과 간의 관계를 분석하기 어렵게 만든다[3]. 최근, 모델이 도출한 결과와 입력이 매핑 되는 방식을 연구하고, 결과를 설명할 수 있도록 도와주는 XAI(eXplainable Artificial Intelligence)가 제안되었다[6]. XAI는 모델이 도출한 결과에 어떤 특징이 영향을 주었는지 설명해주는

기법이다.

본 연구는 RF 모델에 XAI 기법을 적용한다. RF는 SDP 연구에서 많이 사용되는 앙상블 모델이며[15], XAI 기법은 SHapley Additive exPlanations(SHAP)[4]와 최근에 제안된 moDel Agnostic Language for Exploration and eXplanation(DALEX)[5]을 사용한다. 특히, SHAP은 XAI에 대표적인 기법 중 하나이며, 두 XAI 기법을 통해 특징과 소프트웨어 결함의 관계를 설명한다. 또한, 두 기법은 전역 설명(Global explanation)과 지역 설명(Local explanation)을 지원하는 모델이고, 본 연구에서는 SDP 데이터의 특징과 결함과의 관계를 보는 전역 설명만을 대상으로 한다. 두개의 XAI 기법을 통해 RF가 중요하다고 판단한 SDP 데이터 특징을 확인한다. 실험 결과, SHAP와 DALEX가 모델이 중요하다고 판단한 특징이 다소 차이가 있음을 확인할 수 있었다. 또한, RF 모델이 중요하다고 본 특징만을 선별해 성능을 확인한다. 그 결과, RF가 중점적으로 본 특징만을 사용했을 때 보다 모든 특징을 사용한 경우와 특징 선택 기법을 적용했을 때 우수한 성능을 보였다. 본 연구 결과를 통해 RF 모델의 결과를 설명할 수 있고, 여러가지 XAI 기법을 적용해 모델의 결과를 해석해야 함을 확인할 수 있었다.

* 교신 저자(Corresponding Author). 본 연구는 원자력안전위원회의 지원으로 한국원자력안전재단의 지원을 받아 수행한 원자력안전연구사업(No. 2105030)과 정부(과학기술정보통신부)의 지원으로 한국연구재단의 지원(NRF-2019R1G1A1005047)을 받아 수행된 연구사업의 결과임

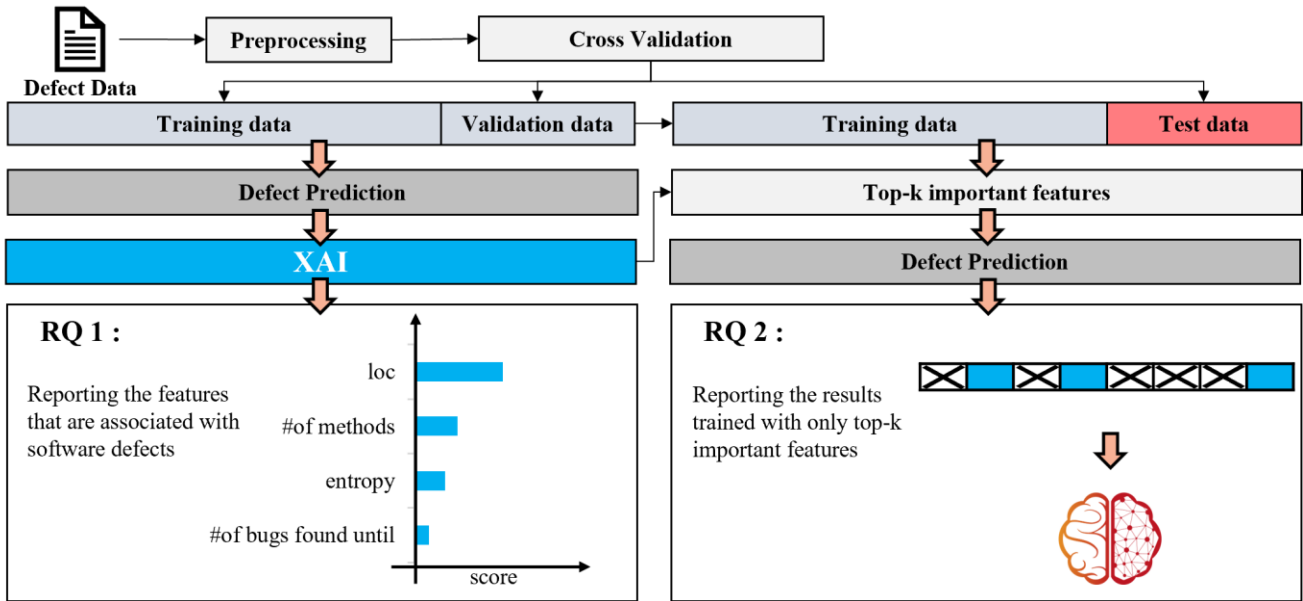


그림 1. 전체 프레임워크

2. 관련 연구

이 섹션에서는 XAI가 SDP에 적용된 관련 연구에 관해 기술한다.

최근 SDP 연구에서 XAI를 적용하는 시도가 많아지고 있다. Wattanakriengkrai et al.[11] 는 XAI 기법을 활용해 결함이 있는 코드 라인을 식별하고 우선 순위를 지정하는 프레임워크를 제안한다. 그들은 제안한 기법이 검사 노력 줄일 수 있음을 실험적으로 보였다. Shin et al.[10]는 지역 설명을 하는 XAI 기법을 대상으로 샘플링 기법과 기계 학습 분류기 등 다양한 시나리오에서 XAI 기법을 적용한다. Jiarpakdee et al.[9] 연구는 XAI 기법을 통해 결함 예측 모델의 결과를 설명하는 것은 개발자가 예측 결과를 이해하는데 도움이 된다고 강조했다.

대부분의 관련 연구는 지역 설명을 하는 XAI 기법을 다양한 SDP 환경에 적용한다. 지역 설명은 인스턴스 별 예측에 관해 설명하지만, 전역 설명은 모델이 도출한 전체 예측에 관해 설명한다. 본 연구에서는 전역 설명을 하는 XAI 기법을 대상으로 SDP에서 자주 사용되는 기계 학습 분류기에 적용한다.

3. 연구 방법

이 섹션에서는 우리가 수행한 연구 방법에 대해 상세히 기술한다. 그림 1은 본 연구에서 수행한 전체적인 접근 방법을 나타낸다.

우리는 전처리 단계에서 최소-최대 정규화를 진행하고, SMOTE를 사용해 클래스 불균형 문제를 해결한다. 대부분의 SDP 연구에서는 전처리 단계에서 정규화를 수행하고 클래스 불균형 문제를 해결한다[14]. 특히, SDP 데이터셋은 클래스 불균형 문제를 가지고 있기 때문에, 클래스 불균형 문제를 해결해주는 것은

매우 중요하다.

본 연구에서는 Stratified-10 fold 교차 검증을 수행한다. 섹션 4에서 수립한 연구 질문 첫번째를 수행하기 위해, 학습 데이터로 RF 모델을 학습시키고 검증 데이터로 RF 모델이 어떤 특징을 중요하게 판단하는지 확인한다.

표 1은 SHAP와 DALEX에 대한 설명이 기술되어 있으며, SHAP은 서로 다른 특성들의 조합을 고려한 Shapley 값을 토대로 특징의 중요도를 계산하고, DALEX는 손실 함수를 사용해 중요도를 계산한다.

연구 질문 두번째를 수행하기 위해, 첫번째 연구 질문에서 사용한 학습 데이터와 검증 데이터를 합쳐 새로운 학습 데이터로 만든다. 그 다음, 모델이 중요하다고 판단한 상위 특징 k개만을 사용해 새로운 RF 모델을 학습시키고 성능을 확인한다.

표 1. XAI 기법 비교

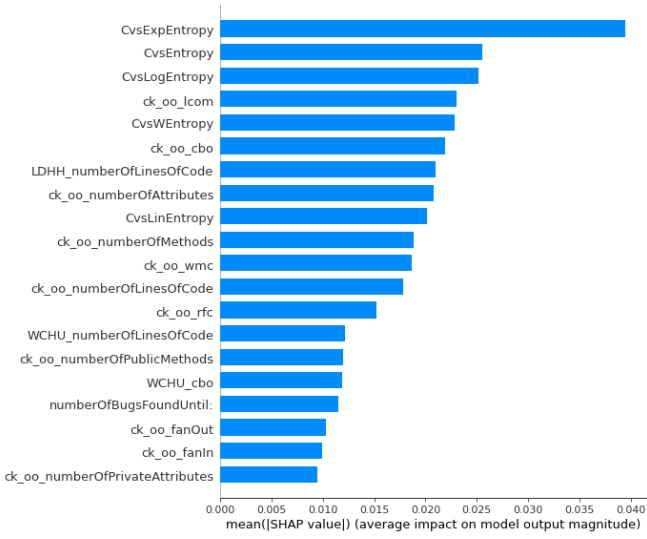
XAI measure	Author	Algorithm basis	API compatible
SHAP	[4]	Game Theory	Yes
DALEX	[5]	Leave-one covariate out	Yes

4. 실험 설정

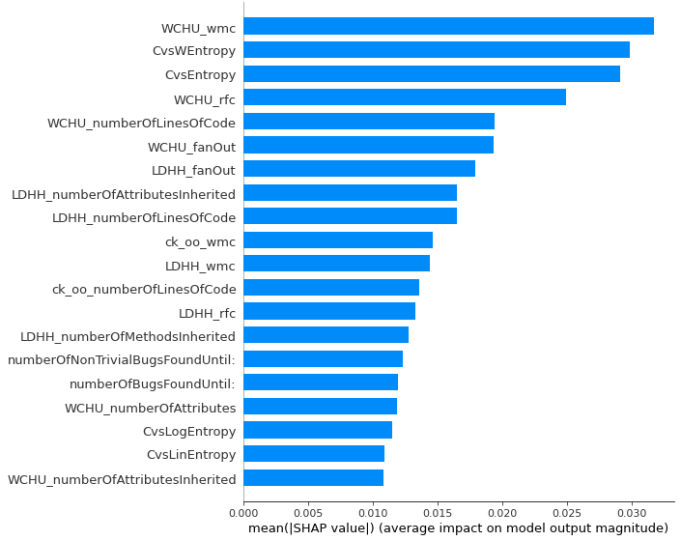
이 섹션에서는 연구 질문과 데이터셋, 평가 척도에 대한 내용을 기술한다. 본 연구의 실험 환경은 Python 3.7 버전과 RAM 16GB, 3.60GHz, Window OS를 이용하였다.

4.1 연구 질문

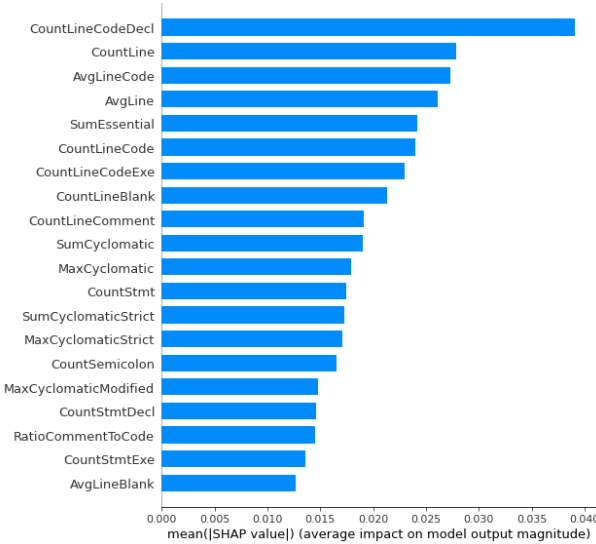
본 연구에서는 두가지의 연구 질문을 수립해 XAI



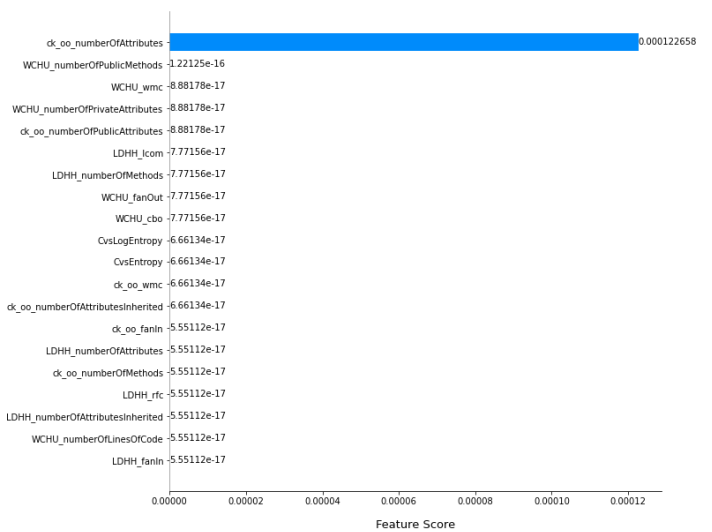
(a) EQ 프로젝트에 SHAP을 적용한 결과



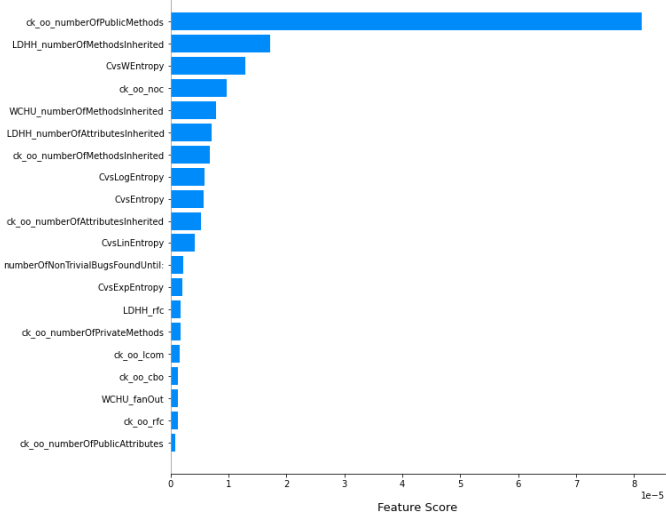
(b) JDT 프로젝트에 SHAP을 적용한 결과



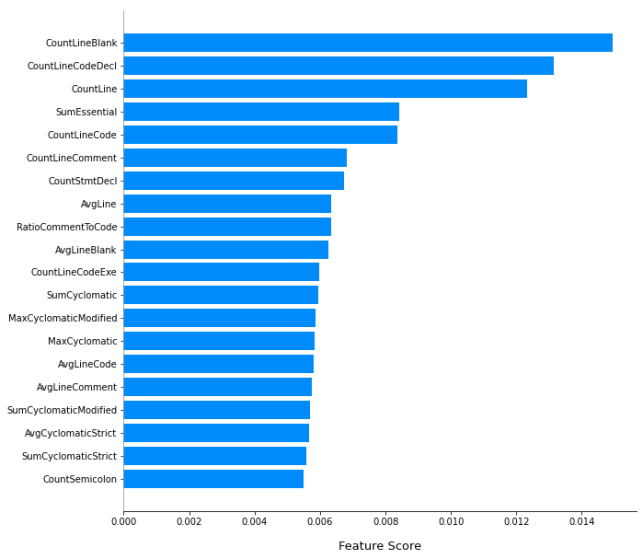
(c) ZXing 프로젝트에 SHAP을 적용한 결과



(d) EQ 프로젝트에 DALEX을 적용한 결과



(e) JDT 프로젝트에 DALEX을 적용한 결과



(f) ZXing 프로젝트에 DALEX을 적용한 결과

그림 2. SHAP과 DALEX를 적용한 결과

표 2 데이터의 전체 크기 및 결함 수

Dataset	Project	#of instances		#of metric	Granularity
		Total	Defect		
AEEEM	EQ	324	129(39.81%)	61	class
	JDT	997	206(20.66%)	61	class
	LC	691	64(9.26%)	61	class
	ML	1862	245(13.16%)	61	class
	PDE	1497	209(13.96%)	61	class
ReLink	Apache	194	98(50.52%)	26	file
	Safe	56	22(39.29%)	26	file
	ZXing	399	118(29.57%)	26	file

표 3. Confusion Matrix

Actual class	Predicted class		
		Defective	Clean
	Defective	TP(True Positive)	FN(False Negative)
Clean	FP(False Positive)	TN(True Negative)	

기법을 SDP에 적용한다.

RQ1. XAI 기법에 따라 모델이 중점적으로 본 특징이 동일한가?

H_{10} : XAI 기법에 따라 모델이 중점적으로 본 특징이 동일하다.

H_{1A} : XAI 기법에 따라 모델이 중점적으로 본 특징이 다르다.

RQ2. RQ1에서 확인한 결과를 토대로 모델이 중요하다고 본 특징이 성능에 영향을 미치는가?

H_{20} : 모델이 중요하다고 본 특징이 성능에 영향을 미친다.

H_{2A} : 모델이 중요하다고 본 특징이 성능에 영향을 미치지 않는다.

$$Recall = \frac{TP}{TP + FN} \tag{1}$$

$$PF = \frac{FP}{FP + TN} \tag{2}$$

$$Balance = 1 - \sqrt{\frac{(0 - PF)^2 + (1 - Recall)^2}{2}} \tag{3}$$

$$FI = \frac{(TP + FP)}{(TP + TN + FP + FN)} \tag{4}$$

$$FIR = \frac{Recall - FI}{Recall} \tag{5}$$

4.2 데이터셋

본 연구에서는 오픈 소스 데이터셋인 AEEEM과 ReLink를 사용한다. 두 데이터셋은 SDP 연구에서 많이 사용되는 데이터셋이다. 데이터에는 Line of Code(LOC)

와 Complexity 등의 소스코드 정적 메트릭 정보가 포함되어 있다. 데이터셋에 대한 상세한 설명은 표 2에 작성되어 있다.

4.3 평가 척도

본 연구에서는 평가 척도로 Recall(1)과 Probability of False Alarm(PF)(2), Balance(3), FIR(5)을 사용한다.

Recall은 올바르게 분류된 결함 모듈의 비율을 의미하며 [7], 1에 가까울수록 성능이 우수함을 나타낸다. PF는 잘못 분류된 비결함 모듈의 비율을 의미하며 [7], 0에 가까울수록 성능이 우수하다. Balance는 Recall과 PF를 모두 고려하는 성능 평가 지표로 1에 가까울수록 우수한 성능을 의미하고, SDP 연구에서 많이 사용되는 성능 평가 지표다. FIR은 동일한 Recall를 달성하기 위해 검사할 파일 수가 감소한 비율을 의미한다 [8]. FIR 성능이 1에 가까우면, 파일의 크기가 작을수록 결함을 잘 찾을 수 있다는 뜻으로 해석할 수 있다.

5. 실험 결과

이 섹션에서는 이전 섹션 4에서 수립한 연구 질문에 대한 실험 결과를 기술한다. 본 연구의 실험 결과는 Stratified 10-fold 교차 검증을 10번 수행한 성능의 평균값이다.

5.1 RQ1 실험 결과

이 연구 질문에서는 RF에 SHAP과 DALEX를 적용하고, 각 XAI 기법을 통해 RF가 중요하다고 본 특징을 확인한다.

그림 2의 (a), (b), (c)는 SHAP를 적용한 결과이고, (d), (e), (f)는 DALEX를 적용한 결과이다. 그래프가 길수록 모델이 중요하다고 판단한 특징임을 의미한다. 본 연구 질문에서 다루지 않은 다른 프로젝트의 실험 결과는 부록에 수록하였다.

그림 2의 (a)와 (b)를 확인한 결과, 'CvsEntropy' 특징이 상위 3번째 안에 공통적으로 포함됨을 확인할 수 있고, 이는 해당 특징이 중요하다고 해석할 수 있다. 해당 특징은 코드 변경의 엔트로피(Entropy of code changes)를 의미한다. 특히, (a)의 EQ 프로젝트는 상위 5개에 엔트로피 관련 특징이 4개가 포함되어 있다. ReLink 데이터셋인 ZXing 프로젝트에 SHAP을 적용한 결과는 그림 (c)에 나타나 있으며, 선언적 소스 코드를 포함하는 코드 라인 수(Number of lines containing declarative source code) 특징을 RF가 가장 중요한 특징으로 판단함을 확인할 수 있다.

그림 2의 (d), (e), (f)는 DALEX를 적용한 결과이다. (d)는 EQ 프로젝트에 적용한 결과이며, 'ck_oo_numberofAttributes' 특징을 제외한 나머지 피쳐는 값이 매우 작아, 숫자로 표현하였다. 마찬가지로 EQ 프로젝트에서는 속성의 수(Number of attributes) 특징을 나타내는 'ck_oo_numberofAttributes' 이 RF가

표 4. RQ2의 실험 결과

Method	Evaluation Metric	AEEEM	ReLink
All features	Recall	0.5149	0.5644
	PF	0.1206	0.3379
	Balance	0.6306	0.5878
	FIR	0.6454	0.2548
RFE – Top 20	Recall	0.5501	0.5895
	PF	0.1267	0.2074
	Balance	0.6608	0.6654
	FIR	0.6287	0.3801
SHAP – Top 20	Recall	0.4920	0.5645
	PF	0.1324	0.3492
	Balance	0.6150	0.5849
	FIR	0.6071	0.2417
DALEX – Top 20	Recall	0.4328	0.5830
	PF	0.1566	0.3730
	Balance	0.5765	0.5748
	FIR	0.5356	0.2400

가장 중요한 특징으로 판단했음을 알 수 있다. (e)는 JDT 프로젝트에 적용한 결과이며, public 메소드 수를 의미하는 ‘ck_oo_numberofAttributes’ 특징이 가장 중요한 특징으로 뽑혔다. (f)는 ZXing 프로젝트에 적용한 결과이며, 빈 줄(blank line)의 수를 나타내는 ‘CountLineBlank’ 특징을 RF가 중요하다고 판단함을 알 수 있다.

동일한 프로젝트에서 SHAP과 DALEX로 중요도가 높은 특징을 비교한 결과는 다음과 같다. EQ 프로젝트에서는 상위 10개의 특징을 비교한 결과, ‘CvsLogEntropy’와 ‘ck_oo_numberofAttributes’가 공통적으로 포함되었고, 그 외의 나머지 특징은 달랐다. JDT 프로젝트에서도 마찬가지로 상위 10개의 특징을 비교한 결과, ‘CvsWEntropy’와 ‘CvsEntropy’가 공통적으로 포함되어 있음을 확인할 수 있었다.

첫번째 연구 질문의 실험 결과를 통해 RF가 중요하다고 판단한 특징이 SHAP와 DALEX 기법에 따라 다소 차이가 있음을 확인할 수 있다. 본 실험 결과를 통해 귀무가설을 기각하고 대립가설을 채택할 수 있다.

5.2 RQ2 실험 결과

이 연구 질문의 목표는 모델이 중요하다고 본 특징들만을 선별하고 이들이 실제로 예측 성능에 영향을 주는지 확인하는 것이다. 또한, 특징 선택 기법으로 XAI 기법을 활용할 수 있는지 확인한다. 이를 위해, 본 연구 질문에서는 모든 특징들을 활용한 경우와 특징 선택을 적용한 경우, 각 XAI 기법을 통해 선별한 특징을 사용해 성능을 비교 실험한다.

표 4에서 ‘All features’는 모든 데이터의 특징으로 학습한 경우를 의미하며, ‘RFE’ 열은 특징 선택 기법인

‘Recursive Feature Elimination(RFE)’을 적용한 경우, SHAP을 통해 확인한 상위 20개의 특징으로 학습한 경우, DALEX를 통해 확인한 상위 20개의 특징으로 학습한 경우의 성능을 비교한 결과이다. 표에 작성된 성능은 각 데이터셋에 포함된 프로젝트들의 성능을 평균 낸 값이며, 굵은 글씨는 우수한 성능을 의미한다.

특징 선택은 SDP 연구에서 오버 피팅을 줄여주는 효과를 쥐 성능 향상에 기여하는 전처리 방법 중 하나이다[14]. 표 4에서 확인할 수 있듯이, AEEEM 데이터셋에서는 모든 특징을 사용한 경우에 Balance 성능이 우수하고, ReLink 데이터셋에서는 특징 선택을 수행한 결과가 모든 데이터셋에서 우수하다. 그 다음으로 RF, SHAP, DALEX 순이다. 이 실험 결과를 통해 RF 모델이 중요하다고 판단한 특징이 성능에 영향을 주지 않음을 확인할 수 있다.

XAI 기법을 통해 확인한 특징의 수를 5개, 10개, 20개로 설정하였을 때의 성능을 비교한다. 표 5는 그 결과이고, 표의 굵은 글씨는 가장 우수한 성능을 의미한다. 5개의 특징을 선택해 학습한 결과, Balance 성능 지표에서 SHAP이 8개 프로젝트 중 4개의 프로젝트에서 우수한 성능을 보였고, RFE와 DALEX는 3개의 프로젝트에서 우수한 성능을 보였다. 상위 10개의 특징을 선택해 학습한 결과, RFE, SHAP, DALEX 순으로 우수한 Balance 성능을 보였다. 상위 20개의 특징을 선택해 학습한 결과, 모든 프로젝트에서 RFE가 우수한 Balance 성능을 보였고, DALEX의 Balance 성능보다 SHAP 성능이 다소 높았다. 이 실험 결과를 통해 특징 수가 많아질수록 특징 선택의 성능이 우수함을 확인할 수 있었다. 또한 DALEX보다 SHAP이 우수한 Balance 성능을 나타낸다는 점에서 중요한

표 5. 상위 k 개의 특징들을 선택해 RF 모델을 학습한 결과

Method	Top- k	Evaluation Metric	AEEEM					ReLink		
			EQ	JDT	LC	ML	PDE	Apache	Safe	ZXing
SHAP	5	Recall	0.7821	0.5889	0.3333	0.3265	0.3875	0.5941	0.5000	0.3724
		PF	0.2622	0.1290	0.1113	0.0725	0.1038	0.5182	0.2833	0.1843
		Bal	0.7584	0.6952	0.5220	0.5210	0.5607	0.5344	0.5885	0.5373
		FIR	0.3783	0.6051	0.5942	0.6750	0.6333	0.0711	0.2167	0.3206
	10	Recall	0.7179	0.6511	0.3933	0.3796	0.3625	0.6647	0.5000	0.3759
		PF	0.2784	0.1336	0.0919	0.0685	0.1039	0.5000	0.3833	0.1117
		Bal	0.7196	0.7358	0.5661	0.5586	0.5432	0.5730	0.5526	0.5515
		FIR	0.3484	0.6160	0.6819	0.7116	0.6176	0.1387	0.1167	0.4473
	20	Recall	0.7107	0.6622	0.3333	0.4164	0.3375	0.6412	0.5834	0.4690
		PF	0.3297	0.1181	0.0831	0.0534	0.0777	0.4318	0.5000	0.1157
		Bal	0.6896	0.7468	0.5249	0.5856	0.5282	0.6017	0.5375	0.6154
		FIR	0.3050	0.6368	0.6698	0.7569	0.6672	0.1824	0.0625	0.4803
DALEX	5	Recall	0.5179	0.5800	0.3867	0.3571	0.3475	0.6706	0.6667	0.3828
		PF	0.3433	0.1697	0.1000	0.0821	0.0962	0.4682	0.5000	0.1117
		Bal	0.5811	0.6795	0.5605	0.5417	0.5336	0.5943	0.5751	0.5561
		FIR	0.1896	0.5480	0.6597	0.6681	0.6249	0.1695	0.1250	0.4516
	10	Recall	0.5000	0.5867	0.3933	0.3387	0.3350	0.6883	0.6667	0.3966
		PF	0.3460	0.1593	0.1008	0.0812	0.1015	0.4636	0.5000	0.1000
		Bal	0.5697	0.6867	0.5651	0.5289	0.5243	0.6041	0.5751	0.5672
		FIR	0.1718	0.5644	0.6626	0.6592	0.6016	0.1829	0.1250	0.4759
	20	Recall	0.5036	0.5867	0.4000	0.3489	0.3250	0.6824	0.6667	0.4000
		PF	0.3406	0.1645	0.1008	0.0805	0.0965	0.4955	0.5000	0.1235
		Bal	0.5736	0.6852	0.5698	0.5361	0.5178	0.5829	0.5751	0.5665
		FIR	0.1837	0.5568	0.6673	0.6669	0.6033	0.1543	0.1250	0.4407
RFE	5	Recall	0.705	0.6233	0.3413	0.4953	0.379	0.5882	0.2727	0.5172
		PF	0.2284	0.135	0.0944	0.1001	0.1665	0.4545	0.0833	0.1765
		Bal	0.7359	0.7169	0.5294	0.6361	0.5453	0.5663	0.4824	0.6365
		FIR	0.3846	0.607	0.6444	0.6926	0.4842	0.1282	0.3623	0.42
	10	Recall	0.6993	0.6882	0.3486	0.4033	0.3755	0.7059	0.2727	0.4138
		PF	0.2738	0.1253	0.0803	0.0774	0.1155	0.4091	0.25	0.1569
		Bal	0.7118	0.7623	0.5358	0.5745	0.5509	0.6437	0.4562	0.5709
		FIR	0.3461	0.6337	0.6856	0.7017	0.5984	0.2372	0.0434	0.3958
	20	Recall	0.7107	0.6882	0.4893	0.4376	0.425	0.7059	0.5455	0.5172
		PF	0.2754	0.1142	0.0822	0.0658	0.0962	0.3182	0.1667	0.1373
		Bal	0.7173	0.7651	0.6342	0.5996	0.5878	0.6936	0.6577	0.6451
		FIR	0.3486	0.6463	0.7403	0.7379	0.6706	0.3098	0.3624	0.4683

특징을 보다 잘 설명한다고 해석할 수 있다.

XAI 기법을 통해 확인한 상위 k 개의 중요한 특징 중 두 기법이 공통적으로 언급한 특징은 몇 개인지 분석한 결과는 다음과 같다. LC 프로젝트의 경우, 상위 5개의 중요한 특징 중 2개, 상위 10개 중 4개, 상위 20개 중 9개가 동일하다. Apache 프로젝트의 경우, 상위 5개의 중요한 특징 중 2개, 상위 10개 중 4개, 상위 20개 중 16개가 동일하다. AEEEM 데이터셋의 경우 총 61개의 특징으로 구성되어 있고, ReLink 데이터셋의 경우 총

26개의 특징으로 구성되어 있다. 따라서 상대적으로 ReLink 데이터셋이 상위 20개의 특징 중 공통된 특징을 분석했을 때, 많은 특징들이 공통됨을 확인할 수 있었다.

이 연구 질문의 실험 결과를 통해 모델이 중요하다고 본 특징이 성능에 영향을 준다고 보기 어려우며, 따라서 귀무가설을 기각하고, 대립가설을 채택할 수 있다.

6. 위협 요소

이 섹션에서는 본 연구의 위협요소에 대해 설명한다.

구성 위험요소는 성능 평가 지표를 Recall과 PF, Balance, FIR 총 4가지만 고려했다는 점이다. 본 연구에서 사용한 성능 평가 지표는 SDP 연구에서 자주 사용되는 평가 지표이다. 향후, 다른 성능 평가 지표를 고려할 계획이다.

내부 위험요소는 기계 학습 모델을 RF만 고려했다는 점이다. 이 위험을 완화시키기 위해 향후 다양한 기계학습 및 딥러닝 모델을 추가할 계획이다. 또한 다양한 XAI 기법 중 두가지의 기법만을 본 연구에서 적용하였다. 이 위험을 완화시키기 위해, 향후 추가적인 XAI 기법을 적용할 계획이다.

7. 결론 및 향후 연구

SDP는 테스트 단계 전에 결함 모듈을 식별함으로써 한정된 품질 보증 자원을 효과적으로 할당하도록 돕는다. SDP 모델의 성능 향상을 위해, 점차 복잡한 기계학습 및 딥러닝 모델이 제안되고 있다.

복잡한 모델은 입력 데이터와 모델의 출력 결과 간의 연결을 어렵게 만든다. 이러한 문제를 해결하기 위해 모델이 도출한 결과에 어떤 특징이 영향을 주었는지 설명해주는 XAI 기법이 제안되었다.

본 연구에서는 다양한 XAI 기법 중 SHAP와 DALEX를 RF 모델에 적용했다. 이후, 과거 소프트웨어 결함과 관련된 특성을 설명하고, RF 모델이 중요하다고 판단한 특징들만 이용해 성능을 확인했다. 실험 결과, 두 XAI 기법에 따라 RF 모델이 중요하다고 판단한 특징이 다소 차이가 있음을 보였다. 또한 모델이 중요하다고 판단한 특징만을 가지고 모델을 학습시킨 결과, 전체 특징을 사용했을 때와 다른 특징 선택 기법을 적용했을 때 다소 성능이 우수했다. 이 결과를 통해 RF가 중요하게 본 특징들이 결함 예측력 향상에 기여하는 바가 적음을 확인할 수 있었다.

향후, 다양한 XAI 기법과 데이터셋을 추가할 계획이다.

참고문헌

[1] Jing, X. Y., Wu, F., Dong, X., & Xu, B. (2016). An improved SDA based defect prediction framework for both within-project and cross-project class-imbalance problems. *IEEE Transactions on Software Engineering*, 43(4), 321–339.

[2] Aljamaan, H., & Alazba, A. (2020, November). Software defect prediction using tree-based ensembles. In *Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering* (pp. 1–10).

[3] Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access*, 6, 52138–52160.

[4] Lundberg, S. M., & Lee, S. I. (2017, December). A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 4768–4777).

[5] Baniecki, H., Kretowicz, W., Piatyszek, P., Wisniewski, J., & Biecek, P. (2020). dalex: Responsible Machine Learning with Interactive Explainability and Fairness in Python. *arXiv preprint arXiv:2012.14406*.

[6] Doran, D., Schulz, S., & Besold, T. R. (2017). What does explainable AI really mean? A new conceptualization of perspectives. *arXiv preprint arXiv:1710.00794*.

[7] Tantithamthavorn, C., McIntosh, S., Hassan, A. E., & Matsumoto, K. (2018). The impact of automated parameter optimization on defect prediction models. *IEEE Transactions on Software Engineering*, 45(7), 683–711.

[8] Shin, Y., Meneely, A., Williams, L., & Osborne, J. A. (2010). Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. *IEEE transactions on software engineering*, 37(6), 772–787.

[9] Jiarpakdee, J., Tantithamthavorn, C., Dam, H. K., & Grundy, J. (2020). An empirical study of model-agnostic techniques for defect prediction models. *IEEE Transactions on Software Engineering*.

[10] Shin, J., Aleithan, R., Nam, J., Wang, J., & Wang, S. (2021). Explainable Software Defect Prediction: Are We There Yet?. *arXiv preprint arXiv:2111.10901*.

[11] Wattanakriengkrai, S., Thongtanunam, P., Tantithamthavorn, C., Hata, H., & Matsumoto, K. (2020). Predicting defective lines using a model-agnostic technique. *arXiv preprint arXiv:2009.03612*.

[12] Shi, K., Lu, Y., Liu, G., Wei, Z., & Chang, J. (2021). MPT-embedding: An unsupervised representation learning of code for software defect prediction. *Journal of Software: Evolution and Process*, 33(4), e2330.

[13] Zheng, J. (2010). Cost-sensitive boosting neural networks for software defect prediction. *Expert Systems with Applications*, 37(6), 4537–4543.

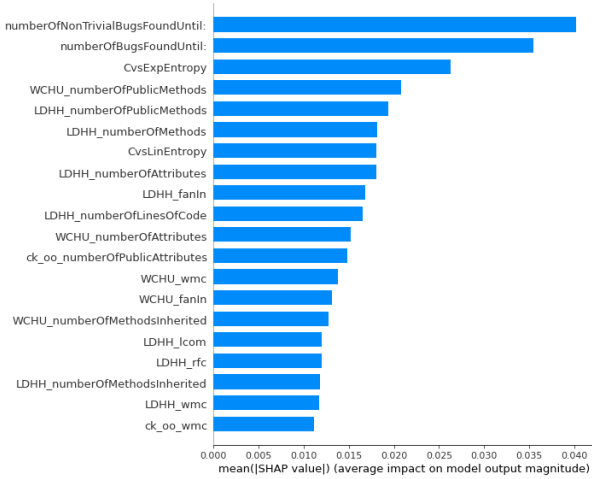
[14] Wang, H., Khoshgoftaar, T. M., & Napolitano, A. (2010, December). A comparative study of ensemble feature selection techniques for software defect prediction. In *2010 Ninth International Conference on Machine Learning and Applications* (pp. 135–140). IEEE.

[15] Matloob, F., Ghazal, T. M., Taleb, N., Aftab, S., Ahmad, M., Khan, M. A., ... & Soomro, T. R. (2021). Software defect prediction using ensemble learning: A

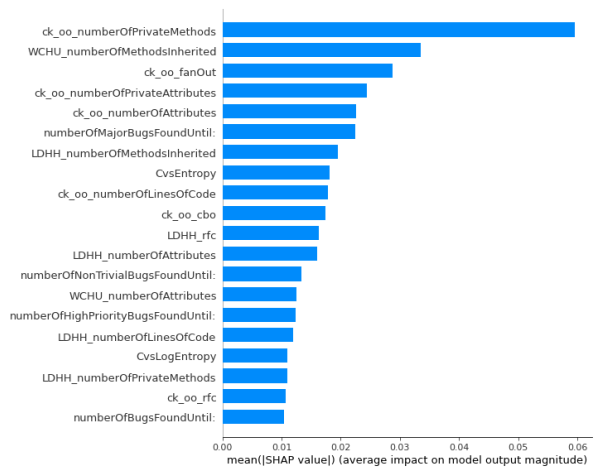
systematic literature review. *IEEE Access*.

부록

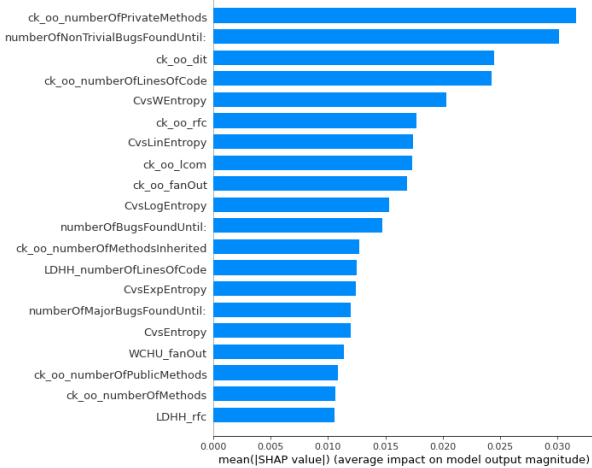
본 부록에서는 연구 질문 1에서 다루지 못한 실험 결과이다.



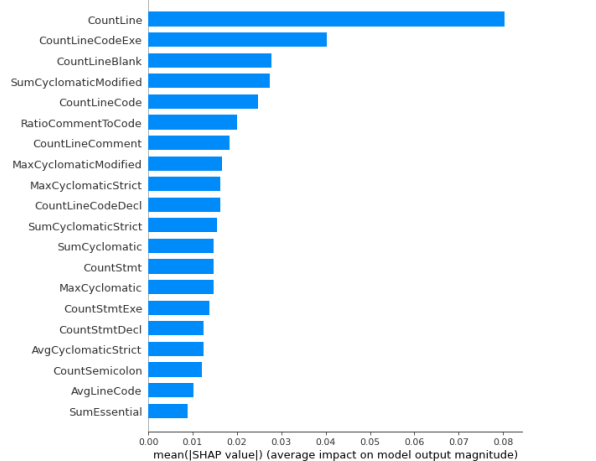
(a) LC 프로젝트에 SHAP을 적용한 결과



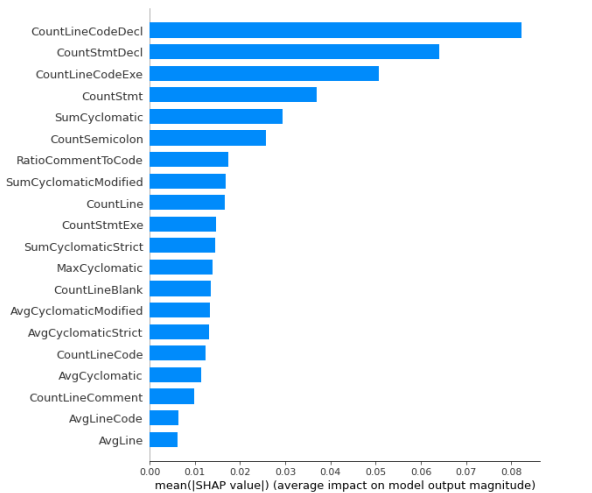
(b) ML 프로젝트에 SHAP을 적용한 결과



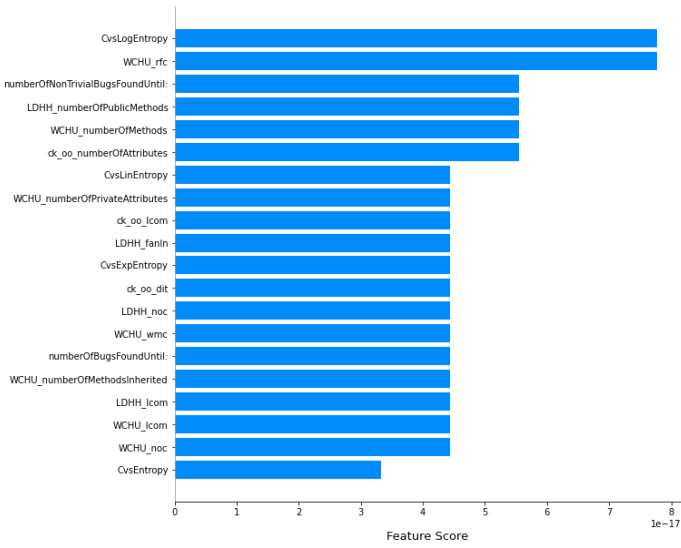
(c) PDE 프로젝트에 SHAP을 적용한 결과



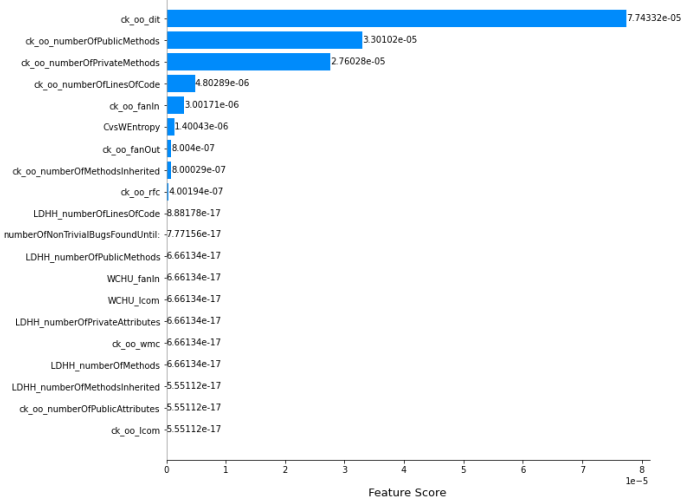
(d) Apache 프로젝트에 SHAP을 적용한 결과



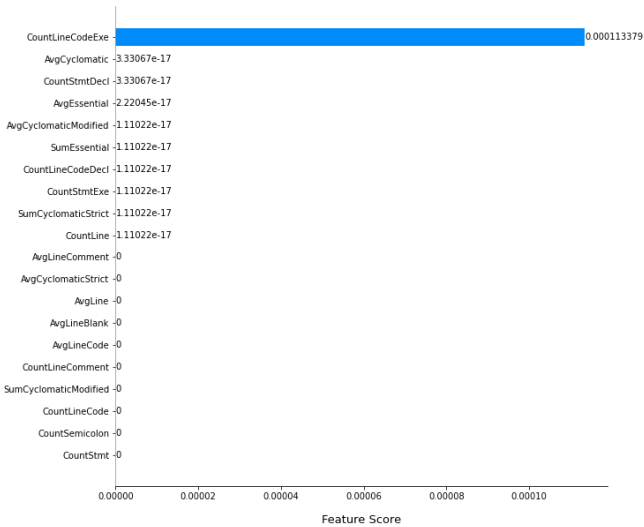
(e) Safe 프로젝트에 SHAP을 적용한 결과



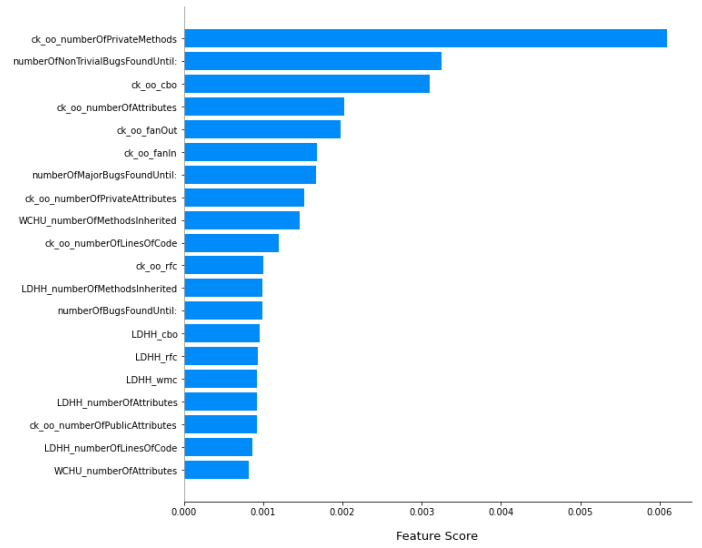
(f) LC 프로젝트에 DALEX를 적용한 결과



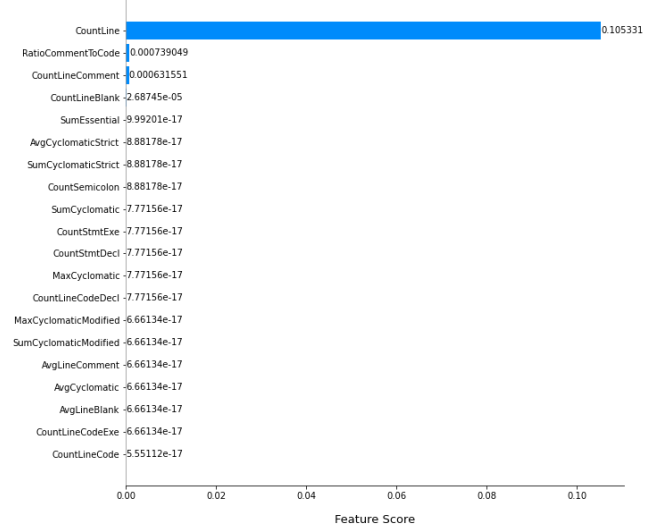
(h) PDE 프로젝트에 DALEX를 적용한 결과



(j) Safe 프로젝트에 DALEX를 적용한 결과



(g) ML 프로젝트에 DALEX를 적용한 결과



(i) Apache 프로젝트에 DALEX를 적용한 결과

복잡한 버그의 템플릿 생성을 위한 Change Action 측면에서의 과거 수정 이력 분석

허진석⁰¹, 이은석²

성균관대학교 전자전기컴퓨터공학과¹, 성균관대학교 소프트웨어대학²

{mrhjs225, jeonghh89, leees}@skku.edu

An Analysis of Fix History for Template of Complex Bugs: In the Aspect of Change Action

Jinseok Heo⁰¹, Eunseok Lee²

Department of Electrical and Computer Engineering, Sungkyunkwan University¹

College of Software, Sungkyunkwan University²

요 약

템플릿 기반 APR 기법은 좋은 성능을 보이는 APR 종류 중 하나이다. 하지만 템플릿 기반 APR 기법이 고칠 수 있는 버그들이 비교적 단순한 버그에 그친다는 지적이 존재한다. 본 논문에서는 더욱 복잡한 버그에 대한 템플릿 생성을 위해 Change Action 및 AST 노드 유형 측면에서 과거 패치들을 분석하였다. 벤치마크 Defects4j 2.0.0 835개의 버그들을 대상으로 분석한 결과 Change Action이 2개 와 3개인 그룹에 대해 우세한 패턴이 있음을 확인하였다. AST 노드 유형별 분석 결과 여러 If 스테이트먼트를 동시에 고치는 패턴이 가장 우세하며 템플릿 생성 시 가장 큰 효과를 볼 수 있는 가능성을 확인하였다.

1. 서 론

여러 APR 기법 종류 중 템플릿 기반의 APR 기법은 좋은 성능을 내는 기법의 하나이다[1]. Anil Koyuncu et al.[2]은 오픈 소스 저장소로부터 과거 패치 이력을 rich edit script 형태로 수집하여 더욱 풍부한 패치 템플릿을 생성하였다. Kui Liu et al.[3]은 과거 연구들에 등장하는 템플릿들을 정리하여 템플릿 기반 APR 기법의 전반적인 분석과 개발자의 패치와 의미상 동일한 패치인 옳은 패치 생성 성능을 확인하였다.

하지만 기존 연구[4]에서 템플릿 기반 기법들이 옳은 패치를 생성할 수 있는 버그들은 매우 간단한 버그들만이라는 지적이 존재한다. Kui Liu et al.[3]은 보다 복잡한 패치를 생성하기 위해서는 템플릿을 점진적으로 적용하면 해결할 수 있다고 설명하였다. 하지만 이러한 방식은 시간이 오래 걸릴 뿐 더러 정확한 방식이 아니다.

따라서 본 논문에서는 더욱 복잡한 버그를 해결하기 위해 Change Action 및 AST 노드 유형 측면에서 버그들을 분석하였다. 실제 복잡한 버그를 해결하기 위한 템플릿을 생성하기 위해서는, 데이터 마이닝 혹은 수작업으로 템플릿을 만들어야 한다. 그 이전에 어떠한 수준으로 복잡한 패치가 이루어지고 빈번하게 등장하는 패턴은 어떤 것인지에 대한 분석을 진행하였다. 이전 연구[5]에서도 기존 버그들에 대해 패치가 적용된 AST 노드 유형 측면에서 분석을 진행하였다. 하지만 실제 패치가 이루어진 스테이트먼트가 여러 줄일 경우 AST 노드 유형이 어떠한 쌍을 가지고 분포되어있는지에 대해서는 분석을 진행하지 않았다.

분석을 진행하기 위해 우리는 도구 PatchParser [5]를 사용하여 각각의 패치를 Change Action 수준으로 표현하였다. 그리고 패치별 해당 Change Action이 가해진 AST 노드 유형의 쌍을 식별하여 분석하였다.

본 논문의 기여점은 다음과 같다.

- ✓ 복잡한 버그에 대한 분석을 통해 템플릿 기반 APR의 접근 방식에 대한 insight 제공.
- ✓ 실제 자바 버그를 대상으로 분석.

이어지는 2장에서는 본 논문을 이해하는데 필요한 배경 지식을 소개한다. 3장에서는 분석 목표, 방법 및 사용된 데이터셋에 관해 설명한다. 4장에서는 분석 결과에 대해 설명하고 마지막 5장에서는 향후 연구와 함께 본 논문을 마무리 짓는다.

2. 배경 지식

2.1 Change Action

기존 논문[4]에서는 다음과 같은 Change Action의 정의와 함께 패치를 Change Action 수준으로 표현하여 분석하였다.

정의1. Change Action: 트리 형태로 표현된 Change Operation의 계층적 시퀀스.

Change Action에 사용되는 Change Operation은 Change instruction, Code element 유형, 부모 operation, 자식 operation의 집합으로 이루어진 하나의 튜플이며, 소스 코드의 수정을 표현하는 가장 작은

단위이다. Change instruction에는 Update(*UPD*), Insert(*INS*), Delete(*DEL*), Move(*MOV*)중 하나로 소스 코드 변경이 표현된다. Change Action의 Root Operation의 Code element는 ReturnStatement, IfStatement와 같은 Statement 유형만 허용된다.

Chart 8	
source/org/jfree/data/time/Week.java	
	@@ -172,7 +172,7 @@ public Week(Date time) {
172	172 */
173	173 public Week(Date time, TimeZone zone) {
174	174 // defer argument checking...
175	- this(time, RegularTimePeriod.DEFAULT_TIME_ZONE, Locale.getDefault());
175	+ this(time, zone, Locale.getDefault());
176	176 }
177	177 }
178	178 /**

(a)

Change Action:

- op_1 : (*UPD*, ConstructorInvocation, null, $\{op_2, op_3\}$)
- op_2 : (*INS*, SimpleName, op_1 , \emptyset)
- op_3 : (*DEL*, QualifiedName, op_1 , $\{op_4, op_5\}$)
- op_4 : (*DEL*, SimpleName, op_3 , \emptyset)
- op_5 : (*DEL*, SimpleName, op_3 , \emptyset)

(b)

그림 1 (a) Chart-8 버그의 개발자 패치
(b) 개발자 패치에 대한 Change Action

그림1은 특정 패치에 대한 Change Action과 Change Operation에 대한 예시를 보여준다. 해당 패치는 APR 벤치마크 Defects4j[6] 중 Chart-8 버그에 대한 개발자 패치이며 그림1(b)은 그에 대한 Change Action의 형태를 보여준다. Change Action의 정의에 따라 ConstructorInvocation statement가 Root Operation이 된다. 그리고 op_1 을 살펴보게 되면 특정 매개변수를 바꾸게 되므로 *UPD*가 튜플의 첫 번째 요소이다. 두 번째 요소로 Code element 유형으로 표기되며 기존 논문[4]에서는 AST 노드 유형을 사용하였다. 세 번째, 네 번째 요소는 각각 부모, 자식 operation으로 op_1 의 경우 Root operation이므로 부모 operation이 존재하지 않고, 자식 operation만 존재한다.

본 논문에서는 해당 패치의 수준을 Change Action 수준으로 표현하여 분석하고자 한다.

2.2 간단한 버그 vs. 복잡한 버그

기존 논문[4]에서는 간단한 버그와 복잡한 버그의 기준으로 Change Action의 수를 사용하였다. 해당 논문에서는 3개 이상의 Change Action을 가지는 패치가 가해질 경우 복잡한 버그로 분류하였다. 해당 수치는 기존 APR 기법들이 옳은 패치를 생성할 수 있었던 버그들을 분석했을 때 Change Action이 3개 이상인 패치는 거의 생성하지 못하였기 때문에 도출된 수치이다.

실제 기존 APR 기법들이 옳은 패치를 생성할 수 있는 버그 100여 개[1]를 분석했을 때 Change Action이

1개인 버그가 70%에 달했다. 따라서 복잡한 버그를 기존의 APR이 해결하기 어려운 버그로 설정하되 기존 기법보다 폭넓게 설정하고자 본 논문에서는 Change Action이 2개 이상이 되는 모든 버그를 복잡한 버그로 고려하여 분석하였다.

3. 분석 방법

3.1 분석 목표

- 기존의 APR 기법이 해결하지 못한 복잡한 버그들의 경우 가해지는 Change Action에 우세한 패턴이 존재하는가?
- 복잡한 버그들의 경우 어떠한 AST 노드 유형들에 대해 패치가 이루어지는가?

3.2 분석 방법

우리는 분석을 위해 과거 패치들을 대상으로 Change Action을 추출하였다. Change Action 추출을 위해 PatchParser라는 도구를 사용하였다. 해당 도구는 기존 연구[4, 5]에서 Change Action 추출 및 패치 생성을 위해 사용된 도구이다.

우선 각 버그의 개발자 패치를 Change Action 수대로 분류한다. 그리고 각 패치를 구성하는 Change Action들을 대상으로 Root operation이 가해진 AST 노드 타입을 식별한다. 각각의 패치별로 식별된 AST 노드 유형의 목록을 하나의 패턴으로 식별하여 해당 패턴이 얼마나 등장하는지를 계산한다.

3.3 데이터 셋

APR 연구에서 활발히 사용되고 있는 자바 기반 벤치마크인 Defects4j[6] 2.0.0의 835개의 버그를 대상으로 분석하였다.

4. 분석 결과

4.1 버그 별 change action 분포

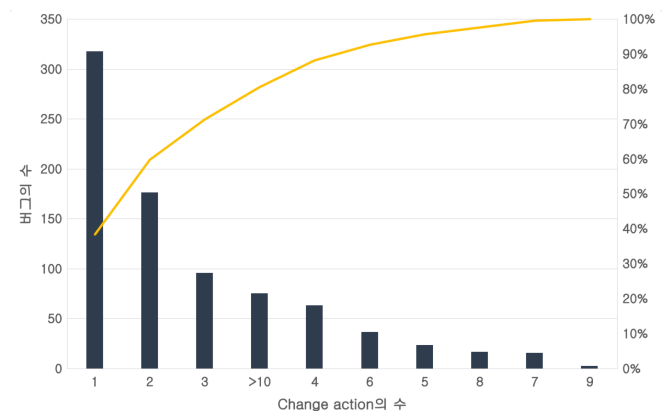


그림 2 change action 수에 따른 버그 분포

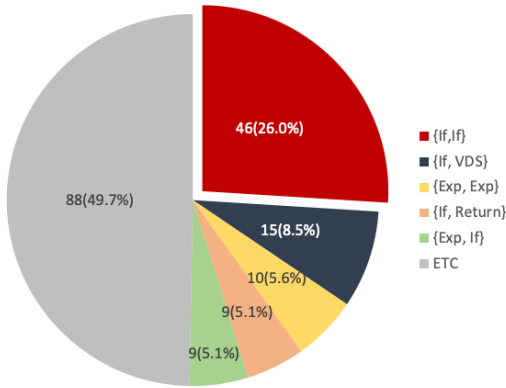


그림 3 Change Action이 2개 일 때의 패턴 분포

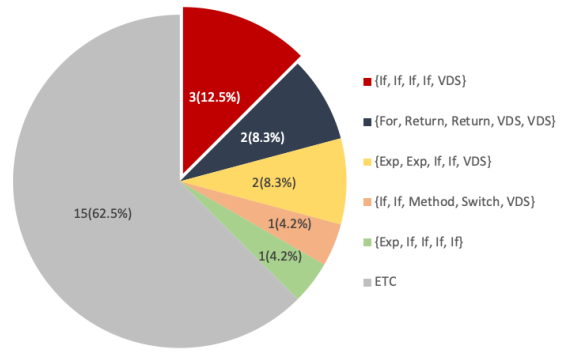


그림 6 Change Action이 5개 일 때의 패턴 분포

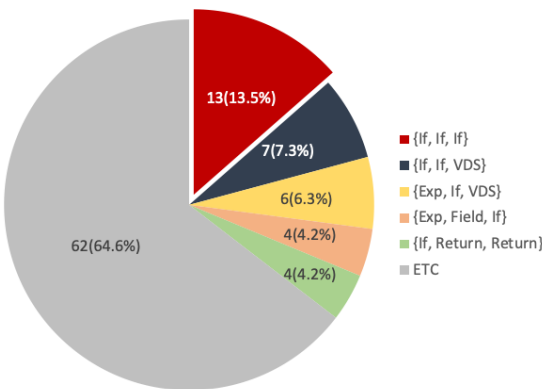


그림 4 Change Action이 3개 일 때의 패턴 분포

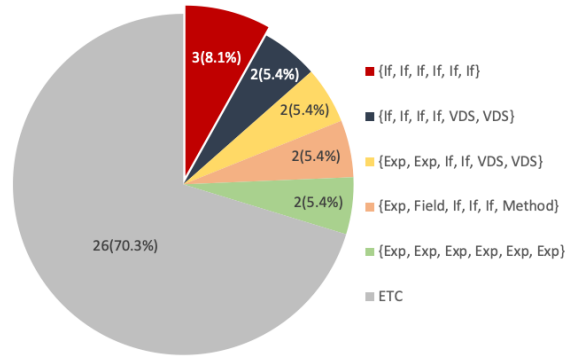


그림 7 Change Action이 6개 일 때의 패턴 분포

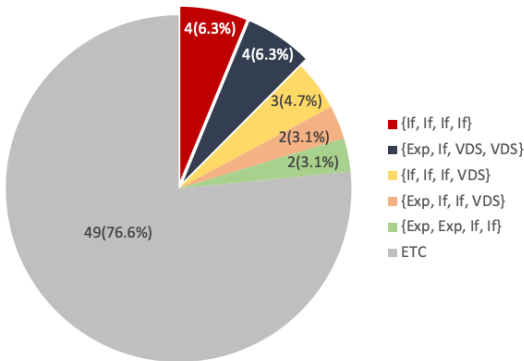


그림 5 Change Action이 4개 일 때의 패턴 분포

그림 2는 Change Action의 수에 따라 버그가 얼마나 얼마나 분포되어 있는지를 나타낸 그래프이다. 막대그래프의 X축은 Change Action 수 항목이며 왼쪽 Y축은 해당 Change action 수로 이루어진 패치를 가지는 버그의 수를 나타낸 것이다. 꺾은 선 그래프는 전체 835개의 버그에 대한 누적 비율을 표현한 것이다. 10개 이상으로 이루어진 경우 Change Action 수별로 버그의 수가 매우 적어 통합하여 표기하였다.

표 1 범례에 사용된 약어의 원말

약어	원말
If	IfStatement
VDS	VariableDeclarationStatement
Exp	ExpressionStatement
Return	ReturnStatement
Field	FieldStatement
For	ForStatement
Method	MethodDeclaration
Switch	SwitchCase

Change Action이 1개로 이루어진 패치를 가지는 버그가 318개로 가장 많았다. 2개 이상으로 이루어진 버그는 517개로 약 62%에 달했다. Change Action의 수가 많아질수록 버그의 수는 작아졌다. 우리는 Change action이 1개로 이루어진 패치를 제외하고 패턴 분포 분석을 위해 최소 20개 이상의 버그가 존재하는 그룹을 대상으로 분석하였다. 대상 그룹은 패치가 Change Action이 2개로 이루어진 버그부터 6개로 이루어진 버그 그룹이다.

4.2 Change action 규모별 패턴 분석

그림 3부터 그림 7은 Change Action 수별 패턴 분포를 보여준다. 각 그룹에서 등장하는 패턴 중 가장 많이 등장한 순으로 나열했을 때 Top 5에 대해 버그의

수와 비율을 따로 나타내었고 나머지 패턴은 모두 ETC에 포함해 표기하였다. 원형 그래프의 데이터 라벨링인 $X(Y\%)$ 은 해당 패턴을 가지는 X 개의 패치와 해당 패치가 Change Action이 n 개인 그룹에서 차지하는 비율 $Y(\%)$ 를 나타낸다. 또한 범례에서 볼 수 있듯 등장한 모든 패턴의 원말을 표기하는 것이 힘들어 모두 약어로 나타내었고, 표1에 그림 3부터 그림 7까지의 범례에 등장하는 약어에 대해 원말을 표기하였다.

우선 우세한 패턴이 뚜렷하게 존재하는 그룹은 Change Action이 2개와 3개인 그룹이었다. Change Action이 2개인 경우 가장 많이 등장하는 패턴의 비중이 1/4까지 달하였으며 Top 5가 차지하는 비중 또한 50%를 넘어갔다. 3개인 경우에도 비슷하게 우세한 패턴이 존재한다는 뚜렷한 양상을 볼 수 있었다.

하지만 Change Action이 점차 많아질수록 이러한 양상은 흐릿해진다. Change Action이 4개인 경우 Top 5 중 2개의 패턴은 2번밖에 등장하지 않았으며 이는 ETC에 포함된 모든 패턴이 2번 이하로 등장하였음을 의미한다. Change Action이 5개, 6개인 그룹도 마찬가지로 우세한 패턴을 찾기 힘들었으며 Top 5에 속하더라도 ETC에 포함된 전체 패턴과 비교했을 때 비슷한 비중을 차지함을 볼 수 있다.

따라서 Change Action이 2개와 3개인 그룹의 경우에는 기존의 템플릿보다 복잡한 템플릿을 생성하는 것이 가능할 것이고 많은 비중의 패치를 고칠 수 있음을 기대할 수 있다. 하지만 Change Action이 커짐에 따라 등장하는 패턴의 비율이 줄어들어 모든 패턴에 대한 템플릿을 생성하는 것이 힘들며, 생성하더라도 많은 수의 패치를 생성할 수 없다. 따라서 Change Action이 비교적 적은 2개 혹은 3개인 그룹에 대하여 데이터 마이닝 및 템플릿을 생성하고 이를 조합하여 Change Action이 많은 패치를 생성하는 방식이 요구된다.

그림 3부터 그림 7까지 Change Action 별 가장 많이 등장한 패턴을 보았을 때 Change Action이 5개일 때를 제외하고 가장 많이 등장한 패치 패턴은 If 스테이트먼트들을 고친 패턴이었다. 기존 연구 [5]에서도 버그 수정을 위해 가장 많이 수정되는 스테이트먼트 유형이 If 스테이트먼트 인 것을 밝혔었다. 이러한 경향은 여러 스테이트먼트를 고치는 버그에 대해서도 동일하게 적용됨을 알 수 있고 Change Action 수와 무관하게 If 스테이트먼트를 동시에 수정하는 템플릿이 가장 많은 비중을 차지하며, 템플릿 생성 시 가장 많은 버그를 고칠 수 있는 가능성을 확인하였다.

5. 결론

본 논문에서는 기존의 템플릿 기반 APR 기법들이 해결할 수 있는 버그보다 복잡한 버그를 해결하고자 복잡한 템플릿 생성을 위한 과거 패치 분석을

진행하였다. Change Action 수준에서 분석한 결과 Change Action 수가 많아 짐에 따라 모든 템플릿을 생성하는 것이 힘들지만 Change Action이 2개, 3개인 경우 템플릿을 생성한다면 많은 수의 버그들을 해결할 수 있음을 확인하였다. 또한 과거 버그 수정을 위해 If 스테이트먼트가 가장 많이 수정되는 것 또한 복잡한 버그이더라도 비슷한 경향을 보이는 것을 확인할 수 있었다. 향후 연구로 Change Operation 수준의 분석과 함께 실제 스테이트먼트 유형 단위로 템플릿을 만들어 패치를 생성하는 기법을 연구하고자 한다.

Acknowledge

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2019R1A2C2006411).

참고 문헌

- [1] Kui Liu, Shangwen Wang, Anil Koyuncu, Kisub Kim, Tegawendé F Bisseyandé, Dongsun Kim, Peng Wu, Jacques Klein, Xiaoguang Mao, and Yves Le Traon, On the efficiency of test suite based program repair: A systematic assessment of 16 automated repair systems for java programs, In Proceedings of the 42nd International Conference on Software Engineering, pp. 615–627, 2020.
- [2] Anil Koyuncu, Kui Liu, Tegawendé F Bisseyandé, Dongsun Kim, Jacques Klein, Martin Monperrus, and Yves Le Traon, Fixminer: Mining relevant fix patterns for automated program repair, Empirical Software Engineering, vol. 25, no. 3, pp. 1980–2024, 2020.
- [3] Kui Liu, Anil Koyuncu, Dongsun Kim, and Tegawendé F Bisseyandé, Tbar: revisiting template-based automated program repair, In Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, 31–42, 2019.
- [4] Deheng Yang, Kui Liu, Dongsun Kim, Anil Koyuncu, Kisub Kim, Haoye Tian, Yan Lei, Xiaoguang Mao, Jacques Klein, and Tegawendé F Bisseyandé, Where were the repair ingredients for Defects4j bugs?, Empirical Software Engineering, vol. 26, no. 122, pp. 1–33, 2021.
- [5] Kui Liu, Dongsun Kim, Anil Koyuncu, Li Li, Tegawendé F Bisseyandé, and Yves Le Traon, A Closer Look at Real-World Patches, In Proceedings of the 2018 IEEE International Conference on Software Maintenance and Evolution, pp. 275–286, 2018.
- [6] René Just, Darioush Jalali, and Michael D. Ernst, Defects4j: A database of existing faults to enable controlled testing studies for java programs, In Proceedings of the 2014 International Symposium on Software Testing and Analysis, pp. 437–440, 2014.

동영상 스트리밍 서비스에 HTTP/1.1 HTTP/2 프로토콜 적용별 프로토콜 성능 평가

도지영^o, 이성원

소프트웨어융합대학 소프트웨어융합학과
2017104589@khu.ac.kr, drsungwon@khu.ac.kr

Protocol Performance evaluation by application of HTTP/1.1 HTTP/2 protocol to video streaming service

Jiyoung Do^o, Sungwon Lee

Department of Software Convergence, Kyung Hee University

요 약

OTT 플랫폼이 등장하고, 동영상 스트리밍 서비스의 수요가 증가하고 있다. 동영상 스트리밍 서비스의 발전을 위해서 효율적인 전송을 가능하게 하는 점이 중요하다. 이 전송을 가능하게 하는 기반에 HTTP 프로토콜이 있다. HTTP/1.1의 취약점을 보완하고자 HTTP/2가 나왔지만 모든 기능적인 면에서 성능이 좋다고 볼 수는 없다. 본 연구는 동영상 스트리밍 서비스에 대해 HTTP/1.1과 HTTP/2 프로토콜의 성능을 두가지 실험을 통해 평가한다. 첫번째 실험은 동영상 해상도에 따라 두 가지 프로토콜의 성능을 측정하는 실험이다. 두번째 실험은 네트워크 대역폭의 제한을 통해 두 가지 프로토콜의 성능을 측정하는 실험이다. 실험을 통해 페이지 로드시간, TTFB, 데이터 전송량 측면에서 성능을 평가하고 각 프로토콜이 적절한 사용예시를 제안한다.

1. 서 론

모바일 기기의 보급이 대중화 되고, 네트워크 기술이 발전함에 따라 여러 모바일 기기를 통해 동영상 스트리밍 서비스를 즐길 수 있는 넷플릭스(Netflix), 유튜브(Youtube) 같은 OTT (Over the Top) 플랫폼이 등장하였다[1]. 이런 OTT 플랫폼들이 제공하는 동영상 스트리밍 서비스의 수요는 더욱 증가하고 있으며, 실시간으로 동영상을 재생하는 기법의 기반에는 프로토콜이 자리잡고 있다. 1999년 HTTP/1.1 프로토콜이 발표된 당시와 현재의 웹 페이지 환경은 많이 달라졌다. 그렇기 때문에 현재의 환경을 예측하지 못한 HTTP/1.1 프로토콜은 HOL 블로킹(HeadofLine 블로킹), TCP 혼잡회피로 인한 속도 저하, 비대칭 헤더 등의 문제점을 가지고 있다. 이런 문제점을 개선하고자 구글에서 SPDY 프로토콜을 기반으로 한 표준 HTTP/2 프로토콜을 공개하였다. HTTP/2가 지원하는 다중화 특성, 동시성 특성, 헤더 압축, 서버 푸시, 프레임링과 같은 핵심 기능은 성공적이었다. 이런 개선으로 인해 2017년 전세계 웹 서비스의 17%를 차지하던 HTTP/2는 2018년 11월 31%로 확산되기도 하였다[2]. 이런 점에서 HTTP/2 프로토콜이 HTTP/1.1보다 현재의 웹 환경에 더 적합한 프로토콜이라고 할 수 있지만 다른 보안 취약점이 나타나기도 한다[3]. 즉, HTTP/2 프로토콜은 이전 버전의 HTTP/1.1 프로토콜의 문제점을 개선하고자 발표되었지만, 모든 부분에서 더 낫다고 볼 수는 없다. 또한 동영상 스트리밍 서비스의 성능적인 측면을 두개의 프로토콜에 대해 분석한 연구 결과는 충분하지 않다. 그렇기에 본 연구에서는 HTTP/1.1, HTTP/2 프로토콜 적용에 따른 동영상 스트리밍 서비스의 성능을 분석하기 위한 두가지 실험을 진행하고자 한다. 첫번째 실험은 동영상 해상도를 기준으로 프로토콜별 성능을 평가하는 것이다. 480p, 720p, 1080p 세 가지 해상도의 동영상을 기준으로 페이지 로드시간, TTFB(Time to First Byte Speed), 데이터 전송량을 측정한다. 두번째 실험은 현재

모바일 기기로 동영상 스트리밍 서비스를 이용하는 사용자가 많기 때문에 이런 모바일 환경을 재현하고자, 네트워크 다운로드 대역폭을 제한하는 실험이다. 1Gbps, 1Mbps, 400Kbps 세가지 대역폭을 기준으로 페이지 로드시간, TTFB, 데이터 전송량을 측정한다. 본 연구는 첫 번째 실험을 통해 동영상 스트리밍 서비스를 구현할 때, 해상도 별로 더 효율적인 프로토콜을 고려하는 것을 도울 수 있는 정량적 지표를 제공하고자 한다. 두 번째 실험 결과를 통해서 모바일 기기로 동영상 스트리밍을 사용하는 사용자에게 제공하는 서비스의 질(Qos, Quality of Service)을 향상하는 방향을 고려할 수 있을만한 정량적인 지표를 제공하고자 한다.

2. 본 론

동영상 스트리밍 서비스를 사용하는 사용자 입장에서 페이지 로드시간은 중요하다. 페이지 로드시간은 웹 페이지에 표시될 때까지 걸리는 시간, 즉 모든 HTTP 요청이 처리되는 시간을 의미한다. TTFB는 페이지 로딩 속도에 기여하는 가장 중요한 요소 중 하나이다. 이 실험 환경에서 보안 계층이 선택적인 HTTP/1.1은 보안 계층 없이 설계하였고, 보안 계층이 필수적인 HTTP/2에서는 보안 계층이 적용되게 설계 하였다. 그러므로 HTTP/2 서버는 클라이언트와 데이터를 주고 받을 때, 서버의 공개키를 송신하는 별도의 과정이 추가된다. 그렇기 때문에 HTTP/2의 TTFB가 상대적으로 느릴 수 밖에 없는 구조를 갖는 것을 고려해야 한다. TTFB는 최종 페이지를 불러 일으키는 속도와 동일하지는 않지만 앞서 중요하다고 언급한 페이지 로드시간에 영향을 미치는 요소라는 점이 주목할 점이다. 데이터 전송량은 HTTP/2에서 제공하는 헤더 압축의 영향을 고려할 수 있기 때문에 중요하다. 실험 환경은 아래 표와 같다.

표1. 실험 환경

CPU	Apple M1
Memory	16GB
OS	macOS
Server	nodejs
Browser	Chrome 95.0.4638.54

서버는 Nodejs로 구축되어 있으며, HTTP/1.1은 Nodejs에서 기본으로 제공하는 기본 http 모듈을 사용하여 서버를 구동하였다. HTTP/2은 npm을 사용하여 ‘spdy’ 모듈을 사용하여 서버를 구동했다. HTTP/2는 보안계층이 필수적으로 존재하므로 openssl을 통해 ssl 인증서를 발급받아, HTTP/2 서버 내에 배치하였다. 현재 대다수의 브라우저에서 HTTP/2 프로토콜을 지원하고 있는데, 해당 실험에서는 Chrome 95.0.4638.54 을 사용하였다. 첫 번째 실험은 480p, 720p, 1080p 동영상 해상도 별로 HTTP/1.1과 HTTP/2 프로토콜의 성능을 평가하는 것이다. 각각 10회씩 실험을 반복하였으며, 평균 값을 표에 정리 하였다. 데이터 전송량은 매 실험 결과가 표에 기입한 값으로 일관되게 측정되었다. 실험 결과는 아래와 같다.

표2. 동영상 해상도 별 HTTP/1.1 와 HTTP/2 실험 결과

480p	페이지 로드시간	데이터 전송량	TTFB
HTTP/1.1	125.5 ms	22.5 MB	1.01 ms
HTTP/2	101.9 ms	27.2 MB	7.565 ms
720p	페이지 로드시간	데이터 전송량	TTFB
HTTP/1.1	118.3 ms	38.2 MB	6.29 ms
HTTP/2	98.4 ms	42.2 MB	6.42 ms
1080p	페이지 로드시간	데이터 전송량	TTFB
HTTP/1.1	141 ms	95.5 MB	1.814 ms
HTTP/2	101.2 ms	95.6 MB	5.327 ms

위의 표2를 보면, 480p, 720p, 1080p 동영상 모두 페이지 로드시간이 HTTP/2가 HTTP/1.1 보다 우수하다. 480p 동영상은 23.6ms만큼의 차이를 보이고, 720p 동영상은 19.9ms, 1080p 동영상은 39.8ms 의 차이를 보인다. HTTP/2가 보안 계층이 적용되었음에도 HTTP/1.1 보다 페이지 로드시간이 우수하다는 것은 속도와 보안적인 면 모두 나은 성능을 보인다고 볼 수 있다. 또한 이 보안 계층으로 인해 커넥션 방식의 차이가 발생하여 TTFB는 HTTP/1.1이 HTTP/2에 비해 모두 우수한 것을 확인할 수 있다. 데이터 전송량은 HTTP/2보다 HTTP/1.1이 더 적었다. 480p 동영상에 대해서는 4.7MB, 720p 동영상은 4MB , 1080p 동영상은 0.1MB의 차이를 보인다. HTTP/2 에서 제공하는 헤더 압축이란 연속된 요청사이의 유사한 내용의 헤더들을 압축시키는 것으로 오버헤드를 막아준다. 보안계층의 영향으로 HTTP/2의 데이터 전송량이 높지만 해상도가 높아짐에 따라 요청사이의 중복을 줄여 1080p 동영상에서는

불과 0.1MB의 차이만을 보이는 것으로 볼 수 있다. 두번째 실험은 네트워크 대역폭을 제한하여 모바일 환경을 재현함으로써 유의미한 결과를 도출하는데 목적이 있는 실험이다. 400kbps, 3Mbps, 4G LTE 세가지 대역폭으로 실험을 진행하였다. 대역폭을 제한하기 위해서 Chrome 95.0.4638.54의 개발자 도구의 Network 탭에서 스로틀링(throttling) 기능을 사용하였다. 세가지 모두 커스텀 하여 400kbps는 400kbit/s 이고, 3Mbps는 3.1Mbit/s , 4G LTE는 1Gbps로 1000Mbit/s 이다. 기본적인 실험 환경 및 방식은 첫번째 실험과 같지만 각각의 실험은 5회씩 반복되었다. 또한 데이터 전송량은 매 실험 결과가 표에 기입한 값으로 일관되게 측정되었다. 실험 결과는 아래와 같다.

표3. 대역폭을 제한한 HTTP/1.1와 HTTP/2 실험 결과

4G LTE	페이지 로드시간	데이터 전송량	TTFB
HTTP/1.1	83 ms	38.3 MB	4.36 ms
HTTP/2	104.6 ms	42.2 MB	6.30 ms
3Mbps	페이지 로드시간	데이터 전송량	TTFB
HTTP/1.1	254.2 ms	38.1 MB	8.324 ms
HTTP/2	264 ms	42.2 MB	8.776 ms
400kbps	페이지 로드시간	데이터 전송량	TTFB
HTTP/1.1	1.488 s	37.8MB	26.34ms
HTTP/2	1.41 s	42.4MB	20.47 ms

페이지 로드시간을 살펴보았을 때, 4G LTE와 3Mbps 대역폭 환경에서 HTTP/1.1 의 성능이 더 우수하다. 하지만 400kbps 대역폭에서는 HTTP/2 의 성능이 우수하다. 2015년도에 진행된 연구는 호스트 사이의 경로에서 패킷 손실율이 높아질 경우 HTTP/2의 성능이 HTTP/1.1보다 저하된다고 보고했다[4]. 패킷 손실율이 높아지면 TCP 연결은 Congestion Control 알고리즘을 동작시켜 일시적으로 Cwnd(Congestion Window) 크기를 줄인다. 이를 통해 HTTP/2는 HTTP/1.1보다 더 큰 처리율 손해를 경험한다[2]. 데이터 전송량을 살펴보았을 때, 처리율 손해를 면이 영향을 미치는 것으로 보인다. HTTP/1.1 프로토콜에서는 대역폭이 낮아질수록 데이터 전송량이 줄어들고, HTTP/2 프로토콜에서는 4G LTE와 3Mbps 에서는 동일하지만 400kbps의 낮은 대역폭에서는 0.2MB 더 늘어난 결과를 보인다. 이 부분에서 HTTP/1.1의 데이터 전송량이 모든 경우에서 HTTP/2 보다 적어 더 우수하다는 점도 주목할 점이다. TTFB 같은 경우 커넥션 방식을 고려하였을 때 HTTP/2가 항상 더 많은 시간을 소요할 것을 예상하였지만, 400kbps의 낮은 대역폭일 때는 HTTP/2의 TTFB가 HTTP/1.1 보다 더 우수한 것을 볼 수 있다. 즉, 400kbps 이하의 대역폭에서는 페이지 로드시간, 데이터 전송량, TTFB 3가지 측면 모두에서 HTTP/2 프로토콜이 HTTP/1.1 보다 우수하다는 결과를 얻을 수 있었다. 하지만 그보다 높은 대역폭에서는 모든 성능적 부분에서 HTTP/1.1의 성능이 우수했다.

3. 결 론

본론에서 동영상 해상도별 프로토콜 성능 분석 실험과 대역폭 제한별 프로토콜 성능 분석 실험을 수행하고 결과를 살펴 보았다. 첫번째 실험인 동영상 해상도별 프로토콜 성능 분석 실험 결과로, 보안 계층의 적용으로 인해 TTFB 는 HTTP/1.1이 HTTP/2 보다 우수하였지만, 그럼에도 불구하고 HTTP/2의 페이지 로드시간이 480p, 720p, 1080p의 동영상 모두에서 더 우수한 것을 확인 하였다. 데이터 전송량은 두 개의 프로토콜 차이로 전반적으로 HTTP/2가 전송하는 양이 많았지만 해상도가 증가함에 따라 요청이 늘어나는 과정에서 헤더 압축의 영향이 있는 부분을 확인 할 수 있었다. 두번째 실험인 대역폭 제한별 성능 분석 실험 결과로, 400kbps 이하의 대역폭에서는 측정한 3가지 성능 평가 지표에 대해서 HTTP/2가 우수한 것을 확인 할 수 있었다. 하지만 이보다 더 높은 대역폭인 3Mbps, 4G LTE의 높은 대역폭에서는 페이지 로드시간 측면에서 HTTP/1.1 프로토콜의 적용이 훨씬 우수한 것을 확인 할 수 있었다. 본 연구의 실험 결과를 통해 동영상 스트리밍 서비스를 구현하며 동영상 해상도와, 대역폭의 조건을 고려하여 프로토콜을 선택시에, 더 나은 동영상 스트리밍 서비스의 질(QoS, Quality of Service)를 제공 할 수 있는 지표로 사용될 수 있다.

4. 참고 문헌

- [1]윤기명,정광수, “DASH 환경에서 QoE 향상을 위한 콘텐츠 특성 기반의 비디오 품질 조절 기법, 정보과학회논문지, 42, 8, 1039-1048, 2015.
- [2]민지원,이영석, “웹 성능 개선을 위한 HTTP/1.1 과 HTTP/2 동시 사용 방법”, 한국정보과학회 학술발표논문집, 1046-1048, 2018.
- [3]류정현,문서연,박중혁, “안전한 Web 환경을 위한 HTTP/2 취약점에 대한 연구”, 추계학술발표대회, 23, 2, 238-240, 2016.
- [4]de Saxcé, Hugues, Iuniana Oprescu, and Yiping Chen, Is HTTP/2 really faster than HTTP/1.1?, Computer Communications Workshops (INFOCOM WKSHPs), 2015 IEEE Conference on. IEEE, 2015.
- [5]송민정,영유성근,박상일, “실감미디어 전송을 위한 차세대 HTTP 기반 적응적 스트리밍 전송 프로토콜 연구, 방송공학회논문지, 24, 4, 602-612, 2019.

멀티 청크 버그 정정을 위한 버그 블럭과 CodeBERT 활용

김지성^{○*}, 호혜민*, 양근석**, 이병정*

* 서울시립대학교 컴퓨터과학과, ** 경남대학교 컴퓨터공학과

* {kimjisung78, hyemin, bjlee}@uos.ac.kr, ** gsyang@kyungnam.ac.kr

Towards Multi-Chunk Bug Repair with Buggy Block and CodeBERT

Jisung Kim^{○*}, Hu Huimin*, Geunseok Yang**, Byungjeong Lee*

* Dept. of Computer Science, University of Seoul,

** Dept. of Computer Engineering, Kyungnam University

요 약

APR(Automated Program Repair)은 버그가 발생한 프로그램의 정정 과정에 대한 자동화 기술이며, 버그 위치를 식별하는 버그 로컬라이제이션(Bug Localization) 단계와 식별된 버그를 정정하는 버그 정정(Bug Repair) 단계 순으로 진행된다. 그리고 버그 청크 개수에 따라 단일 청크 버그(Single-Chunk Bugs)와 다중 청크 버그(Multi-Chunk Bugs)로 구분된다. 본 논문에서는 버그 블럭과 CodeBERT 모델을 적용하여 멀티 청크 버그 정정(Multi-Chunk Bug Repair) 기법을 제안한다. 먼저 사전 학습된 CodeBERT 모델에 버그 청크를 기준으로 전처리한 버그 블럭으로 미세 조정한다. 그 다음, 미세 조정된 모델을 통해 버그 청크 별로 정정 청크를 생성하고 순서대로 매핑한 뒤 후보 패치 집합을 생성한다. 각각의 후보 패치에 평가를 진행하고, 평가 결과에 따라 정정을 판단한다. 또한 본 논문에서는 기계 학습 기반 멀티 버그 정정의 문제점을 소개하고 해결 방법을 제시하고, 적절성을 평가하기 위해 베이스라인과 정성적으로 비교한다.

1. 서론

소프트웨어 버그(이하 "버그"라고 함)는 예상하지 않았거나 잘못된 결과를 만들어 프로그램이 오작동하게 만든다. 버그는 프로그램의 오작동 원인이고 버그 위치를 식별하고 정정하는데 많은 시간과 비용이 소요되기 때문에, 소프트웨어의 개발 생산성과 운영 및 유지보수에 직접적인 영향을 미친다. 이러한 문제를 해결하기 위해 자동 프로그램 정정(Automated Program Repair, 이하 "APR"이라고 함)이 등장하게 되었다. APR은 버그가 존재하는 프로그램의 정정 과정을 자동화하는 기술이며, 버그 위치를 식별하는 버그 로컬라이제이션(Bug Localization) 단계와 식별된 버그를 정정하는 버그 정정(Bug Repair) 단계 순으로 진행된다. APR와 관련하여 많은 영역이 관심을 받고 있고, 해당 영역에 대한 연구가 활발하게 진행되고 있다.

버그가 존재하는 코드에서 1개 이상의 연속되는 버그 라인을 버그 청크(Buggy Chunk)라고 하며, 버그 청크 개수에 따라 단일 청크 버그(Single-Chunk Bugs)와 다중 청크 버그(Multi-Chunk Bugs)로 구분된다. APR의 버그 정정 단계 연구에 기계 학습을 적용하는 여러 기법이 있다. SequenceR[1]은 NMT(Neural Machine Translation) 모델에 복사 메커니즘(Copy Mechanism)을 적용한 APR 연구이다. 그리고 학습 진행을 위해 버그 컨텍스트 추상화(Buggy Context Abstraction) 전처리를 수행한다. DLFix[2]는 2계층 트리 기반 인코더-디코더 모델을 사용하여 NMT 문제점을 해결하려고 한 APR 연구이다. 버그 정정의 컨텍스트를 학습하는 첫번째 계층과 이전 버그 정정의 결과를 학습하는 두번째 계층으로 구성되어 있다. 하지만 2가지 연구 모두 단일 청크 버그 정정만 가능하다.

본 논문은 싱글 청크 버그 정정(Single-Chunk Bug Repair)이 아닌 멀티 청크 버그 정정(Multi-Chunk Bug Repair)을 목표로 한다. 그래서 버그 청크와 버그 청크와 관련된 정보인 버그 컨텍스트(Buggy Context)로 구성된 버그 블럭(Buggy Block)과 소스 코드용 BERT인 CodeBERT[3]를 적용한 멀티 청크 버그 정정 APR 기법을 제안한다. 본 논문은 APR 단계 중 버그 정정에 집중하며, 디버깅에 초점을 맞춘다.

본 논문이 기여하는 부분은 다음과 같다.

- 버그 블럭과 CodeBERT를 적용하여 멀티 청크 버그 정정을 수행하는 APR 기법을 제안한다.
- 기계 학습 기반 멀티 청크 버그 정정 시 발생하는 문제를 분석하고, 해결 방법을 제시한다.
- 소프트웨어를 개발하거나 유지보수 할 때, 개발자의 생산성 향상에 기여한다. 특히 새로운 개발자인 경우에는 생산성과 품질 개선에 크게 영향을 미친다.

본 논문의 구성을 다음과 같다. 2장에서는 관련 연구를 소개한다. 3장에서는 본 논문에서 제안하는 APR 기법과 CodeBERT 모델, 기계 학습 기반 멀티 청크 버그 정정 시 발생하는 문제점을 소개하고 해결 방법을 제시한다. 5장에서는 관련 APR 연구와 정성적 비교를 통해 토의를 진행하며, 6장에서는 결론과 향후 연구를 제시한다.

2. 관련 연구

HECURLS[4]는 AST(Abstract Syntax Tree)와 유전 연산 알고리즘을 통해 멀티 청크 버그 정정을 수행한 APR 연구이다. 반면에 이 연구는 기계 학습 기반으로 버그 정정을 수행하지 않는다.

CoCoNuT[5]은 문맥 인지(Context-Aware) 전략 기반 NMT 모델을 적용한 APR 연구이다. 단어 수준 토큰화(Word-Level Tokenization)으로 전처리하며, 버그 라인을 학습하는 인코더와 버그 컨텍스트를 학습하는 인코더로 구성되어 있다. 두 인코더 모두 CNN(Convolutional Neural Network)을 사용하였다.

CURE[6]는 GPT(Generative Pre-Training) 모델을 코드 인지(Code-Aware) 전략 기반 NMT 모델에 적용한 APR 연구이다. 단어 수준 토큰화와 서브워드 토큰화(Subword Tokenization)로 전처리한 뒤, 길이 제어(Length-Control)와 유효 식별자 검사(Valid-Identifier Check)를 사용하여 후보 패치를 생성하였다. 이 연구는 학습 모델이 다른 APR 연구에 비해 복잡하고, 그로 인해 학습 데이터 집합의 규모가 매우 크다.

또한 [5][6]은 여러 모델 중 가장 성능이 좋은 모델을 찾는 앙상블 학습을 사용하였다. 하지만 학습 모델의 수가 많기 때문에 학습 시간이 오래 소요된다.

3. 멀티 청크 버그 정정

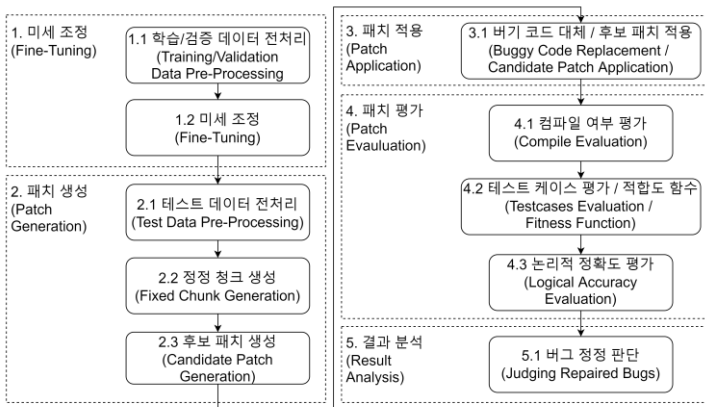


그림 1. 본 APR 기법의 프로그램 정정 과정

3.1. 멀티 청크 버그 정정 개요

본 논문에서 제안하는 APR 기법의 전반적인 과정은 그림 1 과 같고, 미세 조정, 패치 생성, 패치 적용, 패치 평가, 결과 분석의 단계를 포함한다. 또한, 본 APR 기법의 데이터 전처리, 미세 조정, 패치 생성, 패치 적용에 대한 세부적인 과정은 그림 3 과 같다. (1) 미세 조정(Fine-Tuning) : 먼저 학습, 검증 데이터에 대해 버그 블록 전처리를 진행한다. 그 다음 전처리된 버그 블록을 사전 학습된 CodeBERT 모델로 미세 조정한다. (2) 패치 생성(Patch Generation) : 테스트 데이터에 대해 버그 블록 전처리를 진행한다. 테스트 데이터는 미세 조정된 CodeBERT 모델에 입력하면 정정 청크를 생성하기 때문에, 정정 청크는 사용하지 않는다. 미세 조정된 모델에 전처리된 버그 블록으로 입력하면, N 개의 버그 청크는 각각 M 개의 정정 청크를 생성한다. 각 정정 청크별로 순서대로 매핑하여 M 개의 후보 패치 집합 $patch_{candidate}$ 를 생성한다. (3) 패치 적용(Patch Application) : M 번 반복하여 $patch_{candidate}(i)$ (i 번째 후보 패치)를 각 버그 청크별 버그 위치에 맞게 버그

코드를 대체한다. (4) 패치 평가(Patch Evaluation) : 대체가 완료되면, $patch_{candidate}(i)$ 에 대해 그림 2 와 같이 3 단계에 걸쳐 패치 평가를 진행한다. 첫 번째는 $patch_{candidate}(i)$ 의 컴파일 여부를 평가하며, 컴파일 불가능하면 F(Failed), 가능하면 C(Compliable)의 결과를 가진다. 두 번째, 결과가 C 인 $patch_{candidate}(i)$ 의 적합성을 평가하며, 수식 1 의 적합도 함수(Fitness Function)를 적용하여 평가한다. $|Testcases|$ 는 테스트 케이스 총 개수를, $|Testcases_{passed}|$ 는 $patch_{candidate}(i)$ 가 테스트 케이스 통과 개수를 의미한다. $|Testcases_{passed}|$ 가 증가할수록 적합도 함수의 값은 1.0 에 근접해지며, 적합도 함수의 값이 1.0 일 때 적합하다고 평가하며 P(Plausible)의 결과를 가진다. 마지막으로, 결과가 P 인 $patch_{candidate}(i)$ 의 논리적 정확성을 평가한다. 사람이 수작업으로 직접 평가하고, 논리적으로 정확하면 CR(Correct)의 결과를 가지게 된다. (4) 결과 분석(Result Analysis) : 각 프로젝트의 F 를 제외한 평가 결과별 버그 정정 개수를 계산한다. 만약 하나의 모듈에 평가 결과가 CR 인 $patch_{candidate}(i)$ 가 2 개 이상이면 그 모듈은 정정되었다고 판단한다.

$$Fitness(patch_{candidate}(i)) = \frac{|Testcases_{passed}|}{|Testcases|}$$

수식 1. 적합도 함수

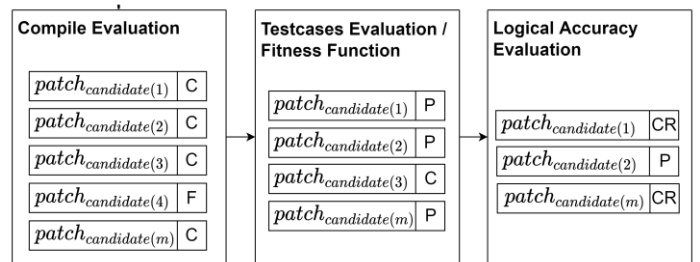


그림 2. 본 APR 기법의 후보 패치 평가

3.2. CodeBERT 모델

본 기법에 Copilot, CodeGPT, GPT-2, GPT-3 등과 같은 GPT 모델도 사용 가능하다. 하지만 GPT 모델은 토큰 생성 정보만 반영하는 디코더만 사용하여 토큰의 문맥 정보를 잘 반영하지 못해서, [6]과 같이 문맥 정보를 학습하는 별도의 NMT 모델을 추가하는 단계가 필요하다. 그로 인해 모델 구성이 거대해지고 복잡해지며 학습 시간이 오래 소요된다. 본 기법은 이러한 면을 고려하여 GPT 모델보다 효과적인 BERT 모델을 사용하며, 그 중 대량의 소스 코드 데이터 집합으로 사전 학습(Pre-Training)시킨 BERT 인 CodeBERT[3]를 사용한다. CodeBERT 는 소스 코드 데이터에 최적화된 학습을 위해 소스 코드 빈칸을 채우는 과정인 MLM(Masked Language Modeling)과 소스 코드의 교체 여부를 예측하는 과정인 RTD(Replaced Token Detection)으로 사전 학습을 진행한다. 본 연구에서는 사전 학습된 모델에 추가 학습 데이터로 버그 블록을 입력하여 미세 조정을 수행하고 패치를 생성한다. CodeBERT 를 사용하면 소스 코드 데이터의 학습 문제, 학습 데이터 양이 적은 문제, 학습되지

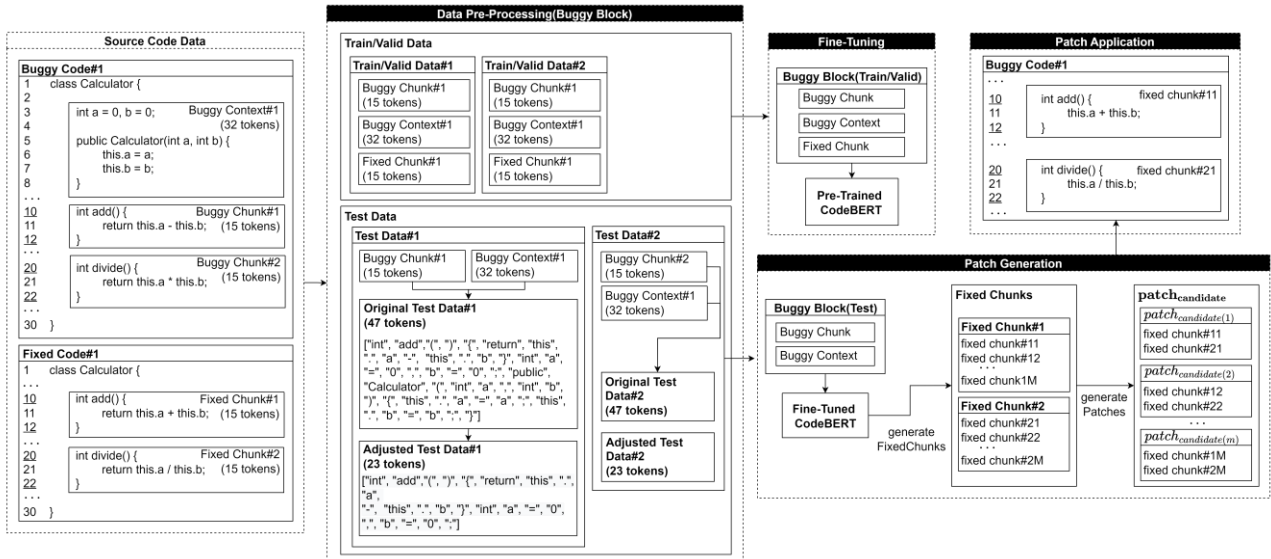


그림 3. 본 기법의 CodeBERT 기반 멀티 청크 버그 정정

하는 토큰으로 발생하는 OOV(Out-Of-Vocabulary) 문제를 해결할 수 있다.

3.2. CodeBERT 기반 멀티 청크 버그 정정

멀티 청크 버그 정정에 기계 학습을 적용하면 발생할 수 있는 문제점과 본 APR 기법에서 제시하는 해결 방법은 그림 3과 같다. (1) 데이터 전처리 : 그림 1의 2.2단계 정정 청크 생성은 테스트 데이터를 사용한다. 일반적으로 테스트 데이터는 버그 청크, 버그 컨텍스트를 하나의 데이터로 구성하여 전처리한다. 전처리 시 각 버그 청크별로 버그 컨텍스트가 중복되는 경우가 발생한다. 그림 3의 "Buggy Code#1" 안에 "Buggy Chunk#1"와 "Buggy Chunk#2"가 있고, "Buggy Context#1"이 중복되는 경우가 발생하여 <"Buggy Chunk#1", "Buggy Chunk#2", "Buggy Context#1">로 테스트 데이터를 구성하게 되면, 버그 청크들을 독립적으로 다룰 수 없어 동일한 정정 청크를 생성하는 문제점이 발생할 수 있다. 그래서 버그 불력은 버그 컨텍스트가 겹치는 경우, 각 버그 청크별로 중복된 버그 컨텍스트가 발생해도 중복을 허용하여 전처리한다. 버그 불력은 그림 3의 "Buggy Code#1" 안에 "Buggy Chunk#1"와 "Buggy Chunk#2"가 중복되는 "Buggy Context#1"가 발생하면, 테스트 데이터를 <"Buggy Chunk#1", "Buggy Context#1">, <"Buggy Chunk#2", "Buggy Context#1">로 구성하여 전처리한다. (그림 1의 1.2단계 미세 조정에서 사용하는 학습, 검증 데이터도 동일하게 전처리한다.) (2) 최대 토큰 수 : 그림 2의 사전 학습된 모델과 미세 조정된 모델은 소스 코드를 토큰화하여 입력한다. 그래서 그림 1의 1.1단계 학습/검증 데이터 전처리와 2.1단계 테스트 데이터 전처리에서 토큰 수를 고려해야 한다. 일반적으로 공백을 제외한 예약어, 식별자 외에 온점(.), 반점(.), 따옴표(", '), 콜론(:), 세미콜론(;), 괄호((,), [,], {, }), 연산자 등을 하나의 토큰으로 취급한다. 그로 인해 소스 코드의 토큰 수가 증가하게 되고, 특수 토큰을 사용한다면 더욱더 증가하게

Patch Generation

1. Input:

- P : Patches(3D)
- BC : Buggy Chunks(2D) FC : Fixed Chunks(3D)
- K : Patch Size
- N : Buggy Chunk Size per Patch
- M : Fixed Chunk Size per Buggy Chunk

2. function generateFixedChunks:

```

model.eval()
for k = 1 to K
  for i = 1 to N
    buggy_chunk = BC[k][i]
    for j = 1 to M
      tokens = tokenizer.tokenize(buggy_chunk)
      predictions = model.predict(tokens)
      fixed_chunk = tokenizer.decode(predictions)
      FC[k][i][j] = fixed_chunk
    end do
  end do
end do
    
```

3. function generatePatches:

```

for k = 1 to K
  for i = 1 to N
    for j = 1 to M
      P[k][j][i] = FC[k][i][j]
    end do
  end do
end do
    
```

그림 4. 그림 1의 2단계 패치 생성의 의사코드

된다. 토큰 수가 증가하면 미세 조정과 정정 청크 생성에 대한 시간과 성능이 저하되는 문제가 발생한다. 또한 CodeBERT 모델의 경우 최대 512 토큰까지 사용 가능한데, 해당 토큰 수를 넘어버리면 나머지 토큰이 절단되는 문제가 발생한다. 그래서 버그 불력은 데이터를 버그 청크, 버그 컨텍스트 순으로 구성하고, 절단되기 전 토큰 수를 확인하여 버그 컨텍스트, 버그 청크 순으로 토큰 수를 조정하여 데이터 품질을 유지한다. 만약 그림 3에서 최대 25 토큰까지 사용

표 1. 다른 연구와의 비교

	본 APR 기법	SequenceR[1]	DLFix[2]	CoCoNut[5]	CURE[6]
학습 모델	버그 블록 기반 NMT, CodeBERT	NMT, 복사 메커니즘	트리 기반 인코더-디코더	문맥 인지 전략 기반 NMT, CNN 기반 인코더, 앙상블 학습	코드 인지 전략 기반 NMT, GPT, 앙상블 학습
전처리 방법	버그 블록	버그 컨텍스트 추상화	트리 기반 전처리	단어 수준 토큰화	단어 수준 토큰화, 서브워드 수준 토큰화
멀티 청크 버그 정정 여부	○	×	×	○	○
최대 토큰 수	512	1,000	제한 없음	설명 없음	1,024
학습 모델 수	1	1	1	100	100
OOV 대처 여부	○	△(일부 대처)	설명 없음	○	○

가능할 때, 기존 "Test Data#1"이 47 토큰이면, "Buggy Context#1"를 3번 라인의 토큰만 사용하여 토큰 수를 조정한다. (3) 후보 패치 생성 : N 개의 버그 청크는 각각 M 개의 정정 청크를 생성한 뒤, 그림 1의 2.3단계 후보 패치 생성을 진행한다. 만약 생성한 정정 청크를 모두 조합하여 M^N 개의 후보 패치를 생성할 시, 후보 패치 생성 시간과 버그 정정 시간이 많이 소요되는 문제가 발생한다. 본 APR의 기법은 그림 4와 같이 generateFixedChunks 함수로 N 개의 버그 청크는 각각 M 개의 정정 청크를 생성하고, generatePatches 함수로 순서대로 매핑하여 M 개의 후보 패치를 생성한다. 그 결과, 그림 2의 $patch_{candidate(1)}$ 은 "fixed chunk#11", "fixed chunk#21"로 구성된다. 그 후 "Buggy Code#1"의 빨셈하는 add 함수인 "buggy chunk#1"와 곱셈하는 divide 함수인 "buggy chunk#2"의 위치에 $patch_{candidate(1)}$ 의 덧셈하는 add 함수인 "fixed chunk#11", 나눗셈하는 divide 함수인 "fixed chunk#21"로 대체한다 (그림 3).

4. 토의

본 연구는 버그 블록과 CodeBERT를 적용하는 APR 기법을 제안한다. 버그 블록은 버그 컨텍스트의 중복과 최대 토큰 수를 전처리하며, 사전 학습된 CodeBERT를 이용함으로써 양의 학습 데이터 문제를 극복한다. 또한 CodeBERT 모델은 저빈도 단어를 여러 서브 워드로 나누는 워드 피스 임베딩(Word-Piece Embedding)으로 사전 학습과 미세 조정을 수행하기 때문에 OOV 문제를 극복한다.

표 1은 본 APR 기법과 다른 연구 간의 비교이다. 비교 베이스라인은 SequenceR[1], DLFix[2], CoCoNut[5], CURE[6]이고, 모두 기계 학습을 사용한다. 모든 베이스라인보다 더 적은 토큰 수, 같거나 더 적은 모델을 사용한다.

또한, 대부분의 기계 학습 기반 APR 기법은 모든 베이스라인과 같이 싱글 청크 버그 정정에 집중되어 있다. [5], [6]는 정정 결과에는 멀티 청크 버그 정정이 포함되어 있지만, N 개의 버그 청크에 대해 몇 개의 정정 청크를 생성하고, 각각 정정 청크를 조합하여 몇 개의 후보 패치를 생성하는지에 대한 정확한 설명이 없다.

5. 결론

본 논문에서는 버그 블록과 CodeBERT 모델을 적용하는 APR 기법을 제안하였다. 미리 사전 학습된 CodeBERT 모델에 전처리한 버그 블록으로 미세 조정을 진행한다. 미세 조정된 모델을 통해 버그 청크 별로 정정 청크를 생성하고 순서대로 매핑한 뒤 후보 패치 집합을 생성한다. 각각의 후보 패치에 패치 평가를 진행하고, 평가 결과에 맞게 버그를 정정한다. 향후에는 본 버그 정정 기법을 효과적으로 구현하는 방향으로 정정 연구를 계속 진행할 예정이다.

6. Acknowledgement

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 한국연구재단 중견연구자지원사업의 지원을 받아 수행된 연구임(NRF-2020R1A2B5B01002467).

참고 문헌

- [1] Z. Chen, S.J. Kommrusch, M. Tufano, L.N. Pouchet, D. Poshyanyk, and M. Monperrus, "SequenceR: Sequence-to-Sequence Learning for End-to-End Program Repair", IEEE Transactions on Software Engineering, vol. TBD, pp. 1-17, 2021.
- [2] Y. Li, S. Wang, and T.N. Nguyen, "DLFix: Context-based code transformation learning for automated program repair", In proc. of the 42nd ACM/IEEE International Conference on Software Engineering (ICSE), pp. 602-614, 2020.
- [3] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, and M. Zhou, "CodeBERT: A Pre-Trained Model for Programming and Natural Languages", In proc. of the 2020 Conference on Empirical Methods in Natural Language Processing(EMNLP), pp. 1536-1547, 2020.
- [4] S. Saha, R.K. Saha, M.R. Prasad, "Harnessing Evolution for Multi-Hunk Program Repair", In proc. of the 41st ACM/IEEE International Conference on Software Engineering (ICSE), pp. 13-24, 2019.
- [5] T. Lutellier, H.V. Pham, L. Pang, Y. Li, M. Wei, and 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 (ISSTA), pp. 101-114, 2020.
- [6] N. Jiang, T. Lutellier, and L. Tan, "CURE: Code-Aware Neural Machine Translation for Automatic Program Repair", In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 1161-1173, 2021.

모바일 어플리케이션 자동화 테스트를 위한 딥러닝 기반 UI 경로 지도의 생성

방혜원¹⁽⁰⁾, 박정민², 한영준*

¹⁽⁰⁾, *숭실대학교 지능시스템학과, ²숭실대학교 AI 융합학부

banghw0815@soongsil.ac.kr, pj4142@naver.com, young@ssu.ac.kr

Construction of UI Path Map based on Deep Learning for Automation Test of Mobile Applications

Hyewon Bang¹⁽⁰⁾, Jungmin Park², Youngjoon Han*

Soongsil University

요 약

본 논문에서는 모바일 어플리케이션 자동화 테스트를 위한 딥러닝 기반 UI 경로 지도 생성을 제안한다. 기존의 연구들은 어플리케이션 각 화면의 객체들을 자동으로 검출하여 정보를 추출하지만, 이를 기반으로 사용자가 테스트 과정을 직접 구성해야 하는 반자동화 형태로 연구되고 있다. 객체 및 문자 인식 딥러닝 모델을 이용하여 UI 컴포넌트 인식과 화면 ID 생성, UI 경로 지도 생성까지의 일련의 과정이 자동으로 이루어지도록 프로그램을 구현하였다. 화면 유사성 측정 실험을 통해 UI 경로 지도 생성의 성능을 검증한다.

1. 서 론

최근 스마트폰의 보급으로 소형 단말기 소프트웨어 이용이 넓게 확대됨에 따라 모바일 어플리케이션 이용이 증가하고 있다. 이에 따라, 모바일 어플리케이션의 품질을 결정하는 UI 기능테스트는 어플리케이션 개발에서 굉장히 중요한 부분 중 하나이다. UI 기능테스트를 수동으로 수행할 때 소프트웨어의 복잡도로 인하여, 전체 기능에 대한 테스트 및 테스트를 진행한 기능이 무엇인지 파악이 어렵기 때문에 이는 개발 비용이 상승하게 되는 주요 요인이 될 수 있다. 이를 해결하기 위해 시간과 비용 감소 및 인적 오류 감소 등의 장점을 가진 자동화 테스트 방식의 필요성이 대두되고 있다[1]. 또한 소프트웨어 생명주기를 이용한 자동화 테스트[2, 3]는 기존에 널리 사용되던 소프트웨어 생명주기 모델을 통해 효율성 측면에서의 자동화 테스트 방식의 필요성을 보였다.

이러한 어플리케이션 UI 테스트 자동화의 요구로 인해 다양한 분야 및 방식으로 기법이 제안되었다. 최근 딥러닝 연구가 활발히 진행됨에 따라, 이를 이용한 자동화 테스트 연구가 진행되었다. 딥러닝 모델을 통해 자동으로 정보를 가공하고, 가공한 정보를 이용하여 트리 기반 테스트 스크립트를 작성하는 딥러닝 기반 GUI 테스트 자동화 프레임워크가 제안되었다[4]. 하지만 이 기법의 테스트 과정은 사용자가 직접

구성해야 하며, 획득한 어플리케이션 화면 정보의 중복을 검증할 수 없으므로 트리 모델이 무한히 깊어지는 문제가 발생할 수 있다.

이러한 문제를 해결하기 위해, 본 논문에서는 딥러닝 모델을 통해 어플리케이션의 각 화면을 분석하고 이를 저장하는 노드 및 이들로 구성되는 경로 지도 생성 기법을 제안한다. 경로 지도는 유한 그래프로 구성되어 화면의 객체들을 에지로 활용한다. 또한 노드의 중복성 문제를 해결하기 위해 화면의 UI 컴포넌트 정보를 가공하여 설명자(Descriptor)를 생성하고 이를 화면 ID로 정의한다. 화면 ID는 유한 그래프의 노드로써 사용된다.

2. 제안하는 방법

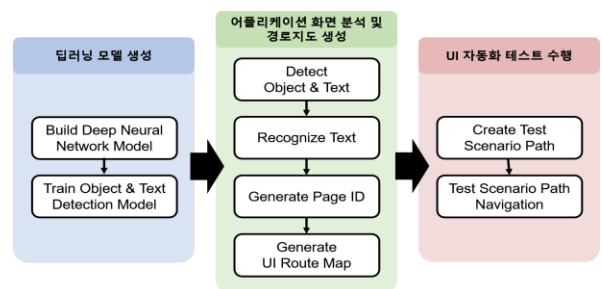


그림 1 딥러닝 기반 UI 자동화 테스트 프레임워크

제안하는 방법은 그림 1과 같이 3단계로 구성되어

있다. 딥러닝 모델 생성 파트에서는 모바일 어플리케이션 화면 내의 UI 컴포넌트 인식을 위해 딥러닝 모델을 선정하고 데이터베이스를 구축 후 학습을 진행한다. 어플리케이션 화면 분석 및 경로 지도 생성 파트에서는 딥러닝 학습을 통해 얻은 모델을 이용하여 모바일 어플리케이션 화면 내의 UI 컴포넌트를 인식한다. 인식한 UI 컴포넌트는 화면 ID 생성 및 경로 지도 생성을 위하여 사용되며 해당 파트를 완료하면 경로 지도를 획득할 수 있다. 마지막 UI 자동화 테스트 수행 파트에서는 이전단계에서 획득한 경로 지도를 이용하여 사용자 테스트 시나리오를 작성하고, 모바일 어플리케이션 테스트를 진행한다. 본 연구는 효율적인 경로 지도 생성을 제안하고 실험을 통해 검증한다.

2.1 UI 컴포넌트 인식을 위한 딥러닝 모델

어플리케이션 화면 내의 컴포넌트 검출과 인식을 위한 딥러닝 모델을 훈련한다. 본 논문에서는 FPN(Feature Pyramid Network)[5]을 이용한 Faster R-CNN[6] 모델을 사용한다. Faster R-CNN은 타 객체 검출 모델과 비교하여 수행시간은 비교적 느리지만 높은 정확도를 보장한다. 텍스트의 경우 Faster R-CNN에서 낮은 검출 성능을 보이기 때문에 이를 보완하기 위해 CRAFT[7]를 이용하여 텍스트의 위치를 추정한다. CRAFT 모델은 문자의 위치 검출은 가능하지만, 해당 문자를 추출하지 못한다. 본 논문에서는 tesseract OCR[10]을 이용하여 문자 검출을 수행한다.

어플리케이션 화면 내의 컴포넌트를 이미지로 인식하기 위하여 어플리케이션 화면에서 획득한 이미지 및 텍스트로 이루어진 UI 학습 데이터를 구축한다. 모바일 어플리케이션의 UI 컴포넌트의 경우 데이터 셋의 양이 적기 때문에 이를 보완해주기 위하여 그림 2와 같이 모바일 어플리케이션 디자인 데이터 셋인 RICO 데이터 셋[11]을 배경으로 모바일 어플리케이션 컴포넌트를 무작위로 배치하여 데이터 증강을 수행한다.

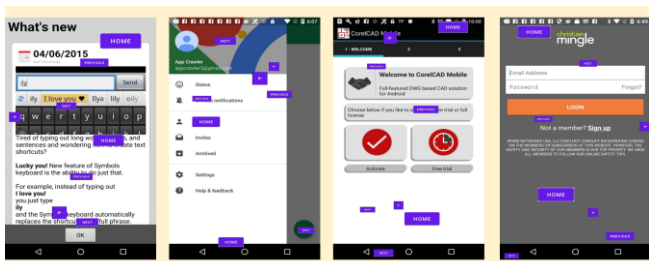


그림 2 컴포넌트를 무작위 배치한 학습 데이터 셋

2.2 모바일 어플리케이션 화면 ID

모바일 어플리케이션의 경우 웹사이트의 URL과 같은 고유한 정보를 가지고 있지 않기 때문에 이를 정의하는 ID가 필요하다. 현재 화면의 컴포넌트들은 이전 분석했던 화면과 연결될 수 있으므로, 이전의 화면으로 돌아갈 수 있다. 이 경우, 이전 화면에 대한 추가적인 노드의 생성을 예방하기 위해 화면을 설명하는 설명자를 생성하고, 이를 통해 이전 화면들과의 유사성을 판단한다. 딥러닝 모델들을 통해 검출한 UI 컴포넌트 정보는 화면의 구성을 설명하는 대표적인 정보가 된다.

본 논문에서는 검출된 UI 컴포넌트들의 문자를 OCR을 통해 추출하고, 사전에 정의된 단어들의 빈도수를 측정하여 벡터화된 ID로 표현한다. 각 단어의 빈도수는 벡터 내 다른 차원으로 표현되기 때문에, 해밍 거리를 통해 각 화면의 유사성을 판단할 수 있다. 화면 ID를 생성하기 위해, 어플리케이션 개발단계에서 사용한 단어들을 수집하고, 사용자는 이를 통해 주요 사용 단어들을 사전에 정의한다. 본 논문에서는 표 1과 같이 단어 사전을 구성하였다.

표 1 사용자 정의 단어 사전

key	value	key	value
First	0	First	0
Fragment	1	Fragment	1
Hello	2	Hello	2
Next	3	Next	3
Third_fragment	4	Third_fragment	4
Home	5	Home	5
Previous	6	Previous	6
Left_arrow	7	FourFragment	7
		FiveFragment	8
		SixFragment	9
		SevenFragment	10
		EightFragment	11
		Second	12

이후 사전에 정의해 둔 단어 사전을 이용하여 문장 내의 단어 빈도수를 검사하여 벡터화한다. 측정된 단어 빈도수는 벡터화된 화면 ID로 표현이 된다. 생성된 화면 ID 들을 이용하여 해밍 거리를 계산하고, 이 값을 화면 유사도 측정에 사용한다. 해밍 거리는 수식 (1)과 같다.

$$d_{\text{hamming}} = \sum_{i=0}^{L-1} |x_i - y_i| \tag{1}$$

식 (1)의 L 은 화면 ID 벡터의 크기이다. x 와 y 는 유사도를 측정하고자 하는 두 개의 화면 ID이고, x_i 와 y_i 는 각각의 화면 ID의 벡터 위치이다. 유사도 측정 결과를 이용하여 화면을 구분하고, 원하는 유사도 임계값을 정하여 유연하게 화면을 구분한다. 향후, 화면 ID를 이용하여 모바일 어플리케이션 내의 동적 광고 UI 컴포넌트에 대해서 대응할 수 있다.

2.3 UI 경로 지도

UI 경로 지도는 어플리케이션 화면의 ID를 노드, UI 컴포넌트의 정보를 에지로 정의하여 그래프로 생성한다. 어플리케이션은 각 컴포넌트로 연결된 화면에서 직전의 화면으로 돌아가는 기능이 없는 경우를 고려해야 한다. 이를 위해 본 논문에서는 유향 그래프를 사용하여 경로 지도를 생성한다. 유향 그래프는 노드를 연결하는 에지가 방향성을 가지며 컴포넌트가 어떤 화면으로 연결되는지를 표현할 수 있다. 각 화면의 정보를 이용하여 노드를 생성한 후, 해당 화면에서 인식된 UI 컴포넌트를 제어하여 그림 3의 UI 경로 지도 생성 흐름도를 따라 어플리케이션의 다음 화면으로 이동하고, 해당 화면을 새로운 노드로 생성하는 작업을 반복한다. 모바일 어플리케이션의 모든 경로를 탐색한 이후 생성된 경로 지도는 JSON 파일로 최종 저장된다.

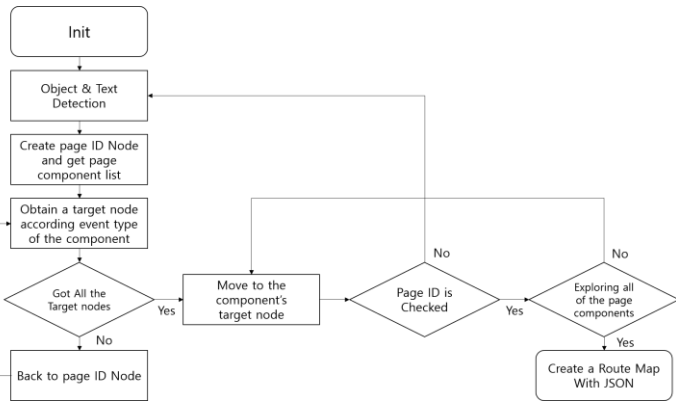


그림 3 UI 경로 지도 생성 흐름도

생성된 JSON 파일은 해당 화면 ID를 노드 이름으로 하여 페이지에 소속된 컴포넌트를 담고, 컴포넌트의 요소는 컴포넌트 이름, 이벤트 타입, 컴포넌트의 좌표, 타겟 노드로 이루어진다. 컴포넌트의 좌표는 딥러닝 모델을 통해서 얻어지며, 타겟 노드는 컴포넌트를 클릭했을 때 나오게 되는 화면을 타겟 노드로 작성한다.

3. 실험 및 결과

제안하는 기법의 평가를 위해 윈도우에서 모바일 어플리케이션 자동화 테스트 도구인 Appium[8]과 안드로이드 스튜디오의 에뮬레이터를 이용하여 모바일 어플리케이션 UI 테스트 환경을 구축한다. 프로그래밍 언어로 python을 사용하였으며 python으로 작성된 스크립트는 Appium 클라이언트로써 Appium 서버와 연결되어 통신하게 된다. 이후, 안드로이드 에뮬레이터와 연결되어 테스트를 수행한다.

테스트를 위해 안드로이드 스튜디오를 이용하여 모바일 어플리케이션 두 개를 제작하였다. 모바일 어플리케이션은 클릭이 가능한 컴포넌트들로 구성되어

있으며, 표 2와 같은 이동 경로를 가진다. 본 논문에서는 클릭 컴포넌트만을 사용하여 실험을 진행하였다.

표 2 모바일 어플리케이션 경로

Node	UI	Previous	Next	Home
1	X	2	X	
2	1	3	X	
3	2	X	1	

(a) (b)

표 2(b)의 어플리케이션 이미지에 대해서 임의의 사각형을 무작위 배치하여 오 검출을 확인하고, 화면 ID 유사도 측정 실험을 진행하였다.

3.1 잡음 추가를 통한 UI 검출 실험

어플리케이션의 화면에 무작위의 잡음을 추가한 후 UI 검출 실험을 진행하였다.

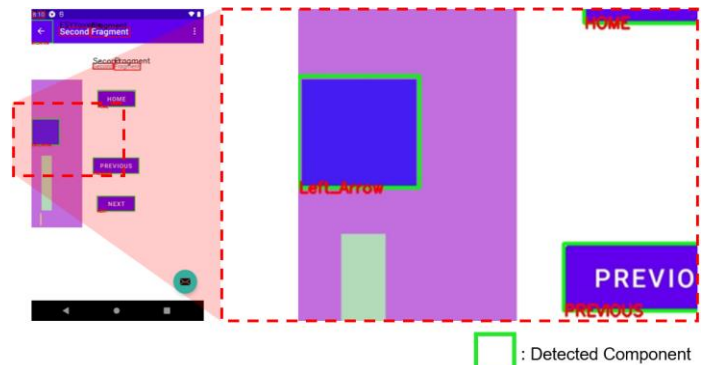


그림 4 UI 오 검출 예시

표 3 무작위 잡음 추가 후 UI 검출 실험 결과

화면 1개 잡음 Rect	총 UI 수	총 잡음 Rect 수	Sample 수	오 검출 UI 수	검출 정확도
0	30	0	30	0	100.00%
1	3000	800	3800	87	97.71%
3	300	240	540	22	95.92%
4	300	320	620	30	95.16%
5	300	400	700	33	95.29%

검출 실험 결과는 표 3에서 보여준다. 잡음이 존재하지 않은 어플리케이션 환경에서의 검출 정확도는 100%의 성능을 보였다. 더 복잡한 환경에서의 실험을 위하여 잡음을 추가하였고, 잡음을 추가한 환경에서도 약 95%의 성능을 확인하였다. 그림 4는 UI 오검출의 예이다. UI 컴포넌트의 색상정보 혹은 크기가 유사할수록 오검출이 큰 것을 확인할 수 있다.

3.2 화면 ID 유사도 측정을 통한 경로 지도 생성 실험

각 화면 ID는 노드끼리의 연결을 설명하는 정보이기

때문에, 경로 지도 생성의 정확성은 화면 ID 유사도 측정을 통해 판단할 수 있다. UI 검출 실험 시 사용한 이미지를 이용하여 ID를 생성하고, 생성된 화면 ID와 목표 화면 ID의 유사도를 측정하였다.

표 2(b) 어플리케이션의 각 화면에 대해서 실험을 진행했으며, 정상적인 화면 1개와 무작위 잡음이 추가된 화면 10개를 이용하여 실험을 진행하였다.

표 4 유사도 측정 실험 결과

분류 화면 개수	유사도 불일치 개수
80	6

실험 결과 텍스트의 경우 오검출되어도, 단어 사진에 의해 화면 ID 생성에 영향이 없지만, UI 컴포넌트의 객체가 오검출되면 화면 ID 생성에 영향을 주어, 같은 화면을 다른 화면으로 인식하는 것을 확인하였다.

4. 결론 및 향후 연구

본 연구에서는 딥러닝 기반 모바일 어플리케이션 UI 자동화 테스트를 위한 경로 지도 생성 기술에 대해 제안하였다. 모바일 어플리케이션 UI 자동화 테스트 구현에 대한 알고리즘을 제시하고, 실험을 통해 화면 ID의 유효성을 검증하고 최종적으로 UI 경로 지도의 자동 생성을 실험하였다.

향후 UI 컴포넌트의 인식 성능을 향상시키고, 다양한 UI 속성 및 상업적 모바일 어플리케이션에 관해 경로 지도 생성을 확장할 계획이다.

Acknowledgement

“ 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음”(2018-0-00209)

참고 문헌

[1] 김성용, 민대환, 임성택, 채봉수, “SW 테스트 자동화구현을 위한 핵심성공요인에 관한 연구: 웹 어플리케이션 소프트웨어를 중심으로”, JITAM, 제27권, 제6호, pp.101-116, 2020.

[2] 안정우, 양호 소프트웨어 개발 생명주기에 대한 자동화된 테스트 체계 연구, 국민대학교 대학원, 박사학위논문, 2020.

[3] 최지훈, 박구락, 박원철, “V모델 기반의 테스트 자동화설계 모델에 대한 연구,” 한국컴퓨터정보학회 학술발표논문집, 제28권, 제2호 pp.593-594, 2018.

[4] 김종우, 허운제, 박예지, 이지민, 손준익, 유준범, “ 딥 러닝 기반 안드로이드 GUI 테스트 자동화 프레임워크”, 한국소프트웨어종합학술대회 논문집, 1737-1739, 2018

[5] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan,

and ‘S. Belongie. Feature pyramid networks for object detection. In CVPR, 2017.

[6] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.

[7] Baek, Youngmin, et al. "Character region awareness for text detection." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

[8] <http://appium.io/>

[9] <https://www.selenium.dev/>

[10] <https://github.com/tesseract-ocr/tesseract>

[11] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibsichman, Daniel Afergan, Yang Li, Jeffrey Nichols and Ranjitha Kumar. "Rico: A mobile app dataset for building data-driven design applications." Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology. 2017.

[12] R. Girshick. Fast R-CNN. In ICCV, 2015.

딥러닝 기반 얼굴인식 손실 함수 기술 동향

임현택, 김태윤, 김창석

(주)고스트패스

ht.lim@ghostpass.ai, ty.kim@ghostpass.ai, cs.kim@ghostpass.ai

Research Trend for Deep Learning-Based Face Recognition Loss Function

Hyun-taek Lim, Tae-yoon Kim, Chang-seok Kim

Ghostpass

요 약

최근 인공지능 기술의 발전으로 인공지능을 이용하여 사람의 얼굴을 인식하고 신원을 확인하는 방법에 대한 연구가 진행되고 있다. 얼굴인식 인공지능 기술에서 이전에는 기계학습 방법론의 고전적인 소프트맥스 함수를 통해 예측하고 학습했으나, 최근 이런 소프트맥스 함수를 이용한 학습의 한계가 지적되면서 소프트맥스의 단점을 보완한 다양한 손실함수들이 제안되고 있다. 본 연구에서는 최근 연구된 손실함수들의 성능을 비교하고 향후 연구 방향을 제시하는 것을 목적으로 한다.

1. 서론

최근 인공지능 기술의 발달로 인해 이 인공지능망 구조를 활용하는 객체인식 기술이 매우 빠른 속도로 발전하고 있다.

또 얼굴인식 인공지능에 이런 객체인식 신경망을 적용함으로써 고성능의 얼굴인식이 가능해지고 많은 얼굴인식 연구를 통해 감시 시스템[1, 2], 생체 인증[3] 등과 같은 분야에 적용되고 있다. 이러한 얼굴인식 기술은 다른 객체인식 기술에 비해 상대적으로 높은 성능을 보이지만, 실제 환경에서 취득되는 얼굴 이미지 또는 비디오에서는 포즈, 조명, 해상도, 흐림 등의 다양성으로 인해 여전히 해결해야 할 문제가 많다.

손실함수란 인공지능망이 학습하면서 최소화시키기 위한 목적 함수이며 대부분은 모형의 예측값과 실제값의 차이, 즉 오차로 설정된다. 인공지능망의 학습은 이 오차를 줄이는 것을 목표로 한다.

그러나 지난 연구에서 다중 분류 모형에 있어 일반적으로 사용되는 소프트맥스를 얼굴인식 모형에 사용되는 것에 대한 문제가 제기되었고 이를 해결하기 위한 각도 기반 손실함수가 제안되었다.[4-6]

얼굴인식은 이미지 또는 비디오 내에 존재하는 얼굴 영역을 나누는 영역 검출 단계, 얼굴 영역에서 추출된 랜드마크 정보를 이용한 얼굴 정렬, 보정, 정규화 등의

작업을 하는 전처리 단계, 전처리 된 얼굴로부터 특징을 추출하는 단계, 모형의 학습 및 검증 단계로 나뉜다.

딥러닝 기반 얼굴 인식 기술에서는 대용량의 데이터를 이용하여 특징 추출과 분류 모형을 학습하여 매우 고차원적인 특징을 확보할 수 있다. 그러나 와일드 환경에서의 얼굴인식 정확도는 공개된 대용량 데이터의 결과보다 감소하게 된다.

본 연구는 최근의 연구에서 제안된 손실함수들의 성능 비교를 통해 장단점을 알아내고, 향후 얼굴인식 인공지능에서 사용되는 손실함수 연구의 방향점을 제시하는 것을 목표로 한다.

2. 선행 연구

컨볼루션 신경망을 사용해 안면을 인식하기 위한 연구 중 대표적인 것은 이미지 피라미드와 얼굴인식 상자 예측 그리고 최종적으로 얼굴의 특징점을 예측하는 Multitask Cascade CNN [7]연구가 있었고 모바일 기기에서 컨볼루션 신경망을 활용하기 위해 연산량을 낮추고 효율성을 높이는 Depth-wise 컨볼루션을 이용한 MobileNet을 제안한 연구가 [8-10] 있었으며, 이 MobileNet을 활용해 얼굴인식 알고리즘을 경량화시키기 위한 많은 연구[5, 11-13]가 있었다.

클래스 분류 신경망의 손실함수로 가장 대표적이고

기본적인 방법으로 소프트맥스[14]가 사용될 수 있지만, 소프트맥스가 암묵적으로 벡터들의 각도 분포를 학습함으로써 유클리드 마진 제약을 부과하는 것은 적합하지 않다는 문제점이 발견되어 클래스 간 분산을 증가시키고 클래스 내 거리를 감소시키기 위한 새로운 손실함수 SphereFace가 제안되었다[6].

그러나 SphereFace는 독립적으로 수렴할 수 없으며, 일반적으로 소프트맥스로 최적화된다. 또한, 클래스마다 마진이 달리 조정된다는 점과 단조 증가하지 않는 cosine 함수 때문에 최적화에 어려움이 존재한다[15].

이어서 SphereFace에서 각도에 직접 여백을 추가하는 방법에 대한 연구에서 ArcFace가 제안되었다[5]. ArcFace는 SphereFace와 달리 소프트맥스를 사용하지 않고 수렴할 수 있다.

기존 각도기반 손실함수는 모두 클래스 간 분산을 확대하고 클래스 내 거리를 줄이기 위해 일정한 마진을 추가하여 클래스 간의 각도 최적화를 목표로 하지만 이 손실함수는 마진이 클래스를 고려하지 않고 모두 일정하다는 문제점이 존재한다.

3. 손실 함수

기존의 연구들은 유클리드 기반 최적화로써, 추가적인 패널티를 부여하지 않는다. 또한, 클래스 간 별개의 각도에 관계없이 마진을 상수 값으로 고정하는 방식을 사용한다.

IAM loss[16]는 마진을 상수 값으로 고정하는 대신 추가적인 적응형 패널티 기법을 사용하여 클래스 간 분산을 증가시켰다.

$$L_{IAM} = \frac{1}{N} \sum_{i=1}^N \log \frac{\frac{1}{c-1} \sum_{j=1, j \neq y_i}^c e^{\cos \theta_{ij}}}{\sum_{j=1}^c e^{\cos \theta_{ij}}} \quad (3.1)$$

\mathbf{x}_i 와 \mathbf{W}_j 사이의 각도가 작아지면, \mathbf{x}_i 와 \mathbf{W}_j 간 평균 유사도는 증가하고, 결과적으로 더 큰 IAM loss를 얻는다. 저자는 IAM loss를 기존 소프트맥스 및 음이 아닌 정규화 파라미터 β 를 사용하여 패널티 항목으로 추가할 것을 제안했다.

$$L = L_{base} + \beta L_{IAM} \quad (3.2)$$

L_{base} 는 CosFace, ArcFace와 같은 현재 설정된 손실을 나타낸다. IAM loss와 설정된 손실함수들의 장점을 활용하여 β 를 통해 적응적으로 가중치를 부여하여 더 나은 손실함수에 도달한다.

기존 손실함수 연구에서는 다른 클래스 사이의 마진을 확대하는 데 중점을 두었으나[5, 6, 15, 17], 이는 불균형 공간 분포로 이어질 확률이 높다는 문제가 지적되었고, 이를 해결하기 위한 EqM loss에 대한 연구가 진행되었다[18]. EqM loss는 하드 샘플 마이닝[19]의 활용과 특징 분포를 균등하게 하여 데이터 불균형의 부정적인 영향을 완화하는 것을 목표로 한다.

$$L_{EqM} = \log \frac{1}{1 + \sum_{j \neq y_i} e^{\cos \theta_{ij} - \cos \theta_{i, y_i}} + |\cos \theta_{i, y_i} - t_1| + |\cos \theta_{i, j} - t_2| + t_1 - t_2} \quad (3.3)$$

$\cos \theta_{i, y_i}$ 는 클래스 내 유사성(클수록 좋음)의 척도로

간주하고 $\cos \theta_{i, j}$ 는 클래스 간 유사도(낮을수록 좋음), t_1 은 클래스 간 유사도의 하한, 클래스 내 유사도의 상한, $t_1 - t_2$ 는 오프셋 항이 된다.

EqM loss는 클래스 내 유사성의 하한과 클래스 간 유사성의 상한을 제어하여 클래스 간, 클래스 내 범위 모두 제약한다. 이는 모든 클래스의 특징 분포를 고르게 하고, 데이터 불균형의 부정적인 효과를 완화한다.

기존의 연구에서는 코사인 함수를 통해 각도 값을 추출하여 기울기 곡선이 비단조적이고, 결정 마진의 중첩 부분이 존재하였으나, 이를 해결하기 위해 제안된 QAMFace[20]에서는 이차 함수를 대상으로 최적화하였다. QAMFace에서는 이차 함수에 이어서 클래스 내 거리를 최소화하기 위해 각도 마진을 추가한다. 이후 편향을 제거하고, 가중치에 l_2 정규화를 적용한다.

$$L_{QAM} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{s(2\pi - (\theta_{y_i} + m))}}{e^{s(2\pi - (\theta_{y_i} + m))} + \sum_{j=1, j \neq y_i}^N e^{s(2\pi - \theta_j)}} \quad (3.4)$$

QAMFace는 특징 벡터와 가중치 사이의 각도가 $[0, \pi]$ 에 속할 때 단조 선형 기울기를 가져 빠른 수렴 속도를 가진다.

기존 연구[5, 6, 15]는 클래스 간 및 클래스 내 변동이 일관되지 않은 실제 데이터에 대해 현실적이지 않아 모델의 판별 및 일반화 가능성을 제한할 가능성이 존재한다. ElasticFace[21]는 클래스 분리 가능성에 대한 유연성을 허용하는 탄력적 마진 손실을 제안하여 고정 마진에 대한 제약을 완화한다. 주 아이디어는 정규 분포로부터 임의의 마진 값을 활용하는 것이다.

ElasticFace에서는 기존의 ArcFace, CosFace에 정규분포를 따르는 임의의 마진값을 추가한다.

ElasticFace-Arc의 손실함수 L_{EArc} 는 식

$$L_{EArc} = \frac{1}{N} \sum_{i \in N} -\log \frac{e^{s(\cos(\theta_{y_i}) + E(m, \sigma))}}{e^{s(\cos(\theta_{y_i}) + E(m, \sigma))} + \sum_{j=1, j \neq y_i}^c e^{s(\cos(\theta_{y_i}))}} \quad (3.5)$$

ElasticFace-Cos는 L_{ECos} 로 표기한다.

$$L_{ECos} = \frac{1}{N} \sum_{i \in N} -\log \frac{e^{s(\cos(\theta_{y_i}) - E(m, \sigma))}}{e^{s(\cos(\theta_{y_i}) - E(m, \sigma))} + \sum_{j=1, j \neq y_i}^c e^{s(\cos(\theta_{y_i}))}} \quad (3.6)$$

$E(m, \sigma)$ 은 평균이 m 이고 표준편차가 σ 인 가우시안 분포에서 임의의 값을 반환하는 정규 함수이다. 확률 밀도 함수는 m 부근에서 절정이다[22]. 따라서, ArcFace, CosFace에서 통합할 때 최적의 마진 값을 m 으로 선택한다.

4. 실험 결과

본 장에서는 각 손실함수의 정확성을 비교한다. 학습 데이터셋은 CASIA-WEBFACE[23]과 MS1MV2[24]이 사용되었고, 테스트 데이터셋으로는 LFW[25], CFP[26], YTF[27], IJB-C[28], MegaFace(R) [5, 29]가 사용되었다.

LFW에는 약 13,000개의 이미지가 포함되어 있으며 검증할 6000개의 쌍이 있다. CFP는 정면 대 정면 및 정면 대 프로파일 이미지를 포함한다. YTF에는 1595명의 다른 사람의 3425개의 비디오가 포함되어 있으며 평균 특징을 사용하여 5000개의 비디오 쌍을 테스트한다. IJB-C에는 총

31,334개의 스틸 이미지와 117,542개의 프레임, 11,779개의 3,531개의 신원이 있는 비디오가 있다. MegaFace 는 검증 이미지가 약 100만개 이미지의 올바른 이미지와 일치해야 하는 매우 어려운 데이터이다.

표 1에서는 데이터셋 CASIA-WEBFACE으로 학습한 손실함수들의 정확도를 비교한다. 모든 손실함수에서 Inception-ResNet-V1[30]을 사용하였고 IAM loss의 실험에서 고정 마진 m 을 0.05로 설정했고, EqM loss의 실험에서는 클래스 간 유사도의 하한 t_1 , 클래스 내 유사도의 상한 t_2 을 각각 0.8, 0.3으로 설정하였다. LFW에서는 EqM loss, YTF에서는 IAM loss, CFP에서는 ArcFace가 각각 가장 좋은 성능을 보여주었다.

<표 1> 데이터 CASIA-WEBFACE에서의 손실함수 정확도

Method	LFW(%)	YTF(%)	CFP(%)
Softmax	98.67	91.28	90.47
SphereFace	99.17	92.58	92.57
CosFace	99.06	95.30	93.02
ArcFace	99.20	93.30	93.37
IAM loss	99.20	95.52	93.01
EqM loss	99.95	94.76	92.33

표 2에서는 데이터셋 MS1MV2으로 학습한 손실함수들의 LFW, IJB-C, MegaFace(R)에서의 정확도를 비교한다. ArcFace, ElasticFace-Arc, ElasticFace-Cos에서는 ResNet-18[31]이 사용되었고, QaMFace에서는 ResNet-50이 사용되었다. LFW에서는 ArcFace, ElasticFace-Cos, QaMFace가 모두 동일한 성능을 보여주었으며, IJB-C에서는 ElasticFace-Cos가 가장 좋은 성능을 보여주었고, MegaFace(R)에서는 Elastic-Arc가 가장 좋은 성능을 보여주었다.

<표 2> 데이터 MS1MV2에서의 손실함수 정확도

Method	LFW	IJB-C	MegaFace (R)
ArcFace	99.82	95.6	98.48
QaMFace	99.82	-	-
ElasticFace-Arc	99.80	96.49	98.81
ElasticFace-Cos	99.82	96.57	98.70

Cos

5. 결론

본 연구에서는 딥러닝 기반 얼굴인식 손실 함수 연구 동향과 최근의 연구에서 제안된 손실함수들이 보편적으로 테스트되는 데이터셋 각각에 대해 정확도를 비교하였다.

최근의 손실함수 연구에서 기존의 손실함수에 비해 점진적인 성능 향상이 나타나고 있음을 확인할 수 있었으며, 테스트 데이터셋의 난이도 따라 적절한 학습 데이터셋과 손실함수를 설정하기 위한 기준점을 마련할 수 있었다.

하지만 추후 연구에서 각 손실함수가 같은 조건으로 실험되지 않아 하이퍼파라미터 설정에 따라 다른 성과가 나타날 수 있다는 부분에 대한 고려가 필요하다.

손실함수 연구는 향후 얼굴 인식 알고리즘의 범용적인 성능 향상을 이루는데 효과적인 방안이 될 수 있으며, 본 논문에서 소개된 연구에서 보이는 부분과 마찬가지로 기존의 손실함수에 새로운 항을 추가하거나 다른 얼굴 인식 방법의 학습 과정과 통합하는 것이 유용한 연구를 이끌어 낼 수 있을 것으로 기대된다.

참고문헌

1. Jose, E., et al. *Face recognition based surveillance system using facenet and mtcnn on jetson tx2*. in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*. IEEE. 2019
2. Byeon, Y.-H., et al., *A surveillance system using cnn for face recognition with object, human and face detection*, in *Information Science and Applications (ICISA) 2016*. Springer. p. 975-984. 2016
3. Zulfiqar, M., et al. *Deep face recognition for biometric authentication*. in *2019 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*. IEEE. 2019
4. Wang, J., et al. *Deep metric learning with angular loss*. in *Proceedings of the IEEE International Conference on Computer Vision*. 2017
5. Deng, J., et al. *Arcface: Additive angular margin loss for deep face recognition*. in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019
6. Liu, W., et al. *Sphereface: Deep hypersphere embedding for face recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017
7. Zhang, K., et al., *Joint face detection and alignment using multitask cascaded convolutional networks*.

- IEEE Signal Processing Letters. **23**(10): p. 1499-1503. 2016
8. Howard, A.G., et al., *Mobilenets: Efficient convolutional neural networks for mobile vision applications*. arXiv preprint arXiv:1704.04861. 2017
 9. Sandler, M., et al. *Mobilenetv2: Inverted residuals and linear bottlenecks*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018
 10. Howard, A., et al. *Searching for mobilenetv3*. in *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019
 11. Deng, J., et al., *Retinaface: Single-stage dense face localisation in the wild*. arXiv preprint arXiv:1905.00641. 2019
 12. Trigueros, D.S., L. Meng, and M. Hartnett, *Face recognition: From traditional to deep learning methods*. arXiv preprint arXiv:1811.00116. 2018
 13. Jiang, M., X. Fan, and H. Yan, *Retinamask: A face mask detector*. arXiv preprint arXiv:2005.03950. 2020
 14. Goodfellow, I., Y. Bengio, and A. Courville, *Deep learning*. MIT press. 2016
 15. Wang, H., et al. *Cosface: Large margin cosine loss for deep face recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018
 16. Sun, J., et al., *Inter-class angular margin loss for face recognition*. *Signal Processing: Image Communication*. **80**: p. 115636. 2020
 17. Wang, F., et al., *Additive margin softmax for face verification*. *IEEE Signal Processing Letters*. **25**(7): p. 926-930. 2018
 18. Sun, J., et al., *An equalized margin loss for face recognition*. *IEEE Transactions on Multimedia*. **22**(11): p. 2833-2843. 2020
 19. Lin, T.-Y., et al. *Focal loss for dense object detection*. in *Proceedings of the IEEE international conference on computer vision*. 2017
 20. Zhao, H., et al. *Qamface: Quadratic Additive Angular Margin Loss For Face Recognition*. in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2020
 21. Boutros, F., et al., *Elasticface: Elastic margin loss for deep face recognition*. arXiv preprint arXiv:2109.09416. 2021
 22. Peebles Jr, P.Z., *Probability, random variables, and random signal principles*. McGraw-Hill. 2001
 23. Yi, D., et al., *Learning face representation from scratch*. arXiv preprint arXiv:1411.7923. 2014
 24. Guo, Y., et al. *Ms-celeb-1m: A dataset and benchmark for large-scale face recognition*. in *European conference on computer vision*. Springer. 2016
 25. Huang, G.B., et al. *Labeled faces in the wild: A database for studying face recognition in unconstrained environments*. in *Workshop on faces in 'Real-Life' Images: detection, alignment, and recognition*. 2008
 26. Sengupta, S., et al. *Frontal to profile face verification in the wild*. in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2016
 27. Wolf, L., T. Hassner, and I. Maoz. *Face recognition in unconstrained videos with matched background similarity*. in *CVPR 2011*. IEEE. 2011
 28. Maze, B., et al. *Iarpa janus benchmark-c: Face dataset and protocol*. in *2018 International Conference on Biometrics (ICB)*. IEEE. 2018
 29. Kemelmacher-Shlizerman, I., et al. *The megaface benchmark: 1 million faces for recognition at scale*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016
 30. Schroff, F., D. Kalenichenko, and J. Philbin. *Facenet: A unified embedding for face recognition and clustering*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015
 31. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016

이더리움 스마트 컨트랙트의 업그레이드 전략에 따른 가스 영향도 분석

김진영[○], 김미수, 이은석

성균관대학교 소프트웨어학과[○], 소프트웨어융합연구원, 소프트웨어대학

{danpoong,misoo12,leees}@skku.edu

Analysis of Gas Impacts on Upgrade Strategy of Ethereum Smart Contract

Jinyoung Kim[○], Misoo Kim, Eunseok Lee

Department of Computer Science and Engineering[○], Institute of Software Convergence,

College of Computing and Informatics, Sungkyunkwan University

요 약

이더리움의 스마트 컨트랙트는 탈중앙화 블록체인 네트워크에 배포된 이후에는 코드의 수정이 불가능하다. 따라서 배포 이후에 기능 또는 결함을 수정하기 위한 전략이 필요하다. 이를 위해 배포 후 업그레이드 가능한 컨트랙트 설계 전략들이 연구되고 있다. 그러나 각 전략들이 프로젝트 특징과 가스(gas)에 미치는 영향을 고려하지 않고 적용되고 있다. 가스는 스마트 컨트랙트를 블록체인에 배포하거나 기능을 수행할 때 소모되는 컴퓨팅 파워를 추상화한 개념이다. 즉 이 가스를 기반으로 채굴자에게 금전적인 수수료를 지급하게 된다. 그러므로 각 업그레이드 전략이 가스에 미치는 영향을 고려하지 않는다면, 기능의 업데이트 마다 지속적으로 낭비되는 금전적인 비용이 발생한다. 따라서 본 논문에서는 업그레이드 전략들이 가스 증감에 미치는 영향을 분석하고, 어떠한 전략이 프로젝트에 효과적으로 적용될 수 있는지 보인다. 대표적인 업그레이드 전략인 Proxy와 Factory 기반 전략들을 기반으로 실험한 결과, 최소 71\$ 차이를 확인하였다. 정량적인 결과를 바탕으로 우리는 스마트 컨트랙트에서 업그레이드시 트랜잭션 사용률이 높게 예상되는 상황에서 Factory 전략을 선택하는 것이 경제적으로 더 이득이 됨을 보인다.

1. 서 론

이더리움 블록체인 네트워크에 배포된 스마트 컨트랙트[1]는 미리 정해진 조건이 만족되면 제3자의 개입없이 계약을 실행하고 증명하는 소프트웨어이다. 계약의 수행은 컨트랙트 개발 언어(Solidity, Vyper, Yul등)[2]의 코드로 구현된다. 탈중앙화 네트워크인 블록체인은 영구성, 불변성의 특징을 가지므로 한 번 배포된 스마트 컨트랙트의 내용을 변경할 수 없다. 하지만 결함이 발견되거나 기능을 수정해야 하는 상황이 필연적이기 때문에, 이를 위해 업그레이드 전략[3]들이 연구되었다. 그러나 이 전략들이 소모시킬 수 있는 가스(gas)[4]에 대한 분석과 연구가 부족하다. 가스는 스마트 컨트랙트를 배포하거나 기능을 호출할 때 발생하는 값으로, 곧 이를 승인하여 블록에 기록하는 채굴자에게 금전적인 수수료[5]를 지급하게 된다. 이는 부적절하게 프로젝트에 적용된 전략은 업그레이드 마다 낭비되는 금전적인 비용이 존재함을 암시한다. 따라서 본 연구에서는 전략들의 가스 사용량과 특징을 분석한다. 분석 결과를 바탕으로, 프로젝트의 특성에 따른 업그레이드 전략을 제시한다. 우리가 제시하는

전략을 통해 약 71\$의 비용을 감축시킬 수 있음을 보인다.

2. 스마트 컨트랙트의 업그레이드

2.1. 업그레이드 전략

그림 1은 전략을 적용하지 않고 일반적으로 업그레이드하는 방법을 설명한다. 기능이 변경된 컨트랙트를 재배포 후 새롭게 생성된 주소를 활용하게 된다. 이는 주소의 변경으로 인해 기존 스마트 컨트랙트는 더 이상 사용할 수 없는 상태가 되며, 새로운 주소로 프로젝트를 수정하고 변경을 알리는 추가적인 비용이 발생하게 된다.

기존의 업그레이드 전략들은 데이터와 기능을 담고 있는 스마트 컨트랙트를 분리하여 배포하는 방법으로 컨트랙트 설계 수준에서 적용된다. 현재 Proxy와 Factory 전략이 대표적으로 사용된다.

그림 2는 Proxy를 활용한 전략을 설명한다. 이는 Proxy Contract와 Logic Contract로 나누어 배포한다. Proxy Contract는 데이터를 기록하지만 기능은 구현하지 않고 사용자가 실행한 기능을 Logic

Contract에 위임하는 역할을 수행한다. 이는 Solidity의 delegatecall을 이용하여 구현되며 Proxy Contract의 Context를 유지한다. 결국 Logic Contract에 저장된 기능을 실행 후 결과값에 따라 데이터를 갱신한다. 만약 기능의 수정이 필요한 경우 Logic Contract만 새롭게 배포 후 Proxy Contract에게 주소를 전달하여 업그레이드를 가능하게 한다.

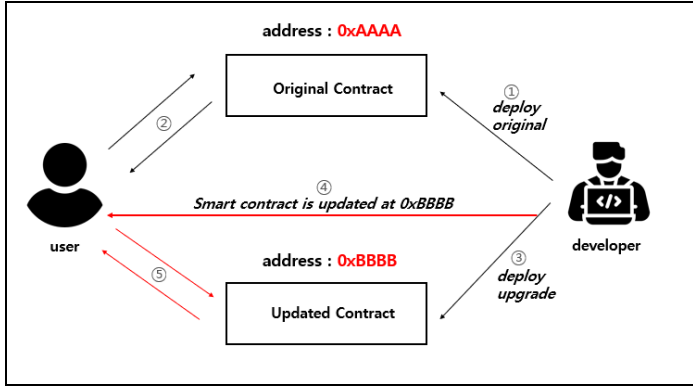


그림 1. 전략을 적용하지 않은 일반적인 업그레이드

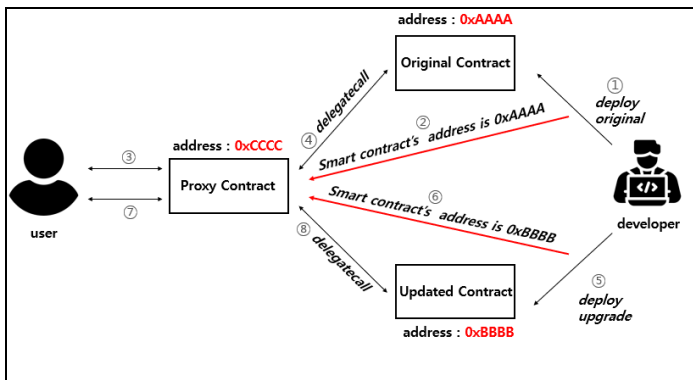


그림 2. Proxy를 활용한 업그레이드

그림 3은 Factory를 활용한 전략을 설명한다. 동일한 주소에 스마트 계약을 재배포하는 목적을 가진다.

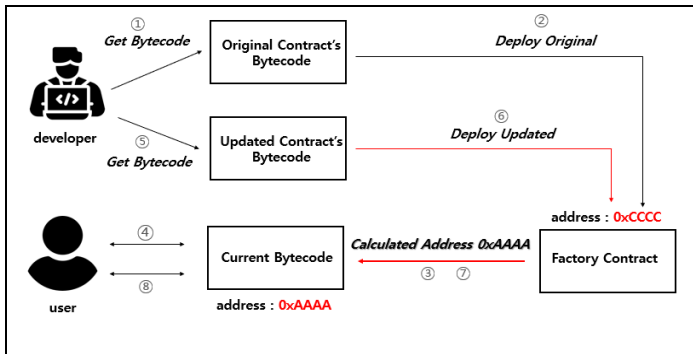


그림 3. Factory를 활용한 업그레이드

이는 Solidity의 create2와 selfdestruct 연산으로 구현된다. 만약 기능의 수정이 필요한 경우 수정된 컨트랙트의 바이트코드를 미리 계산하여 Factory

Contract에 전달한다. 이후 바이트코드 덮어쓰기를 통해 업그레이드를 수행한다.

2.2. 업그레이드에 사용되는 연산 종류

2.2.1. call vs delegatecall

Solidity는 다른 스마트 컨트랙트의 기능을 호출하기 위한 연산인 call과 delegatecall을 가진다. 두 연산의 차이는 실행중인 Context의 주체가 caller 또는 callee인지로 구분된다. 예를 들어, A컨트랙트에서 B컨트랙트의 send 메소드를 실행할 경우이다. call을 사용할 경우 send 메소드는 B컨트랙트의 Context에서 실행되며 B컨트랙트에 상태 변수의 값을 변경한다. delegatecall의 경우 B컨트랙트의 send 메소드 코드를 A컨트랙트의 Context에서 실행하기 때문에 상태값은 A컨트랙트에 반영된다. 이는 위임호출이라고 정의하며 proxy 전략을 구현하기 위해 사용된다.

2.2.2. create vs create2

스마트 컨트랙트를 이더리움 네트워크에 배포하는 것은 create 혹은 create2를 통해 생성된 주소에 바이트코드를 등록하는 행위이다. 그림 4는 create 기능의 원형을 나타내며, create를 호출한 sender의 주소와 nonce값을 이용하여 컨트랙트 주소를 생성한다.

```
// CreateAddress creates an ethereum address given the bytes and the nonce
func CreateAddress(b common.Address, nonce uint64) common.Address {
    data, _ := rlp.EncodeToBytes([]interface{}{b, nonce})
    return common.BytesToAddress(Keccak256(data)[12:])
}
```

그림 4. go-ethereum의 create 기능 원형

그림 5는 create2 기능의 원형을 나타내며, create2를 실행한 sender의 주소와 salt, bytecode를 이용하여 컨트랙트 주소를 생성한다.

```
// CreateAddress2 creates an ethereum address given the address bytes, initial
// contract code hash and a salt.
func CreateAddress2(b common.Address, salt [32]byte, inithash []byte) common.Address {
    return common.BytesToAddress(Keccak256([]byte{0xff}, b.Bytes(), salt[:], inithash[12:]))
}
```

그림 5. go-ethereum의 create2 기능 원형

두 연산은 nonce값 사용유무의 차이를 가진다. 기본적으로 create와 create2는 컨트랙트가 배포되면 nonce값이 1로 설정된다. 이후 배포된 컨트랙트의 주소의 nonce값이 0인지 판단하여 재배포 여부를 결정하게 된다. create는 nonce값을 활용하여 컨트랙트를 생성할 때마다 1씩 자동으로 증가한다. 따라서 항상 다른 주소를 생성하게 된다. 하지만 create2는 salt와 미리 컴파일된 컨트랙트의 바이트코드 값으로 주소를 생성한다.

2.2.3. selfdestruct

스마트 컨트랙트를 파괴하는 명령이며, 블록체인에 등록된 코드를 지울 수 있는 유일한 방법이다. 수행이후

nonce값을 감소시키기 때문에, create2와 함께 삭제된 영역에 새로운 바이트코드를 로딩하여 동일한 주소로 배포를 수행한다.

3. 가스의 정의와 종류

스마트 컨트랙트는 EVM(Ethereum Virtual Machine)위에서 동작하며 실행되기 위해 가스가 필요하다. 가스는 채굴자의 컴퓨팅 연산량을 추상화한 개념으로써 각 연산에 대한 비용이 존재한다. 결국 채굴자는 소모한 가스와 가스당 가격을 통해 수익을 얻는다. 가스의 종류는 크게 두 가지[6]로 나뉘어지며 다음과 같다.

3.1. 트랜잭션 가스

스마트 컨트랙트에 작성된 기능을 호출하여 트랜잭션이 발생한다. 이는 이더리움 네트워크에 대기상태로 등록되어 각 채굴자들의 우선순위 알고리즘을 기반으로 블록에 기록하여 트랜잭션이 승인된다. 이 때, 발생하는 비용을 트랜잭션 가스라고 정의한다. 수행된 코드의 연산을 계산하며 파라미터의 유형, 크기에 따라서 가변적으로 측정된다.

3.2. 배포 가스

계약을 실행하기 위해 스마트 컨트랙트 코드를 이더리움 네트워크에 등록하는 과정이 필요하다. 이 때, 발생하는 비용을 배포 가스라고 정의한다. 배포가스는 트랜잭션 가스보다 높게 측정이 된다.

4. 실험 방법 및 환경

그림 6은 실험에 사용한 스마트 컨트랙트의 pseudocode를 나타낸다.

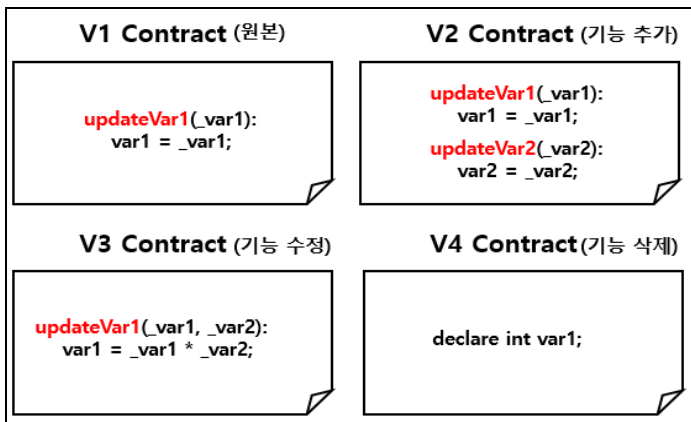


그림 6. 업그레이드에 사용한 스마트 컨트랙트

실험에 사용된 데이터는 업그레이드의 주 목적인 기능 추가, 수정, 삭제로 구성되며 각 전략들에 구현한다. 결과의 객관성을 위해 동일한 시나리오로

평가하며 다음과 같다.

- 1) 수정 이전의 기능을 포함하는 컨트랙트 배포
- 2) 수정 이전의 기능으로 트랜잭션 실행
- 3) 수정 이후의 기능을 포함하는 컨트랙트 배포
- 4) 수정 이후의 기능으로 트랜잭션 실행

컴파일과 트랜잭션, 배포 가스를 측정하기 위해 Truffle Framework[7]를 활용한다. 실험에 사용한 OS는 Windows 10 Home, CPU는 Intel® Core(TM) i7-6700@3.40GHZ, RAM은 16GB이다.

5. 실험 결과

Proxy와 Factory기반 업그레이드 전략을 적용한 이후의 트랜잭션과 배포 가스량을 비교한다. 가스량이 클수록 높은 비용이 소모되어 부정적인 결과임을 의미한다. 모든 결과에서 굵은 표시는 높은 값을 나타낸다. 또한 2021년 12월 5일 기준 가스당 평균 가격인 72Gwei와 1Ether가격인 4,186\$를 기준으로 총 비용을 평가한다. Gwei의 단위와 비용 계산식은 (1)과 같다.

$$(1) \quad 1Ether = 1,000,000,000 \text{ Gwei}$$

$$Cost(\$) = (gas * 72) / (10^9) * 4186$$

5.1. 기능 변경에 대한 가스 변화량

표 1은 Proxy와 Factory전략으로 기능을 추가할 때 발생하는 가스를 비교한다. Proxy는 트랜잭션 가스에서 Factory보다 상대적으로 높은 가스를 소모했다. 반면에 배포 가스에서 Proxy가 더욱 적은 가스를 소모한다. 전체적으로 Proxy가 Factory보다 가스 절감에 효과적이며 비용으로 변환 시 약 86.3\$의 차이를 보인다.

표 1. Proxy, Factory 기능 추가 가스량 비교

	Proxy	Factory	Diff
트랜잭션 가스	24,108	22,402	1,706
배포 가스	293,208	581,277	288,069
총 가스	317,316	603,679	286,363
총 비용(\$)	95.64	181.94	86.3

표 2는 Proxy와 Factory전략으로 기능을 수정할 때 발생하는 가스를 비교한다. Proxy는 트랜잭션 가스에서 Factory보다 상대적으로 높은 가스를 소모했다. 전체적으로 소모된 비용은 Proxy에서 약 98.12\$ 절감 효과를 보인다.

표 2. Proxy, Factory 기능 수정 가스량 비교

	Proxy	Factory	Diff
트랜잭션 가스	28,856	22,969	5,887
배포 가스	308,376	610,972	302,596
총 가스	337,232	633,941	296,709
총 비용(\$)	92.94	191.06	98.12

표 3은 Proxy와 Factory전략으로 기능을 삭제할 때 발생하는 가스를 비교한다. 삭제된 기능은 트랜잭션을 수행할 수 없으므로 결과에서 제외하였다. 추가, 수정과 마찬가지로 Proxy에서 가스 절감 효과를 보인다. 비용으로 변환 시 약 71.32\$의 차이가 존재한다.

표 3. Proxy, Factory 기능 삭제 가스량 비교

	Proxy	Factory	Diff
배포 가스	239,528	476,184	236,656
비용(\$)	72.20	143.52	71.32

5.2. 지속적인 업그레이드에 따른 가스량 비교

앞선 5.1에서 업그레이드 이후 기능을 실행할 때 발생하는 가스를 비교하며 차이가 있음을 보였다. 결과적으로 Factory보다 Proxy 전략이 가스 및 관련 비용 절감 업그레이드에 적합해 보이지만, 해당 결과는 단일 업그레이드에 대한 값이다. 더불어, 트랜잭션 가스는 모든 경우에서 Factory전략이 적은 비용을 소모했다. 결국 업그레이드란 지속적으로 발생하는 것이기 때문에 각각의 특징에 맞는 적합한 프로젝트를 선별하는 것이 중요하므로 선택의 기준을 설명한다.

기본적으로 Proxy와 Factory는 동일하게 새로운 컨트랙트를 배포한다. 하지만 주소의 변경유무에 따라 두 전략의 차이는 존재한다. Proxy는 새로운 컨트랙트를 일반적인 create로 배포하여 획득한 주소를 자신의 상태변수에 재할당하는 *set_address* 기능을 통해 업그레이드를 수행한다. 이후 *delegatecall*을 이용하여 트랜잭션을 위임 실행한다. 반면에 Factory는 새로운 컨트랙트를 배포하기 전에 *selfdestruct* 함수를 통해 바이트코드를 삭제한다. 이후 create2를 이용하여 새로운 바이트코드를 특정 주소에 덮어쓰기 하는 업그레이드를 수행한다. 이후 배포된 컨트랙트에서 직접 트랜잭션을 실행한다. 추가적인 실험에서 Proxy의 *set_address*는 20바이트의 주소값을 파라미터로 전달받아 상태변수에 저장하여 27,766 가스를 소모하는 것을 확인했다. 반면에 Factory의 *selfdestruct*는 13,393 가스를 소모했다. 결과를 바탕으로 트랜잭션의 실행은 배포 이후 지속적으로 수행되기 때문에 실행률이 높게 예상되거나 장기적으로 운영할 준비가 되어있는 프로젝트에 Factory를 적용하면 업그레이드에서 발생하는 비용을 절감할 수 있다. 반대로 업데이트 주기가 짧으며, 트랜잭션 사용률이 적게 예상되거나 단기 프로젝트는 Proxy가 적합함을 의미한다. 만약

적절한 기준 없이 사용된다면 배포과정에서 낭비되는 비용이 발생하거나, 업그레이드 이후 트랜잭션 금액이 증가하여 사용자의 만족성을 해칠 수 있다.

6. 결론

본 논문에서는 스마트 컨트랙트의 업그레이드 전략들에 따라 트랜잭션/배포 가스 및 비용에 미치는 영향을 확인했다. Proxy와 Factory 전략에 따라 차이는 존재했으며 기능을 추가/수정/삭제하는 경우에서 실험하였다. 결국 Proxy는 배포가스를 절감하는데 효과적이며, Factory는 트랜잭션가스를 절감하는 효과를 가짐을 확인했다. 하지만 업그레이드는 지속적으로 발생하는 행위이기 때문에 비용을 절감할 수 있는 적합한 프로젝트를 선별하는 기준이 필요하다. 이를 위해 각 전략의 업그레이드 과정을 상세히 분석하였고, 결과적으로 트랜잭션 실행 빈도에 따라 프로젝트를 선별할 수 있음을 논의하였다. 빈도수가 높게 예상되거나 체계가 잡힌 프로젝트라면 Factory, 빈도수가 낮게 예상되거나 단기적으로 운영할 프로젝트라면 Proxy를 적용하여 비용을 절감할 수 있음을 제안한다. 하지만 두 전략은 저장된 값을 제외하고 기능만 업그레이드 하기 때문에 데이터 유실 가능성이 존재한다. 향후에는, 업그레이드 전략의 데이터 손상 문제를 고려해 연구할 예정이다.

Acknowledge

이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(2019R1A2C2006411).

참고 문헌

[1] BUTERIN, Vitalik, et al. A next-generation smart contract and decentralized application platform. white paper, 2014.

[2] <https://ethereum.org/en/developers/docs/smart-contracts/languages>. 2021, October 17

[3] ZHENG, Gavin, et al. Ethereum Smart Contract Development in Solidity. Springer, 2021.

[4] <https://ethereum.org/en/developers/docs/gas>. 2021, October 12

[5] CHEN, Ting, et al. Towards saving money in using smart contracts. In: 2018 IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER). IEEE, 2018. p. 81-84.

[6] CANFORA, Gerardo, et al. Profiling Gas Leaks in Solidity Smart Contracts. arXiv preprint arXiv:2008.05449, 2020.

[7] <https://www.trufflesuite.com>. 2021, October 15

의심스러운 조건문을 이용한 결함 위치 식별 및 수정 개선

강성민[○] 유신

한국과학기술원

sungmin.kang@kaist.ac.kr, shin.yoo@kaist.ac.kr

Improving Fault Localization and Automated Program Repair with Suspicious Predicates

Sungmin Kang[○] Shin Yoo

KAIST

요 약

본 연구에서는 결함의 위치 및 식별 방법을 효과적으로 알아내기 위한 ‘의심스러운 조건문’을 제안한다. 실패 테스트에서 참이고 성공 테스트에서 거짓인 의심스러운 조건문은 프로그램 상태를 추상화하여, 결함 위치 식별 및 프로그램 수정을 위한 작업에 도움을 줄 수 있다. Siemens 및 Defects4J 데이터셋을 대상으로 한 실험을 통해 의심스러운 조건문이 자동 디버깅에 도움을 줄 수 있음을 보인다.

1. 서 론

자동 디버깅은 컴퓨터의 보조를 통해 사람의 디버깅 수고를 줄이거나 없애는 것을 목표로 한다. 대표적인 자동 디버깅 방법으로는 결함 위치 식별 (Fault Localization, FL) 및 결함 수정 (Automated Program Repair, APR) 방법론이 있다. 결함 위치 식별은 결함이 있는 프로그램 내에서 어느 부분을 개발자가 고쳐야 하는지 찾아주는 것을 목표로 하며 (Sohn, 2017), 결함 수정은 결함이 있는 프로그램을 의도된 행동에 맞게 자동으로 수정하는 것을 목표로 한다 (Kim, 2013).

현재까지 제안된 많은 자동 디버깅 기법은 프로그램의 내부 상태를 쓰지 않는다 (Sohn, 2017) (Kim, 2013). 반면 사람은 프로그램의 상태를 적극적으로 활용하여 디버깅하는 것이 관찰되었다 (Xia, 2016). 이는 프로그램 실행의 양상을 파악하는 데 프로그램 상태를 관찰하는 것이 상당한 도움을 준다는 것과, 더 나아가 프로그램 상태를 이용하여 보다 효율적인 자동 디버깅이 가능하다는 것을 시사한다.

자동 디버깅 기법 가운데 프로그램 상태를 사용하는 방법이 결함 위치 식별 (Jeffrey, 2008) 및 결함 수정 (Mechtaev, 2016) 양쪽에 걸쳐 제안된 바 있다. 그러나 각 방법은 특정 자동 디버깅 분야에 한정된 성능을 보여주었고, 지금까지 제안된 상태 기반 자동 디버깅 방법론이 시너지가 있는지는 밝혀져 있지 않다. 본 논문에서는 프로그램 상태를 사용하는 하나의 방법론으로 결함 위치 식별 및 결함 수정을 동시에 더

효과적으로 할 수 있음을 보이고자 한다.

프로그램 상태를 사용하기 위해, 본 연구에서는 먼저 변수의 대소를 비교하는 등의 **조건문**으로 프로그램 상태를 추상화한다. 이렇게 추상화한 프로그램 상태가 실패 테스트 (Failing test)에서 자주 나타나고 성공 테스트 (Passing test)에서는 자주 나타나지 않는다면, 이를 ‘의심스러운 조건문’이라고 칭한다. 의심스러운 조건문의 형태 및 조건문의 의심스러운 정도에 따라, 어떤 위치에 결함이 있고 결함을 어떻게 수리해야 하는지를 파악할 수 있다. 또한 개발자에게 결함이 일어나는 조건을 제시함으로써, 개발자의 이해를 돕는 부가 효과 역시 기대할 수 있다.

의심스러운 조건문을 이용한 디버깅을 시험하기 위해, Siemens (C) 및 Defects4J (Java) 결함 데이터셋을 이용하여 의심스러운 조건문의 효과를 평가한다. 구체적으로 (⊖) Siemens suite에서 의심스러운 조건문으로 타겟할 수 있는 결함의 위치와 수정 방법을 얼마나 정확히 도출하는지 평가하고, (⊖) Defects4J 데이터셋에서 타겟할 수 있는 결함에 대해서 수정 방법을 정확히 도출하는지 평가한다.

2. 의심스러운 조건문

2.1 의심스러운 조건문을 이용한 디버깅

‘ $a > b$ ’ 연산을 ‘ $a \geq b$ ’로 고쳐야 하는 결함이 있다고 가정하자. 이 경우 잘못된 프로그램 실행은 $a == b$ 일 때 나타나므로, ‘ $a == b$ ’ 꼴의 조건문이 실패 테스트에서 자주 참이고, 성공 테스트에서는 자주 거짓일 것이다. 역으로, ‘ $a > b$ ’ 연산이 소스코드 내에 있을 때 ‘ $a == b$ ’가

의심스러운 조건문이라면, 'a>b' 연산이 잘못되었을 가능성이 높다. 따라서 의심스러운 조건문에 등장하는 변수를 통해 결함 위치 식별 역시 가능하다. 이처럼 각 결함 유형에 대응되는 의심스러운 조건문 끝이 있으리라 기대할 수 있으며, 결함 유형을 파악하여 결함 수정에 이바지할 수 있다.

본 연구에서는 세 가지 유형의 결함을 집중적으로 분석한다. 먼저, 1번 'op1 [>/>=] op2' 결함 유형은 대소를 비교하는 연산에서 off-by-one 에러가 일어나는 결함 유형이다. 이런 경우 '(op1==op2)' 조건문이 의심스러우리라 기대할 수 있으며, 변수끼리 비교되는 위치를 결함 위치로 추정할 수 있다. 다음으로 2번, 조건문 삭제 결함 유형은 조건문을 삭제해야 하는 결함으로, 삭제해야 하는 조건문이 실패 테스트에서 참이고 성공 테스트에서는 거짓이리라 기대할 수 있다. 마지막으로 3번 변수 a를 b로 고쳐야 하는 결함에서는 (a!=b) 조건문이 의심스러우리라 기대할 수 있다. 이 유형의 결함은 결함 유형을 인지했을 때도 결함 위치가 이전 유형에 비해 뚜렷하지 않으나, a 또는 b가 다른 변수에 비해 2번 더 등장하는 시점 (즉, 사용 빈도의 균형이 깨진 시점)을 결함 위치로 생각할 수 있다.

본 연구에서는 의심스러운 조건문이 자동 디버깅에 어떻게 도움을 줄 수 있는지 집중적으로 조명한다. 그러나 자동 디버깅 이외에도 의심스러운 조건문은 개발자의 디버깅을 질적으로 도울 수 있다. 가령 실패 테스트가 일어나는 내부 조건을 밝힘으로써 기존의 자동 디버깅 기법이 제공하기 어려운 정보를 개발자에게 효과적으로 제시할 수 있으며, 더 나아가 자동 디버깅과 결부했을 때 자동 디버깅 방법론을 '설명'하는 하나의 방법으로 기능할 수 있다.

2.2. 의심스러운 조건문 계산

본 연구에서는 의심스러운 조건문을 두 단계에 걸쳐 구한다. 먼저 값을 구할 조건문을 생성하는데, (¬) 소스코드 함수 내에 등장하는 변수와 함수 call의 값을 >, == 등의 연산자를 이용해 비교하는 조건문과 (¬) 함수 내에 원래 등장하는 조건문을 추출한다. 이후 해당 조건문의 값을 출력하는 코드를 원래 코드에 삽입하고, 조건문의 참/거짓 결과를 테스트마다 기록한다. 이러한 참/거짓 기록 결과를 바탕으로, 아래 식을 이용해 조건문 P의 의심스러운 정도를 계산한다.

$$susp(P) = \frac{\text{\# of occurrences in failing tests}}{\text{\# of occurrences in total}}$$

3. 실험

3.1 실험 조건

Siemens suite에서는 의심스러운 조건문을 이용한 결함 위치 식별 및 수정 성능을 평가한다. 따라서

프로그램 내 모든 위치의 프로그램 상태를 수집하여 그 가운데 가장 의심스러운 조건문을 구한다. Defects4J 데이터셋에서는 정확한 결함 위치에서 의심스러운 조건문을 구했을 때, 결함 유형이 정확히 파악되는지 평가한다.

3.2 Siemens suite에서 결함 유형에 따른 위치 식별

Siemens suite에서 앞서 언급한 세 가지 유형에 해당하는 버그 15개를 모아, 결함 위치 식별 성능을 SBFL공식 Ochiai와의 성능을 비교하였다.

<표 1. Suspicious Predicate (SP)와 Ochiai의 결함 위치 식별 성능을 비교하였다.>

	MRR	acc@1	acc@3	max
SP	0.767	53.3%	100%	2
Ochiai	0.127	0%	20%	100

표 1에서 볼 수 있듯, 유형이 맞는 버그에 대해서 의심스러운 조건문을 이용한 결함 위치 식별은 SBFL에 비해서 상당히 정확했다. 의심스러운 조건문은 최대 2번의 localization effort 안에 정확한 결함 위치를 찾은 반면, Ochiai는 acc@3가 20%에 불과했으며, 최대 100번까지 검사해야 정확한 결함 위치를 찾을 수 있었다. 한편, 결함 수정 정확도 측면에서는, 지목한 결함 위치에서 추천한 결함 수정 방법이 항상 정확했다. 이를 통해 의심스러운 조건문이 유형에 맞는 버그에 대해서는 결함 위치 식별에 상당한 도움을 줄 뿐 아니라, 결함을 수정하는 방법도 정확히 제안할 수 있음을 확인하였다.

3.3 Defects4J 데이터셋에서 결함 유형 예측

본 실험에서는 의심스러운 조건문을 이용하여 유형에 맞는 9개의 Defects4J 결함에 대해 각각의 결함 유형을 예측하였다. 실험에 사용된 결함은 유형별로 비교 연산자 결함 2개, 코드 내 조건문을 지우는 결함이 3개, 잘못된 변수를 쓴 결함 4개였다. 각각의 결함에 대해서, 실제 결함이 있는 위치에서 의심스러운 조건문을 구하고, 이에 맞춰 결함 유형을 예측하였다.

표 2에 결과가 실려있다. 다른 언어를 쓰는 Siemens suite에서의 결과와 마찬가지로, 결함 위치가 주어졌을 때 의심스러운 조건문으로 오류 유형을 정확히 예상할 수 있음을 확인하였다. 이는 의심스러운 조건문을 사용한 자동 디버깅이, 특정 데이터셋에 국한되지 않고 일반적으로 사용할 수 있는 방법론임을 시사한다.

<표 2. 연구 대상으로 삼은 Defects4J 결함 (결함ID) 마다 실제 결함 유형 (GT), 의심스러운 조건문으로 예상한 결함 유형 (SP), 추출된 의심스러운 조건문.>

결함 ID	GT	SP	의심스러운 조건문
Chart11	3	3	p1 != p2
Chart24	3	3	v != value
Closure6	2	2	(predicate at location)
Closure62	1	1	charno == sourceExcerpt.length()
Closure115	2	2	(predicate at location)
Closure126	2	2	(predicate at location)
Math59	3	3	a != b
Math85	1	1	fa * fb == 0
Math98	3	3	nRows != nCols

Defects4J 결함 가운데 Math59는 의심스러운 조건문으로 수정 공간을 줄일 수 있는 좋은 예시이다. 그림 1에 나타난 Math59의 diff를 보면, 좋은 결함 위치 식별 알고리즘을 사용하여 결함 위치가 3482번째 줄임을 구해도 statement가 복잡하여 수정 공간이 쉽사리 좁혀지지 않는다.

```

src/main/java/org/apache/commons/math/util/FastMath.java [CHANGED]
@@ -3479,7 +3479,7 @@ public static long max(final long a, final long b)
3479 3479     * @return b if a is lesser or equal to b, a otherwise
3480 3480     */
3481 3481     public static float max(final float a, final float b) {
3482 -     return (a <= b) ? b : (Float.isNaN(a + b) ? Float.NaN : b);
3482 +     return (a <= b) ? b : (Float.isNaN(a + b) ? Float.NaN : a);
3483 3483     }
3484 3484
3485 3485     /** Compute the maximum of two values
  
```

<그림 1. Defects4J Math59의 diff.>

이 위치에서 의심스러운 조건문을 구하면 (a != b)로, a/b misuse (결함 유형 3번)이 의심된다는 점을 암시한다. Section 2.1에서 언급했듯, a/b의 사용 빈도 균형은 식의 마지막 부분에 가서 깨지는데, 실제로 개발자가 이 부분을 수정했음을 확인할 수 있다.

4. 한계 및 향후 연구

본 연구는 유형이 맞는 결함에 대해서 집중적으로 평가했다는 한계가 있다. 실제 디버깅 상황에서는 결함이 의심스러운 조건문으로 설명 가능한 결함인지 미리 알 수 없기 때문이다. 그러나 유형에 맞는 결함에 대해서는 최대 2번의 시도 끝에 찾았다는 점과, 앞서 언급했듯 자동 디버깅 이외에도 의심스러운 조건문은 개발자의 디버깅에 도움을 줄 수 있기 때문에, 추후 연구를 통해 의심스러운 조건문을 실제 상황에서 효과적으로 사용하는 방법을 탐구해야 할 것이다.

본 연구에서는 세 가지 결함 유형에 한해 실험을 진행했으나, 추후 연구에서는 보다 다양한 결함 유형에서 의심스러운 조건문을 이용한 연구를 진행할 계획이다.

5. 결론

본 연구에서는 ‘의심스러운 조건문’을 소개하여, 자동 디버깅에 의심스러운 조건문을 어떻게 쓸 수 있는지

제안하고, 두 데이터셋에 걸친 실험을 통해 의심스러운 조건문이 결함 위치 식별 및 결함 수정에 도움을 줄 수 있음을 보였다. 본 연구를 더 많은 결함 유형에 대해 일반화한다면, 자동 디버깅의 성능 및 이해에 기여하리라 기대한다.

Acknowledgement. 이 성과는 2021 년도 정부 (과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2020R1A2C1013629).

참고문헌

Sohn, J. a. Y. S., 2017. *FLUCCS: Using Code and Change Metrics to Improve Fault Localisation*. s.l., s.n., pp. 273--283.

Kim, D. a. N. J. a. S. J. a. K. S., 2013. *Automatic Patch Generation Learned from Human-written Patches*. s.l., s.n., pp. 802--811.

Xia, X. a. B. L. a. L. D. a. L. S., 2016. *Automated Debugging Considered Harmful" Considered Harmful: A User Study Revisiting the Usefulness of Spectra-Based Fault Localization Techniques with Professionals Using Real Bugs from Large Systems*. s.l., s.n.

Jeffrey, D. a. G. N. a. G. R., 2008. *Fault Localization Using Value Replacement*. s.l., s.n.

Mechtaev, S. a. Y. J. a. R. A., 2016. *Angelix: Scalable Multiline Program Patch Synthesis via Symbolic Analysis*. s.l., s.n.

경량 뉴럴 네트워크 탐색을 위한 가지치기, 지식증류, 그리고 양자화에 기반한 하이브리드 압축 방법

이재원⁽¹⁾, 이상민⁽²⁾

⁽¹⁾ 광운대학교 소프트웨어학부,

⁽²⁾ 광운대학교 정보융합학부

jcicle67@gmail.com, smlee5679@gmail.com

Hybrid Compression Method via Pruning, Knowledge Distillation, and Quantization for Identifying Lightweight Neural Networks

Jaewon Lee ⁽¹⁾, Sangmin Lee ⁽²⁾

⁽¹⁾ School of Software, College of Software and Convergence, Kwangwoon

⁽²⁾ School of Information Convergence, College of Software and Convergence, Kwangwoon

요 약

최근 딥러닝 모델은 범용적인 문제 해결과 정확도 성능 향상을 위해 점점 더 무겁고 깊어지고 있다. 하지만, 비대해진 딥러닝 모델은 저사양의 디바이스에서 효과적으로 동작하는데 적합하지 않은 문제상황이 자주 제기되고 있고, 특히 모바일 등의 제한된 리소스 환경에서 구동을 위한 모델 경량화 연구가 절실하다. 본 연구는 학습된 뉴럴 네트워크 모델을 효율적으로 압축함과 동시에 기존의 정확도 성능 하락을 최소화하는 하이브리드 압축 방법을 제안한다. 제안 기법은 예측 정확도, 손실의 최소화 및 모델 크기 감소, 그리고 추론 속도 향상 모두를 고려하였다. 제안 기법의 효과성 및 우수성을 입증하기 위해 ResNet-34 모델을 기반으로 실험을 진행하였으며, 실험 결과 최근 소개된 기법 대비 더욱 우수한 성능을 보였다.

1. 연구 배경

최근 딥러닝 알고리즘은 컴퓨터 비전, 자연어 처리, 음성 인식 등 다수의 인공지능 연구 분야에서 우수한 성능을 보이고 있다. 딥러닝의 가장 큰 성공 요소는 레이블 정확성이 보장된 수많은 데이터를 학습할 수 있고, 비정형성 데이터에서의 복잡한 패턴을 인식할 수 있도록 모델의 가용능력(capacity)을 무한히 확장할 수 있기 때문이다. 특히, 딥러닝을 구성하는 매개변수의 크기를 늘리고 신경망 층을 더 깊게 설정할수록 그 모델의 성능은 보편적으로 증가한다. 최근 등장하는 성능이 우수한 것으로 평가받는 대다수의 딥러닝 프레임워크들은 공통적으로 많은 수의 파라미터와 매우 큰 모델 크기를 가지게 된다. 이러한 대규모의 학습모델은 모델을 불러오고 추론(inference)을 수행하는 것조차 고성능 PC 또는 서버를 요구하는 경우가 많다. 물론, 수많은 코어를 내장한 CPU, GPU를 다수 사용가능한 상황에서는 큰 문제가 되지 않는다.

하지만, 모바일, IOT 디바이스와 같이 제한된 하드웨어 환경에서는 추론시간이 매우 증가하여 효율이 굉장히 떨어질 뿐만 아니라 다수의 사람들이 사용하거나 또는 모델 개선 및 발전할 수 있는 학습모델의 효용성 및 활용성이 매우 떨어진다. 또한, 실시간 추론 서비스가 필요한 모델을 만드는 경우 빠른 추론 처리가 필수적이지만 딥러닝 모델 크기와 깊이가 커질수록 매우 큰 계산비용이 발생하기 때문에 어려움이 나타나고 있다. 따라서 낮은 사양의 구동 환경에서도 만족스러운 예측 성능을 보장함과 동시에 효율적으로 동작할 수 있도록 만드는 모델 경량화가 절실히 필요하다.

2. 기존 연구의 한계점

최근 모델 경량화를 위해서, 대표적으로 가지치기(neural pruning), 양자화(quantization), 지식 증류(knowledge distillation)으로 3가지에 대한 연구가 다수 존재한다. 개별 기법들이 갖는 알고리즘의 특징 및 한계점은 다음과 같다. 첫째, 가지치기 방식은 영향을 미치지 않는 중요하지 않은 매개변수를 제거하여 연산량 감소 및 모델 크기 축소율이 좋은 방법이다. 하지만, 가지치기의 수준을 정하는 것은 휴리스틱 방식에 의존하는데 과도한 가지치기는 모델 정확도 성능을 높이는 핵심 weight를 손상시켜 재학습과정(retraining) 후에도 회복이 불가하게 만든다.

둘째, 양자화는 심층 신경망 모델에서 계산에 활용되는 매개변수의 타입을 부동 소수점(floating point) 정밀도 보다 낮은 대역폭(bandwidth)인 고정 소수점(fixed point)로 저장하여 모델의 크기를 감소시키고 추론속도를 개선하는 효과를 얻는데 효과적이다. 하지만 학습모델의 중요도가 높은 가중치 또한 고정 소수점 변환됨에 따라 정보 손실에 따른 성능 편차가 심하고 정확도 성능 하락이 불가피하다.

셋째, 지식 증류 기법은 기존 성능이 좋은 큰 모델(teacher model)로부터 성능을 만들어내는 필수적인 요소(knowledge)들을 작은 모델(student model)로 이전시켜 학습시키는 방법이다. 지식 증류 기법을 활용하여 다른 경량화 기법들과 결합으로 더 좋은 압축효과를 만들어낼 뿐만 아니라 다른 기법에서 공통적으로 나타나는 정보 손실에 따른 정확도 하락을 보완해줄 수 있다[1]. 그러므로, 효율적이고 효과적인

딥러닝 모델을 학습하기 위한 하이브리드 방식이 필요하다.

특히, 모델의 압축과정에서 재학습과정(retraining) 및 파인 튜닝(fine-tuning)과정이 필수적이므로, 지식 증류와 여타 압축 기술을 혼합한 하이브리드 압축 방법이 필요하다 [2]. 예를 들어, 양자화(quantization) 기법과 지식 증류 기법의 결합을 적용하거나 가지치기와 지식 증류기법 결합을 적용하여 기존 정확도의 손실을 최소화와 동시에 모델 크기 감소 및 추론 속도 향상을 획득할 수 있다 [3,4]. 이런 모델 경량화 결합 기법들에는 공통적으로 추론 시 중요하고 필수적인 정보(dark knowledge)만 추출하여 기존 모델의 정확도는 유지하며 더 작은 모델로 이전하는 것을 목표로 한다.

하지만 기존 하이브리드 모델 압축 방법에는 다음과 같은 문제점이 존재한다. 첫째, 모델 압축 최적화 측면에서 양자화-증류방법(quantized distillation)은 중요하지 않은 매개변수와 중요한 매개변수 모두를 고정 소수점으로 나타내게 되어 정보 손실이 발생하는 한계가 있다 [3]. 가지치기 기반의 지식 증류 방법(pruning with knowledge distillation)은 중요한 매개변수들이 대다수를 존재할 경우, 정확도 손실이 크거나 압축 효과가 떨어질 수 있는 한계가 있다.

이러한 기존 모델 압축 방법론의 문제점을 해결하고자, 본 연구에서는 미리 학습된 모델에서 필수적이고 중요한 정보만 이전하여 정확도 손실을 최소화하고 최대의 압축 효과와 추론 속도의 향상을 나타낼 수 있는 하이브리드 방식을 제안한다. 본 논문에서는 제안하는 기법의 상세 알고리즘과 함께 정확도, 모델 크기, 그리고 추론속도 측면에서 실험을 수행하여 검증하였다.

3. 제안 방법론

3.1 방법론 개요

그림 1은 본 연구의 전체 프로세스를 보여준다. 전체적인 프로세스는 크게 2가지로 나눌 수 있다. 첫째, 미리 학습된 모델(pre-trained)을 가지치기(pruning)과정과 지식 증류(knowledge distillation)과정을 반복적으로 진행한다. 목표로 하는 희소성(sparsity) 비율에 도달할 때까지 반복적으로 가지치기과정과 지식 증류 과정을 진행한다. 이 과정은 연산과정에서 복잡성을 증가시키는 중요하지 않는 가중치들을 가지치기 과정으로 제거하고, 이 가지치기 과정에서 손실된 부분을 지식 증류로 재학습과정(retraining)을 진행한다. 둘째, 1차 압축된 모델(pruned Model)로 양자화 인식 학습(quantization-aware training; QAT)을 진행한다. 이 과정은 불필요한 하게 많은 공간을 차지하고 있는 가중치를 축소하여 효율적인 양자화(quantized)된 정수형 모델을 만든다. 가지치기가 완료된 pruned model을 teacher model로 설정하여 teacher model에서 양자화된 정수형 모델로 변환하는 과정에서 양자화, 역양자화(dequantization)모듈을 삽입하여 정수형 양자화에 의해 생기는 반올림(round)의 효과를 테스트하며 학습이 진행된다[5]. 이 과정에서 발생하는 손실에 대해서 지식 증류로 재학습과정을 거치게 되면서 제안하는 압축 모델이 완성된다. 일반적으로 모델 압축 기법인 가지치기 과정과 양자화 과정을 거치게 되면 정보 손실에 따른 정확도 하락이 발생하기 때문에 파인-튜닝(fine-tuning)으로

재학습과정을 거치게 되지만, 파인-튜닝 보다 정확도 손실 회복이 효과적인 지식 증류(knowledge distillation) 기법을 통해 재학습을 진행하였다 [3][4].

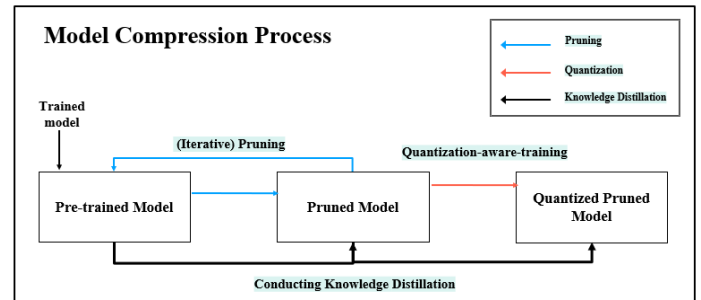


그림1. 하이브리드 모델 압축 방법 절차

3.2 비용 함수

본 연구에서 반복적으로 진행되는 지식 증류를 통한 재학습(retraining)과정은 그림2. 와 같다. 반응 기반 지식 증류(response-based knowledge distillation)를 활용하여 데이터를 통해 pre-trained된 모델과 압축모델에 같은 입력 값에 대한 출력 값의 차이와, 정답 값(ground truth)과 압축모델의 출력 값의 차이를 통해 weight를 업데이트 하기 위한 역전파(backpropagation)를 진행한다.

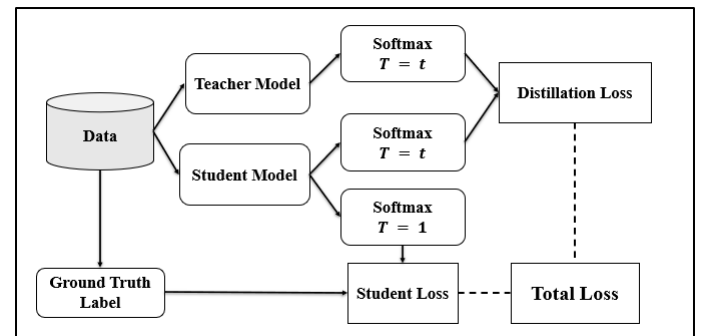


그림2. 지식 증류(knowledge distillation)를 활용한 전체 비용 함수

그림1. 의 pruning단계에서는 pre-trained model이 teacher모델로 설정이 되고 pruning이 진행된 모델은 student 모델로 설정된다. 두 모델의 결과값(logits)을 temperature T로 나누어 soft한 결과값을 만든 후 softmax과정을 거치고 난 뒤 발생한 손실에 대해서 distillation loss로 정의하였다 [1]. 따라서 지식 증류를 활용한 재학습과정의 전체 비용 함수는 수식(1)과 같다. 다음 수식(1)은 그림2.의 total loss를 나타낸다.

$$L_{total} = \alpha T^2 * KL(\sigma(\frac{Zs}{T}), \sigma(\frac{Zt}{T})) + (1 - \alpha)L_{ce}(\sigma(Zs), \hat{y}) \quad (1)$$

수식(1)에서 T와 S는 각각 teacher model과 student model이고, α 값은 hyperparameter를 나타낸다. Zt, Zs는 각각 teacher와 student의 logits값을 가리킨다. 두 logits값을 KL-divergence를 통해 S가 T를 모방할 수 있도록 만들어주고 교차 엔트로피 L_{ce}를 통해 실제 정답 레이블 ŷ와 S 예측오차의 편차인 σ(Zs)의 차이를 줄이도록 비용함수를 설정하였다.

3.3 제안하는 알고리즘의 의사코드(pseudocode)

Algorithm 1: Framework of Hybrid model compression method(Iterative Pruning, Quantization, and Knowledge Distillation)

```

Input:
    pre-trained model T
    initial student model S
    student model sparsity p: Sp
    target sparsity t: St
    the number of epochs of QAT process step n
Output:
    student model S*

#Conducting model compression
S ← T #Initialize
Loop (True):
    #Step1. Pruning(T,S)
     $\hat{\theta}^s = \underset{\theta}{\operatorname{argmin}} |P(\mathbb{D}|\theta^T) - P(\mathbb{D}|\theta^S)| + \alpha_1 \|\theta^S\|$ 
    #Step2. KnowledgeDistillation(T,S)
     $\hat{\theta}^s = \underset{\theta}{\operatorname{argmax}} |P(\mathbb{D}|\theta^T) + \alpha_2 \cdot P(\theta^P, \theta^S)|$ 
    # Termination condition
    if Sp ≥ St break

#Step3. Quantization aware training(S)
T ← S #Initialize
 $\hat{\theta}^s = \operatorname{round}(q_{\theta} \theta_f^s - \operatorname{round}(\min_{\theta_f^s} q_{\theta}))$ 

#Step4. KnowledgeDistillation(T,S)
for j = 1 to n do:
     $\hat{\theta}^s = \underset{\theta}{\operatorname{argmax}} |P(\mathbb{D}|\theta^T) + \alpha_2 \cdot P(\theta^P, \theta^S)|$ 
end for
return S
    
```

Algorithm 1의 pseudocode는 제안하는 기법의 전체 흐름을 보여주고 있다. 가용능력 제한 없이 구축한 학습모델(pretrained model) 인스턴스 T 와 압축 후 모델 인스턴스 S, 모델 희소성(sparsity) 비율 S_p, 목표 희소성 비율 S_t, 그리고 양자화 인식학습(QAT) 횟수 n을 입력으로 받는다. 먼저, 초기화 단계에서는 T 인스턴스를 S 인스턴스로 deep copy하고 모델 압축을 진행한다.

모델 압축 단계는 총 3단계로, 가지치기, 지식 증류, 그리고 양자화로 구성된다. 첫째, 가지치기 단계에서는 S 의 예측 성능을 유지하며 희소성을 높이고자 사전 정의한만큼 가지치기를 진행한다. 이 때 α₁ 을 설정하여 가지치기 수준을 조정할 수 있다. 둘째, 지식 증류 단계에서는 수식(1)을 통해 S

에 대한 예측 오류를 최소화함과 동시에 모델 T 처럼 S 가 예측하도록 유도한다. 이 때 α₂ 를 설정하여 T 와 S 의 유사도를 반영하는 비율을 조정 가능하다. 이 단계는 종료조건이 만족될 때까지 반복한다.

마지막으로, 양자화 단계에서는 가중치(weight)와 활성화(activation)을 8비트로 양자화하여 부동 소수점을 잘라내는 단계를 거친다. 이 때 최적의 s* 를 찾기 위해 반복적인 탐색 방식을 택하며 지식 증류를 통해 손실된 값을 다시 학습한다.

4. 실험

4.1 실험 설정

본 연구에서는 CIFAR-100데이터셋과 ResNet34을 기반으로 실험하였고, 해당 baseline model은 아래의 표1과 같다.

Table 1. CIFAR100 dataset에 대한 baseline model인 ResNet34의 Top-1 정확도, 크기, 추론 지연도

Model	Top-1 acc.(%)	Size(MB)	Inference latency(ms)
ResNet34	78.1	83.4	171.3(CPU)/14.9(GPU)

본 실험에서는 제안 기법을 우수성을 보이기 위한 그림1에서의 baseline model, pruned model (PKD), 그리고 quantized pruned model(proposed hybrid method) 세가지를 비교실험을 진행하였다. 그리고 quantization과정에서 정수형 양자화는 8비트(bits)로 설정하여 실험을 진행하였다. 첫째, 그림1의 pre-trained단계를 baseline으로 지정하고 각 스텝 별로 모델의 정확도와 크기, 그리고 추론 지연시간(inference latency)를 동일한 횟수만큼 반복 수행하며 실험결과를 비교하였다. 또한 추론 지연시간의 경우, 저 사양의 하드웨어 환경에서의 추론 시간이 중요하므로 표2와 같이 CPU에서의 직접 추론 지연시간을 측정하였다. 추가적으로 각 단계에서 모델 압축 수행 시간에 대해 측정하였다. 둘째, 모델 경량화에 큰 영향을 끼치는 목표 희소성 비율 S_t의 크기를 50%, 75%, 90%, 95%로 나누어 모델 정확도와 크기 축소에 끼치는 영향을 실험하였다. 표3은 세가지 비교 학습모델에 대한 결과를 보여주고 있다.

Table 2. 모델의 정확도, 크기, 추론 속도 비교 (sparsity: 90%)

Model name	Top-1 acc.(%)	Size(MB)	Inference latency(ms)	Operation time of model compression (min)
ResNet-34 (Baseline)	78.1	83.4	171.3	-
ResNet-34 (PKD)	76.6	14.1	28.9	28.4 ± 5.0
ResNet-34 (Hybrid)	79.2	9.3	6.1	57.0 ± 5.0

Table 3. Sparsity에 따른 모델 정확도, 크기 비교 (Baseline, Top-1 acc. : 78.1 (%), Size: 83.4(MB))

Sparsity	50%		75%		90%		95%	
	Top-1 acc.(%)	Size(MB)	Top-1 acc.(%)	Size(MB)	Top-1 acc.(%)	Size(MB)	Top-1 acc.(%)	Size(MB)
ResNet-34 (PKD)	78.7	44.9	79.0	27.2	76.6	14.1	74.2	11.5
ResNet-34 (Hybrid)	81.3	13.4	80.7	11.6	79.2	9.3	72.5	7.6

4.2 실험 결과

표2는 모델 압축에 대한 비교실험 결과를 보여주고 있다. 특히, 제안 기법인 hybrid 방식에 따라 압축할 경우 희소성 비율을 90%만큼 줄였음에도 정확도가 오히려 증가하는 경우를 목격할 수 있다. 이러한 실험 결과는 모델의 압축이 적절하게 적용될 경우, 오히려 학습효과를 향상시킬 수 있다는 부분을 시사하고 있다. 본 논문에서 제안하는 hybrid 압축 알고리즘을 적용한 결과, 정확도 증가 또는 보존하는 효과 뿐만 아니라, baseline 대비 추론시간이 단축되었음을 알 수 있다. 압축연산 수행을 위한 오버헤드는 PKD과정에서 약 28분, 그리고 quantization과정을 거쳐 완성된 압축과정까지 약 57분이 소요되었다.

표3은 희소성 비율이 증가할수록 모델 압축효과가 증가한다는 점을 확인할 수 있다. 하지만, 희소성 비율이 95%를 초과하게 되면 기존 모델 대비 성능 하락이 발생하기 시작한다.

5. 결론

본 논문에서는 딥러닝 모델을 제한된 리소스 환경에서 효율적으로 동작하지 못하는 문제를 해결하기 위해서 모델 경량화 기법들의 하이브리드 결합형태의 알고리즘을 제안하였다. 실험 결과, 압축효과를 내는 것과 동시에 정확도 향상이라는 성능 결과를 도출하였다. 특히, 추론 속도 측면에서 최신 알고리즘(state-of-the-art algorithm) 대비 우월한 성능 향상을 보였다. 하지만, 성능 테스트 과정에서 여럿의 데이터셋과 다양한 아키텍처를 고려하면서 보다 실험 결과에 대해서 검증할 필요성이 있다.

아직은 실험적으로 제안 방식의 성능을 검증하는 절차가 더욱 필요하나, 실험을 통해서 제안한 하이브리드 모델 압축 알고리즘에 대한 가능성과 방향성을 확인할 수 있었다. 향후 연구에서는 보다 다양한 데이터셋과 아키텍처를 통해 제안하고 압축연산 수행을 위해 요구되는 시간을 단축시킬 수 있는 후속 연구도 수행할 계획이다. 또한 정확도를 보장함과 동시에 더 높은 압축률을 확보하기 위해 알고리즘 개선을 꾀하고자 한다.

참고 문헌

[1] Distilling the knowledge in a neural network: G Hinton, O Vinyals, J Dean - arXiv preprint arXiv:1503.02531, 2015 - arxiv.org
 [2] Knowledge Distillation: A Survey J Gou, B Yu, SJ Maybank, D Tao - International Journal of Computer Vision, 2021

[3] Model compression via distillation and quantization: A Polino, R Pascanu, D Alistarh - arXiv preprint arXiv:1802.05668, 2018 - arxiv.org
 [4] Combining Weight Pruning and Knowledge Distillation for CNN Compression : Nima Aghli, Eraldo Ribeiro; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2021, pp. 3191-3198
 [5] Quantizing deep convolutional networks for efficient inference: A whitepaper: R Krishnamoorthi - arXiv preprint arXiv:1806.08342, 2018 - arxiv.org

FPGA 네트워크 기능 가속기를 지원하기 위한 빠르고 효율적인 소프트웨어 스택*

부준혁^o 김장우

서울대학교 전기정보공학부

junehyuk@snu.ac.kr, jangwoo@snu.ac.kr

Fast and Efficient Software Stack Support for FPGA Networking Function Accelerators

Junehyuk Boo^o Jangwoo Kim

Dept. of Electrical and Computer Engineering, Seoul National University

요 약

최근 워크로드(workload)들의 네트워크 성능 요구량이 늘어나게 되면서 100Gbps 수준의 높은 네트워크 대역폭이 일반적으로 사용되고 있다. 이러한 고성능 네트워킹(networking)을 지원할 때 필요한 많은 CPU 사용량을 절약하고 유지 어플리케이션들이 CPU를 더 많이 사용할 수 있도록 하기 위해 네트워킹 기능(function)을 가속하는 하드웨어 기술들이 주목받고 있다. 이 중 유망한 하나의 후보는 네트워크 카드(network interface card)에 FPGA를 설치하여 FPGA로 네트워킹 기능들을 가속하는 것이다. 하지만 이러한 하드웨어 가속기를 효율적으로 사용하기 위해서는 이를 뒷받침해주는 소프트웨어 스택 개발이 필요하다. 따라서 본 연구에서는 효율적으로 네트워킹 기능 가속을 하는 하드웨어를 사용할 수 있도록 이에 맞는 소프트웨어 스택이 만족해야 하는 조건들을 나열하고 이를 해결할 수 있도록 하는 한 가지 구현 방법을 제시한다.

1. 서 론

최근 워크로드(workload)들이 점점 더 복잡해짐에 따라 하나의 노드에서 워크로드를 처리하는 것이 아닌 여러 노드에 걸쳐 계산을 수행하는 scale-out 계산 방식이 많이 채택되고 있다. Scale-out 계산을 하기 위해서는 네트워크를 사용하여 여러 노드에서 분산적으로 처리되고 있는 정보들을 서로 주고받아 하나의 기능을 수행할 수 있도록 해주어야 한다. 또한, 네트워킹 기능들을 효율적으로 처리하고 보다 다양한 기능들을 지원하기 위해 복잡한 네트워킹 기능들이 추가되고 있다. 이에 따라서 네트워크 성능 요구량이 점점 더 증가하게 되었고, 이를 지원하기 위한 CPU 사용량이 증가하여 더 이상 CPU만으로는 네트워크 대역폭을 포화시킬 수 없게 되었다.

이러한 변화 속에서 네트워킹 기능들을 지원하기 위한 CPU 사용량을 덜어주기 위해서 네트워킹 스택의 일부분을 하드웨어로 offloading하고자 하는 연구들이 등장하였다. 이들은 네트워킹의 5개 layer (application, transport, network, link, physical) 중 대부분인 두 번째

단계인 link layer에서부터 다섯 번째 단계인 application layer까지 다양한 네트워킹 기능에 대한 가속을 시도하였다. 이들은 네트워킹 기능들을 가속하기 위해 네트워크 카드에 존재하는 ARM 코어와 같은 CPU를 사용하여 가속하려는 시도와 네트워크 카드에 ASIC을 추가하여 가속하는 시도 등이 존재한다. 그 중에서도 높은 처리 성능을 제공해주면서도 유연성(flexibility)을 제공하는 FPGA를 사용하여 네트워킹 기능들을 가속하는 연구들이 주목받고 있다 [1,2,3].

워크로드들이 이러한 하드웨어 가속기를 효율적으로 활용할 수 있도록 하기 위해서는 적절한 소프트웨어의 지원이 필수적이지만 아직까지는 가속기들을 위한 소프트웨어 스택 지원이 부족하다. 기존 연구들은 그보다는 하드웨어 아키텍처를 어떻게 설계하고 구현해야 하는지에 대해 초점을 맞추었다. 하지만 하드웨어 가속 방식을 채택하기 위해서는 충분한 하드웨어 가속기의 성능과 더불어 소프트웨어 스택이 기존의 소프트웨어 네트워킹 스택과 동일한 기능들을 지원할 수 있어야 한다. 이에 더해, 워크로드들이 요구하는 높은 네트워크 성능을 제공하기 위해 필요한 높은 CPU 사용량을 효과적으로 덜어주기 위해 적은 CPU cycle을 소모하면서 높은 성능을 달성할 수 있도록 해야 한다. 하지만 이러한 일은 크고 복잡한 기능들을 지원하여 많은 CPU 사용량을 요구하는 기존의

* 본 연구는 IDEC의 EDA Tool을 지원받아 수행하고, 서울대학교 자동화시스템연구소와 반도체공동연구소의 지원을 받음.

소프트웨어 스택을 통해서도 달성할 수 없다.

따라서 본 연구에서는 하드웨어 가속기들을 잘 사용할 수 있도록 하는 조건들에 대해 탐색한다. 이를 위해 하나의 예시로 많은 어플리케이션들이 사용하는 대표적인 네트워킹 기능인 Transmission Control Protocol(TCP)를 가속할 때 소프트웨어 스택이 갖춰야 하는 조건들에 대해 분석한다. 본 연구에서는 TCP를 가속할 때 다음의 두 가지 조건을 만족시켜야 한다는 결론을 도출하였다. 첫째, 하드웨어 가속이 유의미하도록 소프트웨어 스택은 CPU를 적게 사용하면서도 높은 성능을 달성할 수 있어야 한다. 둘째, 유저 어플리케이션이 쉽게 하드웨어 가속기를 사용할 수 있도록 어플리케이션에 큰 수정사항 없이 하드웨어 가속을 사용할 수 있도록 지원해야 한다.

더 나아가 이러한 조건들을 만족하도록 하는 소프트웨어 스택 아키텍처를 제안한다. 해당 아키텍처의 두 가지 특성이 이러한 조건을 만족할 수 있도록 해준다. 첫째, 높은 성능을 달성하면서 CPU를 많이 사용하지 않기 위해 유저 영역(userspace)에서 직접 하드웨어와 통신하는 구조를 채택한다. 이러한 방식은 커널을 거칠 필요가 없어 높은 성능을 유지할 수 있다. 둘째, 유저 어플리케이션을 최소한으로 수정하면서 하드웨어 가속을 지원하기 위해 어플리케이션들이 기존의 네트워킹 기능을 사용하기 위해 필요했던 API들을 최대한 유저 소프트웨어에서 지원한다. 이를 통해 유저 어플리케이션들은 원래의 기능을 그대로 유지하면서 하드웨어 가속의 장점들을 모두 누릴 수 있게 된다.

2. 배경 지식

2.1 네트워킹 스택

서로 다른 컴퓨터간 통신을 하기 위해서는 프로토콜로 정해져 있는 네트워킹 스택(networking stack)을 통해 통신을 해야 한다. 이러한 네트워킹 스택은 보내고자 하는 정보가 어느 방향으로 보내져야 하는지 (layer 2), 네트워크 구조에서 어떻게 전송이 될 것인지 (layer 3), 각각의 컴퓨터가 받은 데이터를 어떻게 처리할 것인지 (layer 4), 그리고 정보에 대한 어플리케이션의 전처리 정보를 담아 보내게 된다 (layer 5). 그리고 이러한 정보들과 보내고자 하는 실제 데이터를 합쳐 패킷(packet)이라 부르며, 네트워킹을 위해 네트워킹 스택이 계산을 통해 생성하는 추가 정보들을 패킷 헤더(header)라 한다.

이러한 네트워킹 기능들을 효과적이면서도 안정적으로 지원하기 위해 Linux 등의 운영체제에서는 커널(kernel) 영역에서 이러한 네트워킹 처리를 한다. 하지만 커널의 기능들을 사용하기 위해서 유저-커널 전환, cache pollution 등의 overhead가 발생한다 [4]. 또한, 커널은 주로적으로 코드를 실행하는 것이 아닌

유저 어플리케이션이 요청을 했을 때에만 코드를 실행하기 때문에 높은 성능을 얻도록 하기 위한 최적화를 넣기 힘들 수 있다.

2.2 FPGA

Field Programmable Gate Array(FPGA)는 LUT 등을 사용하여 ASIC과 같이 datapath 아키텍처를 채택함과 동시에 재프로그래밍이 가능하다. Datapath 아키텍처는 명령어 등을 읽고 인식하여 계산을 하는 것이 아닌 계산 자체가 회로에 내장되어 있어 회로가 계산의 결과물 자체를 출력하게 된다. 이러한 아키텍처는 pipelining과 같은 게이트 수준의 최적화를 할 수 있게 해주며 이를 통해 높은 성능을 달성할 수 있도록 해준다. 또한 FPGA는 칩을 만든 이후 LUT등을 재설정함으로써 FPGA 내에 존재하는 연산 유닛들을 다시 프로그래밍 할 수 있다. 이를 통해 구현하고자 하는 연산에 있어 높은 유연성을 제공한다.

Host 시스템은 일반적으로 FPGA와 PCI Express(PCIe)를 통해 통신하며 이를 위해 host에서는 PCIe device driver를 구현하여 어떻게 통신할 것인지에 대한 프로토콜을 명시한다. 이 때, host에서 FPGA로 정보를 보내기 위해서는 일반적으로 CPU가 주도적으로 FPGA의 메모리 영역에 직접 읽기 또는 쓰기를 수행하는 memory-mapped I/O(MMIO)와 FPGA가 주도적으로 host의 DRAM에 읽기를 수행하는 direct memory access(DMA)를 사용한다. 또한, FPGA가 host에게 정보를 보내기 위해서는 일반적으로 값을 host DRAM에 DMA를 통해 적어놓은 뒤 interrupt를 보내 host가 그것을 처리하도록 하거나, 정해진 영역에 값을 쓴 뒤 host가 주기적으로 해당 영역의 값을 읽어서 (polling) 데이터가 왔음을 확인하는 방법을 사용한다.

3. 소프트웨어 디자인 및 구조

3.1 구현 목표

본 연구에서는 FPGA를 사용하여 네트워킹 기능들을 가속할 때 달성해야 하는 목표들을 나열하고 이를 달성하는 방법을 제시한다.

첫째, CPU cycle을 많이 소모하는 네트워킹 기능들을 가속하여 높은 성능과 다양한 기능을 얻으면서도 CPU를 많이 사용하지 않아야 한다. 이러한 목표를 설정한 이유는 만약 하드웨어 가속을 사용하더라도 CPU를 많이 사용하면 하드웨어 가속을 하는 장점이 사라지는 것이고, 만약 CPU를 많이 사용하지 않더라도 소프트웨어를 사용하는 것보다 낮은 성능을 얻거나 제한된 기능만을 지원할 수 있다면 하드웨어 가속을 사용하는 것이 더 나은 선택이 아닌 강점과 약점이 공존하는 상태가 되는 것이다. 이러한 경우 사용자가 노력을 들여 하드웨어 가속이 더 나은 상황인지를

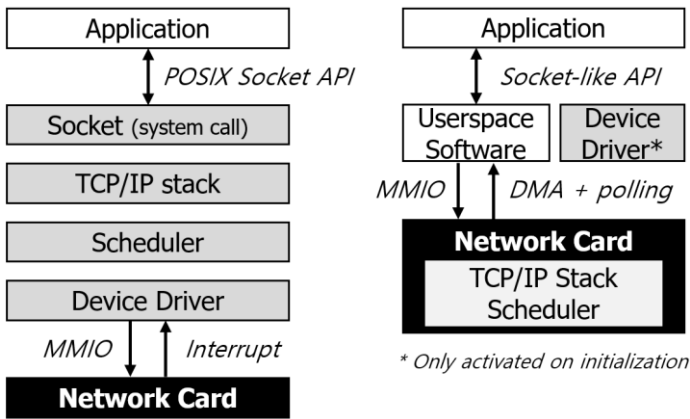


그림 1. 기존 네트워킹 스택(왼쪽)과 본 논문에서 제안하는 네트워킹 스택(오른쪽) 도식

판단한 뒤 특수한 상황에 대해서만 하드웨어 가속을 하도록 해주어야 하므로 적용성이 낮아지게 된다.

둘째, 기존 소프트웨어 어플리케이션을 큰 수정사항 없이 지원할 수 있어야 한다. 이러한 경우 소스 코드를 수정할 수 없는 어플리케이션들은 하드웨어 가속을 사용할 수 없게 되고, 소스 코드를 수정할 수 있다고 하더라도 해당 코드에 대한 깊은 이해가 있어야만 하드웨어 가속을 적용시킬 수 있게 된다. 이렇듯 하드웨어 가속을 하기 위해 어플리케이션들을 모두 수정해주어야 한다면 이는 사용자들에게 매우 큰 부담으로 다가오며 사용자들이 하드웨어 가속을 하지 않도록 할 것이다.

3.2 소프트웨어 스택 구조

앞서 언급한 목표를 달성하기 위한 소프트웨어 스택 구조를 제안한다. 그림 1의 왼쪽 그림이 기존 소프트웨어 중 Linux의 네트워킹 스택을 보여주고 오른쪽 그림이 본 논문에서 제안하고 있는 하드웨어 가속을 위한 소프트웨어 스택을 보여준다. 이 때, 흰색 상자는 유저 영역에 대한 소프트웨어를, 회색 상자는 커널 영역에 대한 소프트웨어를, 검은색 상자는 하드웨어를 나타낸다.

3.2.1 효율적인 유저 영역 소프트웨어 스택

첫째, CPU를 많이 사용하지 않으면서도 높은 성능을 달성하기 위해 유저 영역에서 효율적으로 하드웨어와 통신할 수 있도록 해야 한다. Linux 등의 운영체제에서 커널을 통해 네트워크 카드와 통신을 하는 것은 커널에서 안전하게 네트워킹 기능들을 지원하도록 하기 위함이다. 하지만 하드웨어 가속을 할 경우 이러한 네트워킹 기능들을 더 이상 소프트웨어에서 구현할 필요가 없다. 그러므로 유저-커널 전환 overhead와 커널을 사용함으로써 발생하는 overhead들을 없애기 위해 유저 영역에서 직접 하드웨어와 통신을 하도록 해서 높은 성능을 유지하면서 CPU의 사용량을 줄일 수

있다.

한편, 유저 영역에서 FPGA와 통신을 하더라도 운영체제에서 FPGA를 인식하기 위해서는 device driver가 필요하다. 이를 위해 device driver에서 가상 character device를 만들어 유저 소프트웨어가 하드웨어와 통신할 수 있도록 해준다. 또한, 디바이스 드라이버는 해당 device에 대해 open(), mmap() 등의 함수에 대해 FPGA와 통신하기 위해 MMIO를 위한 PCIe BAR region과 DMA를 위한 DMA buffer를 생성하고 유저 소프트웨어가 해당 영역에 접근할 수 있도록 리턴 값들을 정의해주게 된다. 이렇게 만들어진 가상 character device를 유저 소프트웨어가 mmap() 함수를 통해 자신의 virtual memory address space에 매핑함으로써 포인터를 통한 일반적인 메모리 접근을 통해 MMIO를 발생시키고 DMA 영역에 있는 DMA된 값들을 읽을 수 있게 된다.

또한, 유저 영역에서는 FPGA가 보내는 interrupt를 쉽게 처리할 수 없다. 이를 위해 유저 영역에서는 DPDK[5]와 비슷하게 polling 방식을 채택할 수 있다. 비록 polling을 하기 위해서는 하나의 CPU 코어를 독점적으로 사용해야 하지만, 하나의 코어를 사용하여 다른 모든 코어들이 커널 영역 접근 없이 효율적으로 동작할 수 있게 된다. 구체적으로는 FPGA가 interrupt를 발생시키는 대신 사전에 정의된 영역에 값을 적고 유저 소프트웨어가 해당 영역을 지속적으로 읽고 있을 경우 값이 변할 경우 하드웨어가 어떠한 동작을 했음을 확인할 수 있다. 이를 통해 하드웨어가 어떠한 동작을 수행했는지를 다른 커맨드 영역에 적어 놓게 된다면 소프트웨어는 polling을 통해 동작이 수행되었음을 알고 커맨드 영역에서 커맨드를 읽어 구체적으로 어떤 동작이 수행되었는지를 확인할 수 있게 된다.

3.2.2 어플리케이션 수정을 최소화하기 위한 API 제공

둘째, 유저 어플리케이션을 크게 변경하지 않고 CPU를 많이 사용하지 않는 가벼운 유저 소프트웨어 스택을 활용하기 위해 기존 API와 거의 동일한 API를 제공하고 동일한 기능들을 지원할 수 있어야 한다. Linux에서 TCP를 사용할 경우 유저 어플리케이션은 listen(), connect(), accept()와 같은 POSIX socket API를 사용하여 TCP 스택을 사용한다. 이러한 API들과 최대한 비슷한 API를 제공해야 기존 어플리케이션들을 최소한으로 수정하면서 효율적으로 하드웨어 가속을 지원할 수 있다.

POSIX socket API와 비슷한 API를 제공하기 위해 유저 라이브러리(library)를 구현하여 비슷한 API들을 제공해준다. 예를 들어, connect() 함수의 경우 시스템 콜이 아닌 유저 라이브러리에 user_connect() 함수를 새로 정의하고 해당 함수는 socket API와 동일한 형식의 argument를 받도록 한다 (connect 함수의 경우 연결하고자 하는 상대의 IP와 port번호, 그리고 자신이

사용할 port 번호를 argument로 받는다). 사용자가 어플리케이션 코드를 수정하여 connect() 시스템 콜을 user_connect() 유저 함수로 대체해주기만 한다면 해당 기능을 커널이 수행하지 않고 유저 소프트웨어가 받아 하드웨어가 처리할 수 있도록 한다. 이를 통해 유저 소프트웨어가 커널을 거치지 않고 TCP 기능들을 하드웨어에서 처리될 수 있도록 해준다.

유저 소프트웨어에서 TCP와 같은 네트워킹 기능을 직접적으로 처리하지 않는다고 하더라도 기능들을 보고하고, 또 효율적으로 통신하기 위해 어느 정도의 처리를 해주어야 한다. TCP의 경우 하나의 socket이 하나의 TCP 연결과 1대1로 대응이 되며, FPGA로 요청을 보낼 때 몇 번째 TCP 연결인지를 같이 보내주게 되면 FPGA에서는 이를 다시 한 번 더 파싱(parsing)할 필요 없이 처리를 할 수 있게 된다. 또한 TCP에서 보내거나 받기 위한 버퍼의 크기가 정해져 있는데 (예를 들어, 64KB) 현재 버퍼의 크기가 얼마나 남아있는지를 기록해두면 어플리케이션에서 데이터 전송 요청을 보냈을 시 버퍼가 부족할 경우 바로 버퍼가 부족하여 당장 보낼 수 없다는 신호(flag)를 리턴해줄 수 있게 된다. 이러한 작은 양의 메타데이터를 유지함으로써 소프트웨어에서 너무 많은 overhead를 갖지 않으면서도 효율적으로 통신을 지원해줄 수 있다.

4. 구현 및 결과

섹션 3에서 언급한 아키텍처를 Linux kernel v5.3에서 구현하였다. Host의 경우 Intel사의 Xeon Gold 5118 CPU를 사용하였으며 FPGA의 경우 Xilinx사의 U280 FPGA를 사용하였다. FPGA에서는 BAR 영역을 4KB만 할당하여 MMIO를 받을 수 있도록 해주었고, FPGA device driver를 구현하여 dma_alloc_coherent() 함수를 통해 DMA buffer를 할당해주었다. MMIO 영역과 DMA 영역 모두 앞선 섹션에서 언급한 대로 virtual character device를 통해 유저 소프트웨어 스택이 mmap() 함수를 통해 접근할 수 있도록 해주었다. 특히, DMA 영역에는 TCP 통신을 위한 buffer과 함께 송신 및 수신 명령 큐들을 할당하여 MMIO 또는 FPGA가 정해진 영역에 write를 한 뒤 유저 소프트웨어가 그것을 polling을 통해 확인한 뒤 명령 큐에 값을 읽어 구체적으로 어떤 것들이 수행되었는지 확인할 수 있도록 하였다.

이렇게 구현한 소프트웨어 스택과 Linux의 TCP/IP 소프트웨어 스택을 사용하여 동일한 write() socket API 함수를 128B 크기로 호출한 결과 유저 소프트웨어는 같은 성능을 달성하기 위해 유저 소프트웨어 구현이 Linux의 구현 대비 12% 수준으로 CPU를 사용함을 확인할 수 있었다. 이 때, 유저 소프트웨어는 polling에 사용되는 CPU 사용량까지 모두 포함한 수치이다. 128B 크기의 요청을 사용한 이유는 작은 크기의 요청을

사용할 때 같은 대역폭을 달성하기 위해 함수 호출을 더 많이 해야 하므로 소프트웨어 스택에 높은 부하를 주기 때문이다. 이러한 실험을 통해 본 연구에서 제시된 소프트웨어 스택이 Linux의 소프트웨어 스택보다 CPU를 약 8배 적게 사용함을 확인할 수 있었다. 반대로 CPU 사용량(코어 개수)을 동일하게 설정한 뒤 write() 함수를 호출할 경우 약 8배 높은 성능을 달성함을 확인할 수 있었다.

5. 결론 및 향후 계획

네트워킹 성능 요구량이 많아지고 기능들이 복잡해짐에 따라 요구되는 CPU 사용량이 많아져 FPGA 등의 하드웨어로 네트워킹 기능들을 offloading하는 연구들이 제안되고 있으나 이들을 지원하기 위한 소프트웨어 스택 지원이 부족하다. 본 연구에서는 이러한 하드웨어 가속기들을 효율적으로 활용하기 위해 두 가지 조건, (1) CPU를 많이 사용하지 않도록 효율적으로 소프트웨어 스택을 구성하는 것과 (2) 유저 어플리케이션의 수정을 최소화하도록 해야 한다는 것을 밝힌다. 또한, 이 두 가지 조건을 만족하는 소프트웨어 스택 구현 방법을 제시한다.

미래에는 더 나아가 최소한의 기능을 지원해야 하는 소프트웨어 스택이 가지고 있는 최적화 가능성에 대해서 더 구체적으로 분석해보고 어떻게 하면 overhead들을 최소화할 수 있는지에 대해 조사를 할 계획이다. 또한, 커널 영역을 거치지 않고 유저 영역을 통해서만 통신을 하게 되면 커널이 더 이상 이러한 통신 등을 볼 수 없기 때문에 네트워크 모니터링 등의 관점에서 잃어버리고 있는 기능들이 존재할 수 있는데, 이러한 기능들에 대해 더 조사할 계획이다.

참고 문헌

- [1] Brunella et al., "hXDP: Efficient Software Packet Processing on FPGA NICs," USENIX Conference on Operating Systems Design and Implementation, 2020.
- [2] Lin et al., "PANIC: A High-Performance Programmable NIC for Multi-tenant Networks," USENIX Conference on Operating Systems Design and Implementation, 2020.
- [3] Firestone et al., "Azure Accelerated Networking: SmartNICs in the Public Cloud", USENIX Symposium on Networked Systems Design and Implementation, 2018.
- [4] Kaufmann et al., "TAS: TCP Acceleration as an OS Service," European Conference on Computer Systems, 2019.
- [5] "Data Plane Development Kit (DPDK)." <https://www.dpdk.org/>

딥 러닝을 활용한 메소드와 주석 간의 적합성 검증 모델

박재현[○] 이동건[○] 장용훈[○] 서영석

영남대학교 컴퓨터공학과

kat3160@yu.ac.kr, dklee77@yu.ac.kr, killerwise@ynu.ac.kr, ysseo@yu.ac.kr

Verifying Conformance Model between Methods and related Comments based on Deep Learning

Jaehyun Park[○] Dong-Gun Lee[○] Yonghun Jang[○] Yeong-Seok Seo

Dept. of Computer Engineering, Yeungnam University

요 약

소프트웨어 공학에서 주석 (Comment)은 높은 중요도를 차지한다. 특히, 소프트웨어 개발 단계 중 유지보수 단계에서 주석의 중요성이 부각된다. 그럼에도 불구하고 현업에서는 시간적 한계로 인해 코드에 대한 주석이 해당 코드에 대해 불충분하며 잘못된 설명을 전달하는 현상이 빈번하게 발생한다. 이러한 현상이 누적되면 코드의 수정과 이해가 어려운 레거시 코드 (Legacy code)가 되며, 이는 소프트웨어의 유지보수를 어렵게 만든다. 따라서 본 논문에서는 이러한 문제점을 해결하기 위해 자연어 처리에서 높은 성능을 보이는 Long Short-Term Memory (LSTM)와 워드투벡터 (Word2vec) 기반의 코드-주석 쌍 적합성 판별 모델을 제안한다. 제안된 모델을 통해 개발자들은 작성한 주석이 코드의 메소드에 대해 충분하며 정확한 설명을 전달하는지를 검증하는 결정 지표로 활용할 수 있게 되며, 해당 지표를 기반으로 주석을 개선할 수 있게 된다. 제안된 모델의 검증을 위해 Java 프로그래밍 언어에서 기본으로 제공하는 패키지에서 코드-주석 쌍을 수집한 뒤 실험을 수행하였다. 실험 결과 본 논문에서 제안된 모델이 유의미한 성능을 보여주는 것을 평가 지표로 확인할 수 있었다.

1. 서론

소프트웨어 유지보수란 이미 개발된 소프트웨어에 대해 결함을 수정하거나 성능과 특성을 개선하는 등의 작업을 의미한다. 소프트웨어 유지보수는 실행 환경 및 버전, 메모리 등 다양한 측면에서 이루어지고 있다. 이러한 유지보수는 전체 소프트웨어 개발 주기에서 상당히 높은 비중을 차지하기 때문에 소프트웨어 사용 기간 동안 지속적으로 이루어져야 한다. 소프트웨어 유지보수를 고려하며 개발할 경우 단기적인 작은 수고를 통해 장기적으로 큰 수고를 덜 수 있으며 프로그램 확장성과 성능관리에도 큰 이점을 얻을 수 있다. 따라서 소프트웨어 유지보수의 중요성은 점점 더 부각되어가고 있으며 이러한 추세에 맞추어 기업들은 소프트웨어 유지보수 단계에 많은 투자를 하고 있다. 궁극적으로 유지보수는 기업의 이윤 최대화뿐만 아니라 소프트웨어 사용자의 만족도 강화 측면에서 매우 중요한 단계이다 [1-2].

이러한 소프트웨어의 유지보수가 원활하게 이루어지기 위해서는 소프트웨어 이해도를 높일 수 있어야 하는데 이를 위해서는 완성된 코드를 잘 설명할 수 있는 문서들과 주석들이 필수적이다 [3]. 그러나 실제 개발 환경에서는 시간적 한계로 인해 코드에 대한 주석이 불충분하며 잘못된 설명을 전달하고 있는

실정이다. 이러한 현상이 누적되면 기존 코드는 수정과 이해가 어려운 레거시 코드가 된다. 레거시 코드는 개발자들 간 코드 공유 및 의사소통에 큰 걸림돌이 되어 시간적 비용을 많이 낭비하게 할 수 있으며 개발자 본인이 개발한 프로그램임에도 불구하고 프로그램 분석을 어렵게 만들기도 한다. 궁극적으로 레거시 코드는 소프트웨어의 유지보수를 어렵게 만든다 [4].

따라서 본 논문에서는 원활한 유지보수를 위해 메소드에 매칭된 각 주석이 적절하게 작성되어 있는지 판별하기 위한 딥 러닝 기반의 주석 적합성 검증 모델을 제안하고자 한다. 제안된 모델은 전처리를 위한 Word2vec 기반의 워드 임베딩 (Word Embedding) 기술과 자연어 처리에 강세를 보이는 LSTM 기반 학습 모델을 사용한다 [5-6]. 또한 성능 평가를 위해 객체 지향 언어 중 높은 이식성과 다양한 오픈소스 라이브러리를 제공하는 자바 (Java)를 학습 및 검증 코드로 설정하였고 java.util, java.math 등 자바에서 기본으로 제공하는 36가지 패키지에 대해 메소드 코드-주석 쌍을 수집하였다 [7].

이렇게 제안된 모델을 통해 개발자들은 개발 중인 소프트웨어의 각 메소드에 대한 주석의 적합성 정보를

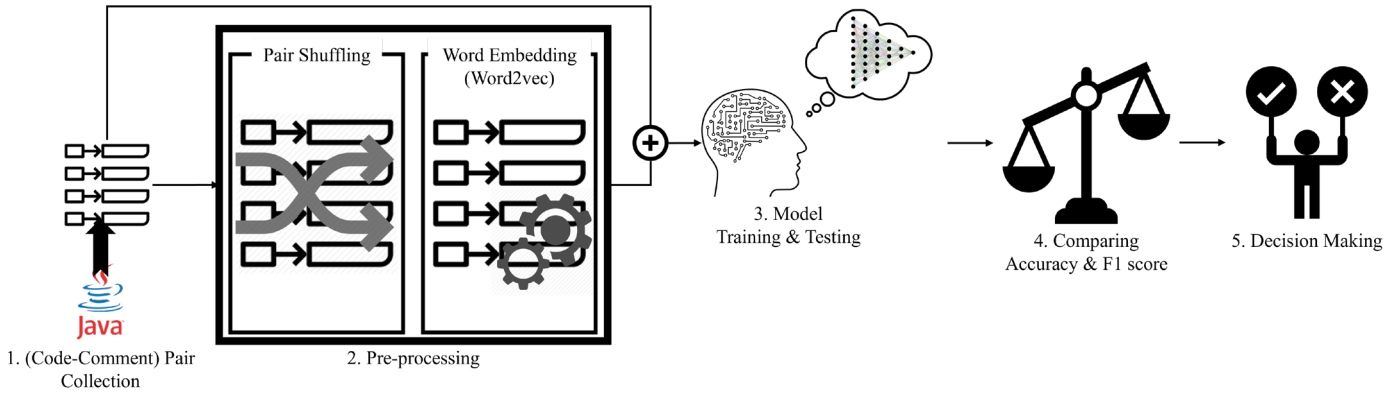


그림 1 Overall approach of the proposed method

제공받음으로써 원활한 유지보수를 위한 결정 지표로 활용할 수 있다.

따라서 개발자들은 주석에 대한 적합성을 검증하는 결정 지표로 활용할 수 있으며, 유지보수를 보다 더 효율적으로 진행할 수 있을 것이다.

이하, 1장 서론에 이어 본 논문의 구성은 다음과 같다. 2장에서는 원활한 유지보수를 위한 주석과 관련된 연구를 기술한다. 3장에서는 본 논문에 대한 전반적인 접근들을 설명한다. 다음 4장에서는 실험 설계 및 실험 결과에 대해 서술한다. 마지막으로 5장에서는 본 논문의 결과를 요약하며 결론 및 향후 연구를 기술한다.

2. 관련 연구

본 장에서는 원활한 유지보수를 위한 주석 관련 선행 연구들에 대해 소개한다. 주석 위치 추천, 주석 생성 등 수많은 연구들이 제안되었으며, 이 연구들은 소프트웨어 유지보수에 큰 영향을 끼쳤다 [8-9].

2.1. Comment position prediction

Y. Huang은 코드를 입력으로 받아 주석 위치를 추천하는 LSTM 기반의 주석 위치 예측 모델을 제안했다. 제안된 모델은 68.4%의 F1 score를 보였으며, 기존 연구 대비 11.4% 높은 성능을 보였다 [8].

2.2. Comment generator

X. Hu는 코드를 입력으로 받아 주석을 생성하는 Gated Recurrent Unit (GRU) 기반의 주석 생성 모델을 제안했다. 제안된 모델은 5가지 기존 연구 대비 7가지 평가 지표에서 높은 성능을 보였다 [9].

관련 연구에서 볼 수 있듯이 원활한 유지보수를 위한 주석 연구는 다양한 관점에서 이루어지고 있다. 그러나 작성된 주석에 대한 적합성을 판단하는 연구는 부족하다. 따라서 본 논문에서는 작성된 주석이 코드의 메소드에 대해 충분하며 정확한 설명을 전달하고 있는지 적합성 측면에 초점을 둔다.

3. Overall Approach

그림 1은 본 논문의 전반적인 접근들이다. 딥 러닝

학습을 위해 자바의 코드-주석 쌍을 수집한다. 수집된 데이터에 대해 2가지 전처리 작업인 페어 셔플링 (Pair Shuffling)과 워드 임베딩 (Word Embedding)을 거친다. 전처리 작업을 거친 후 나온 데이터와 원본 데이터를 이용해 딥 러닝 모델을 학습 및 검증한다. 검증 단계에서 Accuracy와 F1 score를 평가 지표로 사용하며 이 평가 지표를 통해 코드-주석 쌍에 대한 적합성을 'Good'과 'Bad' 2가지 레이블 (Label)로 표현한다.

3.1. Pre-processing

본 장에서는 수집된 데이터에 대한 2가지 전처리 작업인 페어 셔플링과 워드 임베딩에 대해 서술한다. 이 작업들은 모델 성능 향상에 기여한다.

3.1.1 Pair Shuffling

수집된 데이터에 대해 의도적으로 적절하지 않은 코드-주석 쌍을 만들기 위해 수집된 코드-주석 쌍 중 50%를 무작위로 추출한 뒤 서로 간의 주석을 섞었다. 이 작업은 추출된 모든 코드-주석 쌍이 원본 코드-주석 쌍과 서로 다른 결과를 가질 때까지 반복된다. 주석이 섞인 50%의 쌍은 코드에 대한 주석이 불충분하며 잘못된 설명을 전달하고 있다는 의미로 'Bad' 레이블이 된다. 나머지 50%의 쌍은 코드에 대한 주석이 충분하며 정확한 설명을 전달하고 있다는 의미로 'Good' 레이블이 된다.

3.1.2. Word Embedding

자연어를 기계가 이해할 수 있도록 데이터를 변환하는 방식 중 원-핫 인코딩 (One-hot Encoding)은 저장 공간 측면에서 비효율적이며, 단어 간의 연관성과 유사도를 표현할 수 없다는 한계가 있다 [10]. 따라서 이러한 한계를 해결하기 위해 단어 간의 연관성과 유사도를 고려해 단어와 실수를 1:1 매칭하는 다양한 워드 임베딩 방식이 제안되었으며, 본 논문에서는 대표적인 워드 임베딩 방식인 Word2vec을 사용한다 [6]. 코드는 주석과 달리 자연어 형태가 아니다. 그러나 Word2vec은 자연어를 위한 워드 임베딩 방식이다. 따라서 코드에는 Word2vec을 적용하지 않았다. 반면에 수집된 주석에 대해 Word2vec을 이용해 단어 간의

연관성과 유사도를 측정하였으며, Word2vec의 결과를 모델 학습 및 검증과 함께 사용한다.

4. 실험

4.1. 실험 설계

4.1.1 데이터 수집

본 논문은 실험을 위해 자바에서 기본으로 제공하는 패키지에서 메소드에 대한 코드-주석 쌍을 수집한다. 수집된 패키지는 표 1과 같으며 약 3,700개의 코드-주석 쌍을 수집하였다. 코드는 특별한 전처리 과정 없이 원본 코드를 학습 및 검증에 사용한다. 주석의 경우 모델 성능 향상을 위해 주석 내용 중 버전 정보와 같이 메소드 역할에 대해 설명하지 않는 내용은 제거하였다.

표 1 Packages used in our experiment

Parent Package	Sub Package
io	-
lang	annotation, constant, invoke, module ref, reflect, runtime
math	-
net	spi
nio	channels, charset, file
security	cert, interfaces, spec
text	spi
time	chrono, format, temporal, zone
util	concurrent, function, jar, random regex, spi, stream, zip

수집된 전체 데이터 중 train set, validation set, test set은 일반적으로 사용되는 비율인 6:2:2로 나누었다 [11-13]. 또한, 클래스 분포 불균형을 해결하기 위해 각 dataset은 동일한 ‘Good’ 레이블과 ‘Bad’ 레이블 수를 가진다.

4.1.2 모델 구성

본 논문은 Tensorflow 2.0 기반의 딥 러닝 모델을 사용하였으며, 자연어 처리에서 높은 성능을 보이는 LSTM을 사용하였다. 코드와 코드에 대한 주석을 입력 변수로 사용하며 ‘Good’ 레이블과 ‘Bad’ 레이블 중 하나가 output으로 출력된다. 하이퍼 파라미터의 경우 시간적 비용이 적게 소모되면서 최적의 성능을 보이는 값으로 설정하였으며, 표 2에서 상세한 값을 확인할 수 있다 [14].

4.1.3 평가 지표

본 장에서는 실험에 사용된 평가 지표에 대해 서술한다. 실험에 사용된 모델은 분류 모델이기 때문에 이에 적합한 2가지 평가 기준인 accuracy와 F1 score를 사용한다 [15-16].

4.1.3.1 Accuracy

Accuracy는 정확하게 예측된 샘플의 수를 전체 샘플

표 2 Hyperparameter used in the proposed model

Hyperparameter	Value
Unit	50
Activation function	relu
Optimizer	Adam
Loss function	binary_crossentropy
Learning rate	0.001
Batch_size	64
Epoch	300

의 수로 나눈 값이며, 전체적인 예측 정확도를 나타낸다. 0에서 1 사이의 값을 가지며 그 값이 클수록 더 높은 예측 성능을 의미한다 [15].

4.1.3.2 F1 score

Accuracy는 클래스 분포 불균형이 존재할 때 클래스가 많은 쪽으로 예측해 accuracy를 높일 수 있다는 한계가 존재한다. F1 score는 클래스 분포 불균형이 존재할 때 유용하게 사용될 수 있다. Accuracy와 동일하게 0에서 1 사이의 값을 가지며 그 값이 클수록 더 높은 예측 성능을 의미한다 [16].

4.2. 실험 결과

Word2vec 대신 원-핫 인코딩을 사용한 ‘without Word2vec’과 Word2vec을 사용한 ‘with Word2vec’을 서로 비교하였다. Test set을 활용하여 accuracy와 F1 score 측면에서 실험한 결과를 정리하면 표 3과 같다.

표 3 Accuracy and F1 score of the proposed model

	Accuracy	F1 score
without Word2vec	64.53%	60.97%
with Word2vec	77.51%	75.23%

Accuracy 측면에서는 Word2vec 기반의 제안된 모델이 Word2vec를 사용하지 않은 모델 대비 약 13% 향상된 77.51%라는 결과를 보였다. 이는 제안된 모델이 test set에서 정답을 맞힌 비율이 77.51%가 됨을 의미한다. 또한, F1 score 측면에서는 Word2vec 기반의 제안된 모델이 Word2vec를 사용하지 않은 모델 대비 약 14% 향상된 75.23%라는 결과를 보였다. 이는 제안된 모델이 클래스 분포 불균형을 반영했을 때 test set에서 정답을 맞힌 비율이 75.23%가 됨을 의미한다.

본 실험을 통해 내린 결론은 다음과 같다. Word2vec이 실제로 모델의 성능 향상에 기여하였다. Word2vec 기반의 제안된 모델이 Word2vec를 사용하지 않은 모델보다 더 향상된 성능을 보임을 두 평가 지표를 통해 알 수 있다. 또한, F1 score를 통해 클래스 분포 불균형에서도 큰 문제가 없음을 확인하였다.

본 논문을 통해 현업 개발자들은 코드-주석 쌍에 대한 적합성을 기존보다 더 정확하게 판별할 수 있으며, 궁극적으로 소프트웨어의 원활한 유지보수에 기여하게

된다.

5. 결론 및 향후 연구

본 논문에서는 원활한 유지보수를 위해 이미 작성된 주석이 코드에 대해 충분하며 정확한 설명을 하는지 적합성을 판별하는 Word2vec 기반 딥 러닝 모델을 제안했다. 모델 검증을 위해 Java에서 기본으로 제공하는 패키지에서 메소드에 대한 코드-주석 쌍을 수집하였으며, 객관적인 실험과 검증을 위해 accuracy와 F1 score를 평가 지표로 사용했다.

Word2vec 기반의 제안된 모델은 Word2vec를 사용하지 않은 모델보다 더 향상된 성능을 보임을 실험에 사용된 평가 지표인 accuracy와 F1 score를 통해 확인할 수 있었다. accuracy 측면에서는 약 13% 향상된 77.51%라는 결과를 보였으며, F1 score 측면에서는 약 14% 향상된 75.23%라는 결과를 보였다. 두 평가 지표를 종합해보았을 때 제안된 모델이 유의미한 성능을 보이며, 클래스 분포 불균형에서도 큰 문제가 없음을 결론 내렸다.

본 논문에서 제안된 모델을 통해 현업 개발자들은 코드-주석 쌍에 대한 적합성을 판별할 수 있으며, 원활한 유지보수 측면에서 결정 지표로 활용할 수 있을 것으로 기대된다.

본 논문을 기반으로 우리는 실제 오픈소스 프로젝트에서 좋은 주석과 나쁜 주석에 대한 특징을 추출할 예정이다. 추출된 주석에 대한 특징을 기반으로 딥 러닝 모델을 학습시킨 주석 판별 모델을 향후 연구로 발전시켜 나갈 계획이다.

Acknowledgement

이 성과는 2020년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2020R111A3073313) (corresponding author: Yeong-Seok Seo)

Reference

[1] KH. Bennett and VT. Rajlich, "Software maintenance and evolution: a roadmap" in Proceedings of the Conference on the Future of Software Engineering, pp.73-87, 2000.

[2] N. Chapin and JE. Hale, "Types of software evolution and software maintenance" in Journal of software maintenance and evolution: Research and Practice, Vol.13, No. 1, pp.3-30, 2001.

[3] B. Yang and Z. Liping, "A survey on research of code comment" in Proceedings of the 2019 3rd International Conference on Management Engineering, Software Engineering and Service Sciences, pp.45-51, 2019.

[4] BW. Weide and WD. Heym, "Reverse engineering of legacy code exposed" in Proceedings of the 17th

international conference on Software engineering, pp.327-331, 1995.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory" in Neural computation, Vol.9, No.8, pp.1735-1780, 1997.

[6] KW Church, "Word2vec" in Natural Language Engineering, Vol.23, No.1, pp.155-162, 2017.

[7] GS. Itzstein and M. Jasiunas, "On implementing high level concurrency in Java" in Asia-Pacific Conference on Advances in Computer Systems Architecture, pp.151-165, 2003.

[8] Y. Huang and X. Hu, "CommtPst: Deep learning source code for commenting positions prediction" in Journal of Systems and Software, Vol.170, pp.1-14, 2020.

[9] X. Hu and G. Li, "Deep code comment generation" in 2018 IEEE/ACM 26th International Conference on Program Comprehension, pp.1-11, 2018.

[10] P. Rodriguez and MA. Bautista, "Beyond one-hot encoding: Lower dimensional target embedding" in Image and Vision Computing, Vol.75, pp.21-32, 2018.

[11] Y. Zhang and Y. Xie, "The adoption of deep neural network (DNN) to the prediction of soil liquefaction based on shear wave velocity" in Bulletin of Engineering Geology and the Environment, pp.1-8, 2021.

[12] Q. Wei and Y. Zhao, "A new deep-learning framework for group emotion recognition" in Proceedings of the 19th ACM International Conference on Multimodal Interaction, pp.587-592, 2017.

[13] Z. Xu and T. Shen, "Apple Defect Detection Method Based on Convolutional Neural Network" in International Conference on Multimedia Technology and Enhanced Learning, pp.398-404, 2021.

[14] F. Hutter and H. Hoos, "An efficient approach for assessing hyperparameter importance" in international conference on machine learning, pp.754-762, 2014.

[15] W. Zhu and N. Zeng, "Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations" in NESUG proceedings: health care and life sciences, Vol.19, pp.1-67, 2010.

[16] M. Sokolova and N. Japkowicz, "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation" in AI 2006: Advances in Artificial Intelligence, .45, No.1, pp.12-19, 1994.

그래프 합성곱 신경망을 활용한 취약점 탐지 벤치마크

최유라^o 권영우

경북대학교 컴퓨터학부

yura0812@knu.ac.kr, ywkwon@knu.ac.kr

A Benchmark on Vulnerability Detection with Graph Convolutional Networks

Yoola Choi^o Young-Woo Kwon

School of Computer Science and Engineering,

Kyungpook National University

요 약

소프트웨어 내의 취약점을 탐지하기 위한 다양한 기법이 등장한 가운데, 그래프 신경망을 기반으로 하는 취약점 탐지 기법 역시 좋은 성능을 보여주었다. 딥 러닝 기반 취약점 탐지 연구에서는 사용하는 모델이 성능에 영향을 주기 때문에 모델 선정이 매우 중요한 과제이다. 하지만 그래프 신경망을 활용한 취약점 탐지 연구는 비교적 시작된 지 오래되지 않아 참고할 만한 연구가 적어 적합한 모델을 선정하기 쉽지 않다. 본 논문에서는 포인터 관련 6개의 취약점을 선정하고, 그래프 합성곱 신경망을 활용한 취약점 탐지 벤치마크를 수행한다. 결과에 따르면 GCN이 평균 84.39의 F1 점수로 가장 좋은 성능을 보였다. 이는 취약점 탐지 연구에 적합한 그래프 합성곱 신경망 모델을 선정하는 데 도움이 될 것으로 기대한다.

1. 서 론

소프트웨어가 점점 복잡해지고, 취약점의 수가 점차 증가함에 따라 소프트웨어에 내재된 취약점의 탐지 역시 어려워지고 있다. 기존에는 취약점 탐지를 위해 주로 패턴 기반의 정적 분석 도구를 활용하였으나, 이는 사람이 수동적으로 각각의 취약점에 대한 패턴을 정의해야 한다는 단점이 있다. 따라서 최근에는 딥 러닝을 활용한 취약점 탐지 연구가 활발히 진행되고 있다. 딥 러닝을 활용할 경우, 모델이 데이터에서 자동으로 패턴을 학습할 수 있으므로 기존의 취약점 탐지 기법이 가지고 있는 단점인 수동적인 패턴화 작업의 비중을 줄이는 것이 가능하다. 대부분의 딥 러닝 모델은 이미지와 텍스트 같이 구조화된 데이터를 처리하는데 좋은 성능을 보여왔다. 하지만 이와 같은 딥 러닝 모델을 취약점 탐지에 사용할 경우, 프로그램의 문맥과 같은 중요한 정보가 손실되는 상황이 발생할 가능성이 있다.

소스 코드 내의 다양한 문맥 정보는 여러 가지 형태의 그래프(추상 구문 트리, 제어 흐름 그래프, 프로그램 종속성 그래프 등)로 표현되기 때문이다. 따라서 텍스트 정보와 그래프 정보를 모두 고려하기 위해, 즉 그래프의 형태로 된 데이터를 직접적으로 처리하기 위해 취약점 탐지에 그래프 신경망을 활용하는 연구는 지속적으로 증가하고 있는 추세이다. 딥 러닝 기반의 취약점 탐지는 모델에 따라 취약점의 탐지율이 달라지기 때문에 모델을 선정하는 것은 중요한 과제로 여겨진다. 하지만 그래프 신경망 혹은 텍스트를 입력으로 하는 딥 러닝 기반의 취약점 탐지 벤치마크는 쉽게 찾아볼 수 있었던 반면,

그래프 신경망을 기반으로 하는 취약점 탐지에 관한 벤치마크 연구는 비교적 수가 적기 때문에 적합한 모델을 선정하기는 쉽지 않다.

그래프 신경망 기반의 취약점 탐지 연구를 진행하기 위해서는 우선 소스 코드를 그래프로 변환해야 하고, 이에 대해 추가적인 전처리 과정을 진행해야 한다. 전처리 과정은 총 세 가지로 구성되어 있는데, 그래프의 노드를 구성하는 소스 코드 간의 차이를 최소화하기 위한 심볼화, 임베딩 계산에 필요한 특징 벡터를 구성하기 위한 벡터화, 취약점의 내포 여부를 나타내기 위해 각 노드에 0 또는 1의 라벨을 추가하는 라벨링이 이에 해당한다. 전처리가 끝난 그래프는 그래프 신경망 모델의 입력으로 사용되며, 학습된 모델은 주어진 노드가 취약한지 아닌지 이진 분류를 진행한다. 이때 0으로 분류했을 경우 취약점의 미포함을 나타내고, 1로 분류했을 경우 취약점을 포함을 나타낸다.

본 논문에서는 그래프 신경망 중에서도 그래프 합성곱 신경망을 활용한 취약점 탐지에 대한 벤치마크를 수행한다. 기존의 정적 분석 도구, Clang Static Analyzer를 사용했을 때 취약점을 전혀 탐지하지 못한 경우가 포인터 관련 취약점에 집중되어 있었음에 주목하여 포인터와 관련된 6개의 취약점을 선정했다. 대표적인 취약점 데이터셋 Juliet Test Suite[1]로부터 해당 취약점과 관련된 소스 코드를 수집한 후, Joern[2]을 사용하여 프로그램 종속성 그래프를 추출했다. 그리고 추출된 그래프에 전처리 과정을 거쳐 그래프 신경망 모델의 입력으로 사용 및 취약점 포함 여부를 예측했다. 모델의 성능은 네 가

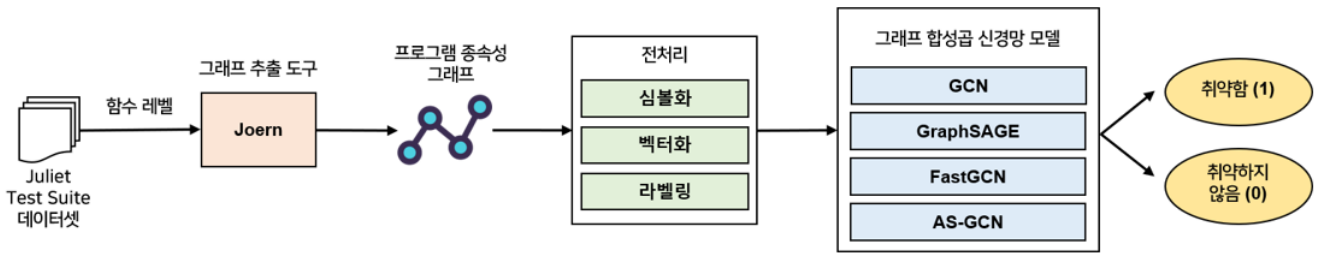


그림 1 벤치마크 개요

지의 성능 평가 지표에 따라 평가되었다. 각 모델의 성능을 바탕으로 벤치마크를 수행한 결과, Pooling Aggregator를 사용한 GraphSAGE[3]가 3개의 취약점, GCN[4]은 2개의 취약점에서 좋은 성능을 보였다. 그리고 AS-GCN[5]은 1개의 취약점에서 높은 탐지율을 기록했다. 하지만 FastGCN[6]의 경우에는 비교적 좋은 성능을 나타내지 못했다. 이처럼 취약점마다 가장 좋은 성능을 보인 모델은 달랐지만, GCN이 평균 84.39의 F1 점수로 전반적으로 가장 좋은 성능을 나타냈다. 본 논문에서 수행한 벤치마크는 향후 그래프 합성곱 신경망을 활용한 취약점 탐지 연구를 진행할 때, 모델 선정에 도움이 될 것으로 예상된다.

2. 배경지식

딥 러닝 기반 취약점 탐지 연구는 주로 소스 코드 혹은 벡터화된 소스 코드를 입력으로 해왔다. 하지만 대부분의 프로그램 중간 표현은 그래프의 형태를 띄고 있으며, 소스 코드의 텍스트 정보만 고려하게 될 경우 문맥 정보가 제대로 파악되지 않을 수도 있다는 한계가 있다. 따라서 문맥 정보의 손실을 최소화하고, 중간 표현을 활용하기 위해 취약점 탐지에 그래프 신경망을 활용하는 연구가 점차 증가하는 추세이다. 정적 분석 도구인 Clang Static Analyzer를 활용해 Juliet Test Suite[1]에 대해 취약점 탐지를 진행했을 때, CWE-468과 CWE-469에 대해서는 전혀 탐지하지 못했는데 이는 모두 포인터와 관련된 취약점이었다. 본 논문에서는 기존 포인터 분석에서 제어 흐름 혹은 문맥을 고려하기 때문에, 포인터 관련 취약점 탐지에는 문맥 정보를 포함할 수 있는 그래프 신경망을 활용하는 것이 더 효과적일 수 있다고 가정해 데이터셋을 구성했다.

프로그램 종속성 그래프 (Program Dependence Graph)는 프로그램 중간 표현 중 하나로, 제어 흐름과 데이터 흐름을 모두 포함하는 그래프이다. 각 노드는 프로그램 내의 문장들로 구성되어 있으며, 간선은 노드 간의 종속성을 나타낸다. 이전 연구[7]에서 프로그램의 전반적인 흐름을 포함하는 프로그램 종속성 그래프가 가장 좋은 성능을 보여주었기 때문에, 프로그램 종속성 그래프를 그래프 합성곱 신경망의 입력으로 선정했다.

그래프 합성곱 신경망 (Graph Convolutional Neural Network)은 그래프 데이터를 처리하기 위해 고안된 인공 신경망 중 하나로, 이미지에 주로 사용하던 합성곱을 그래프 데이터로 확장했다. 이웃 노드의 가중치를 곱하여 각 노드를 업데이트하며, 학습이 끝난 후에는 업데이트된 노드의 값을 바탕

으로 예측을 진행한다. 2017년 GCN[4]이 제안되고, 여러 과제에서 좋은 성능을 보여줌에 따라 다양한 버전의 그래프 합성곱 신경망 모델이 등장했다. 본 논문에서는 GCN을 비롯하여 GraphSAGE[3], FastGCN[6], AS-GCN[5]의 4가지 모델을 벤치마크 대상으로 선정했다.

3. 벤치마크

벤치마크는 다음과 같이 진행된다. 우선 포인터 관련 취약점 6개를 선정 및 이에 대한 그래프를 추출한다. 추출된 그래프는 일련의 전처리 과정을 거쳐 그래프 합성곱 신경망 모델의 입력으로 사용되고, 학습된 모델은 취약점 여부를 이진 분류로 탐지한다.

3.1. 데이터셋

본 논문에서는 포인터에 관련된 6개의 취약점을 선정해 벤치마크를 진행했으며, 데이터셋으로는 Juliet Test Suite v1.3 for C/C++[1]를 활용했다. 또한 그래프 신경망 모델의 입력으로 사용하기 위해서 Joern[2]을 통해 데이터셋의 소스 코드부터 프로그램 종속성 그래프를 추출했다. 실험에 사용한 취약점의 목록과 추출된 그래프의 개요는 표 1과 같다.

3.2. 전처리

데이터의 전처리 과정은 심볼화, 벡터화, 라벨링 3가지로 구성된다. 심볼화는 변수명, 함수명 등으로 인해 발생하는 소스 코드 간의 차이를 줄여 데이터의 일관성을 확보할 수 있으므로 모델의 성능을 높이는 데 효과적이다. 심볼화에는 VulDeePecker[8]에서 제공하는 소스 코드를 사용했다. 또한 그래프 신경망 모델에서는 노드의 임베딩 계산에 각 노드의 특징 벡터를 활용한다. 이때 특징 벡터를 구성하는 과정이 백

표 1 벤치마크에 사용한 취약점의 개요

취약점 ID	설명	그래프 개수	노드 개수	간선 개수
CWE-467	Use of sizeof() on a Pointer Type	147	2376	4911
CWE-468	Incorrect Pointer Scaling	98	1146	2592
CWE-469	Use of Pointer Subtraction to Determine Size	98	1230	3497
CWE-476	NULL Pointer Dereference	1097	8840	16474
CWE-587	Assignment of a Fixed Address to a Pointer	49	369	627
CWE-588	Attempt to Access Child of a Non-structure Pointer	107	887	1593

표 2 벤치마크 결과

	CWE-467				CWE-468				CWE-469			
	ACC	P	R	F1	ACC	P	R	F1	ACC	P	R	F1
GCN	80.00	88.01	82.24	85.03	77.34	97.90	75.85	85.48	80.15	95.45	78.75	86.30
GraphSAGE (Mean)	93.52	95.93	94.43	95.17	89.34	85.04	82.44	83.72	88.39	92.73	89.04	90.85
GraphSAGE (Pooling)	94.58	95.93	95.93	95.93	90.10	81.10	87.29	84.08	89.89	94.81	89.54	92.10
FastGCN	67.13	97.71	67.26	79.67	70.05	98.07	69.82	81.57	67.36	90.39	69.16	78.36
AS-GCN	64.95	61.96	96.50	75.47	66.41	72.52	87.73	79.40	66.81	69.60	92.88	79.57

	CWE-476				CWE-587				CWE-588			
	ACC	P	R	F1	ACC	P	R	F1	ACC	P	R	F1
GCN	77.83	99.51	77.89	87.38	73.90	86.44	76.50	81.17	73.55	90.89	72.98	80.96
GraphSAGE (Mean)	90.93	77.38	83.11	80.14	62.59	100.0	62.59	76.99	65.35	78.91	51.27	62.15
GraphSAGE (Pooling)	90.39	81.52	78.63	80.05	63.31	98.85	63.24	77.13	82.25	75.78	75.19	75.49
FastGCN	75.75	99.67	75.94	86.20	69.12	91.67	70.51	79.71	67.20	87.97	67.94	76.67
AS-GCN	75.67	78.34	94.93	85.90	62.87	63.27	98.19	76.95	61.27	81.89	89.41	85.48

터화로, 본 논문에서는 Doc2Vec[9] 모델을 사용해 각 노드를 구성하는 소스 코드를 벡터로 변환했다. 위의 두 과정이 끝나면 각 노드에 대해 취약점 여부를 표시하는 0 또는 1의 라벨을 붙인다. 벤치마크에 사용된 데이터셋은 취약점이 존재하는 함수의 이름에는 bad, 존재하지 않는 경우에는 good을 포함해 명명했기 때문에 함수명에 따라 취약점 유무를 라벨링했다. 각 노드의 라벨은 모델의 학습이 완료된 후 예측을 진행할 때 이진 분류에 활용된다.

3.3. 그래프 합성곱 신경망 모델

최초의 그래프 합성곱 신경망 모델 GCN으로부터 여러 그래프 합성곱 신경망 모델이 제안되었다. 본 논문에서는 GCN[4], GraphSAGE[3], FastGCN[6], AS-GCN[5] 4개의 그래프 합성곱 신경망 모델을 사용한다. GraphSAGE의 경우, Pooling과 Mean의 두 가지 Aggregator를 사용했다. 사용한 모델은 모두 GCN을 기반으로 하지만 각 노드의 이웃 노드를 샘플링 하는 방법에 차이가 있다. GCN의 경우 노드의 임베딩을 계산하기 위해 해당 노드의 이웃 노드를 모두 사용하기 때문에 대규모 그래프를 계산할 경우 메모리의 병목 현상을 초래한다는 단점이 있다. 따라서 이와 같은 단점을 보완하기 위해 이후에 등장한 모델은 샘플링을 통해 이웃 노드의 범위를 한정시키고자 하였다. GraphSAGE의 경우 해당 노드에서 k 거리만큼의 이웃 노드만 샘플링하며, FastGCN은 이웃 노드가 아니라 해당 노드에 연결된 간선을 각 레이어마다 독립적으로 샘플링한다. 마지막으로 AS-GCN은 레이어 별로 샘플링을 적용하고, 각 레이어를 연결한다.

각 모델의 파라미터는 에포크를 제외하고는 CWE에 따라 다르게 조절했다. 데이터셋은 훈련 65%, 검증 15%, 테스트 20%의 비율을 적용했으며, 실행마다 랜덤으로 분할한다. 에포크는 500 (GCN, GraphSAGE) 또는 1000 (FastGCN, AS-GCN)으로 설정했다. 학습 비율은 0.01 또는 0.005를 사용했으며 드롭아웃은 항상 0.6으로 고정했다. L2 정규화는 $\lambda=0.0005$ 또는 $\lambda=0.0002$ 로 설정했다. 벤치마크의 결과로는 총

10번 실행한 것 중에 가장 좋은 성능을 사용했다.

4. 벤치마크 결과 및 분석

각 모델은 정확도(ACC), 정밀도(P), 재현율(R), F1 점수(F1) 총 4개의 성능 평가 지표를 통해 평가된다. 정확도는 전체 데이터 중 예측에 성공한 케이스의 비율을 의미한다. 정밀도는 참이라고 예측한 케이스 중 실제 참인 케이스의 비율을 나타내며, 재현율은 실제 참인 케이스 중에서 모델이 참이라고 예측한 케이스이다. F1 점수는 정밀도와 재현율의 조화평균을 계산한 것으로, 정밀도와 재현율 간의 균형을 나타낸다.

포인터와 관련된 6개의 취약점에 대한 각 모델의 취약점 탐지 성능은 표 2와 같다. 본 논문에서 두 개 이상의 지표에서 가장 높은 값을 보인 모델을 해당 취약점 탐지에 좋은 성능을 보이는 모델로 선정하였으며, 만약 지표가 여러 모델에 분산되어 있을 경우 F1 점수가 가장 높은 모델을 선정했다. 취약점마다 가장 좋은 성능을 보인 모델은 다르게 나타났다. GCN은 총 3개의 취약점(CWE-468, CWE-476, CWE-587) 탐지에서 가장 좋은 성능을 보였다. 또한 Pooling Aggregator를 사용한 GraphSAGE은 총 2개의 취약점 (CWE-467, CWE-469), AS-GCN은 1개의 취약점(CWE-588)에서 가장 좋은 성능을 보였다. 반면에 FastGCN은 6개의 취약점 중 어느 취약점에서도 좋은 성능을 보여주지 못했다. 전반적으로 가장 성능이 좋은 모델은 전체 취약점의 평균 F1 점수가 높은 모델로 선정했다. 그 결과 평균 84.39로 GCN이 선정되었으며, Pooling Aggregator를 사용한 GraphSAGE 역시 84.13의 점수로 GCN과 근소한 차이를 보여주었다.

일부 취약점의 벤치마크 결과에서는 정밀도와 재현율 사이에는 큰 차이가 있었다. 이때 높은 정밀도와 낮은 재현율을 보였을 경우는 가짜 음성으로 검출된 케이스는 많으나 가짜 양성으로 검출된 케이스가 적은 것을 의미하므로 비교적 양성을 검출하기에 적합하다고 볼 수 있다. 반대로 낮은 정밀도와 높은 재현율을 보였을 경우는 음성을 검출하는데 비교적 적합하다고 해석할 수 있다. 그러므로 취약점 탐지를 진행할 때 낮은 오

탐을 중요시한다면 정밀도가 높은 모델을 선택하는 것이 좋고, 낮은 미탐율을 중요시한다면 재현율이 높은 모델을 선택하는 것이 효과적이다. 또한 F1 점수가 높은 모델의 경우 전반적으로 낮은 오경보를 필요로 하는 경우에 적합하다. 취약점을 탐지하는 목적에 따라 사용하기 적합한 모델의 추정 역시 가능하다. 만약 미탐율이 적은 모델을 선택하고자 한다면, 비교적 높은 재현율을 보인 AS-GCN 혹은 Pooling Aggregator를 사용한 GraphSAGE가 적절하다.

5. 관련 연구

G. Lin et al.[10]의 딥 러닝 기반의 취약점 탐지 벤치마크에서는 6개 모델을 비교하고, 정밀도와 재현율 두 가지의 지표로 모델을 평가했다. Z. Li, et al.[11]의 딥 러닝 기반의 취약점 탐지 벤치마크 또한 6개 모델을 사용했으며, 성능은 오탐율, 미탐율, 정확도, 정밀도, F1 점수의 5가지로 평가했다. 해당 논문에서는 본 논문과 마찬가지로 데이터셋으로부터 프로그램 종속성 그래프를 추출하지만, 그래프는 코드 가젯을 생성하는 데 활용될 뿐 신경망의 입력으로는 활용되지 않았다. 위의 두 연구는 Juliet Test Suite 데이터셋을 활용하고, 같은 성능 평가 지표를 사용한다는 측면에서 유사하지만, 텍스트 혹은 벡터를 입력으로 하고 그래프 신경망을 포함하지 않는다는 점에서 본 논문의 벤치마크와는 차이가 있다. 그래프 신경망 모델들을 7개의 데이터셋에 대해 비교 및 평가한 벤치마크 [12]도 있었으나, 해당 논문에 사용된 데이터셋은 취약점 탐지와 거리가 멀고 FastGCN과 AS-GCN을 포함하지 않는다는 점에서 본 논문과 차이가 있다. 따라서 취약점 탐지에 적합한 그래프 신경망 모델을 찾기 위해 참고하기에는 적절하지 않다.

이와 같이 그래프 신경망 혹은 딥 러닝 기반의 취약점 탐지 기법에 대한 벤치마크는 이미 존재하지만 두 내용을 함께 다룬 연구를 찾아보기 어렵다. 취약점 탐지에 그래프 신경망을 사용하는 경우가 증가하고 있으나, 실험에 참고하기 위한 벤치마크는 충분하지 않으므로 본 논문에서는 그래프 신경망, 구체적으로 그래프 합성곱 신경망을 활용한 취약점 탐지 벤치마크를 제공하고자 한다.

6. 결론

딥 러닝 기반의 취약점 탐지 연구는 기존 방법의 단점으로 지적됐던 수동적인 취약점 패턴화 작업을 보완할 수 있다는 측면에서 많은 관심을 받아왔다. 이전의 딥 러닝 기반 연구들이 주로 텍스트 혹은 벡터를 입력으로 받는 CNN 혹은 RNN 기반의 모델을 사용한 반면, 최신의 취약점 탐지 연구에서는 주로 프로그램의 다양한 중간 표현(제어 흐름 그래프, 프로그램 종속성 그래프 등)이 입력으로 사용되고 있다. 따라서 본 논문에서는 그래프를 활용한 딥 러닝 기반의 취약점 연구를 진행하고자 하는 사람들에게 가이드라인으로서 벤치마크를 제공한다. 벤치마크에서는 총 4개의 그래프 합성곱 신경망 모델을 대상으로 Juliet Test Suite에서 선정한 포인터 관련 6개의 취약점의 탐지를 진행했다. 해당 취약점을 내포하는 소스 코드에서 프로그램 종속성 그래프를 추출한 후, 심볼화, 벡터화, 라벨링의 전처리 과정을 거쳐 그래프 합성곱 신경망의 입력으

로 사용했다. 사용한 모델은 GCN, GraphSAGE, FastGCN 그리고 AS-GCN이며, 모델은 4가지의 성능 평가 지표를 사용해 평가되었다. 벤치마크 결과에 따르면 취약점 별로 가장 좋은 성능을 보이는 모델은 다르게 나타났다. 하지만 전반적으로 GCN이 평균 84.39의 F1 점수로 가장 좋은 성능을 보여주었다. 본 논문은 그래프 신경망 기반 취약점 탐지에 적합한 모델을 선정하는 데 도움을 줄 수 있을 것으로 기대한다.

Acknowledgement

이 논문은 2021년도 정부(교육부)의 재원으로 한국연구재단기초연구사업의 지원을 받아 수행된 연구임(NRF-2021R111A3043889)

7. 참고 문헌

- [1] P. E. Black, P. E. Black, "Juliet 1.3 test suite: Changes from 1.2.", US Department of Commerce, National Institute of Standards and Technology, 2018.
- [2] Joern, <https://joern.io/>, 2021. Accessed: 2021-12-17
- [3] W. L. Hamilton, R. Ying, J. Leskovec, "Inductive representation learning on large graphs.", International Conference on Neural Information Processing Systems, 2017.
- [4] T. N. Kipf, M. Welling, "Semi-supervised classification with graph convolutional networks.", arXiv preprint arXiv:1609.02907, 2016.
- [5] W. Huang, T. Zhang, Y. Rong, J. Huang, "Adaptive sampling towards fast graph representation learning.", arXiv preprint arXiv:1809.05343, 2018.
- [6] J. Chen, T. Ma, C. Xiao, "Fastgcn: fast learning with graph convolutional networks via importance sampling.", arXiv preprint arXiv:1801.10247, 2018.
- [7] Y. Choi, Y. Kwon, "Comparison of Program Representations on Vulnerability Detection with Graph Neural Networks.", IET Transactions on Smart Processing and Computing, Vol. 10, No. 6, pp. 477-482, 2021.
- [8] Z. Li et al, "Vuldeepecker: A deep learning-based system for vulnerability detection.", arXiv preprint arXiv:1801.01681, 2018.
- [9] Q. Le, T. Mikolov, "Distributed representations of sentences and documents.", International conference on machine learning, 2014.
- [10] G. Lin, W. Xiao, J. Zhang, Y. Xiang, "Deep learning-based vulnerable function detection: A benchmark.", Information and Communications Security, pp. 219-232, 2020.
- [11] Z. Li, et al., "A comparative study of deep learning-based vulnerability detection system.", IEEE Access, Vol. 7, pp. 103184-103197, 2019.
- [12] Z. Wu, et al., "A comprehensive survey on graph neural networks.", IEEE transactions on neural networks and learning systems, Vol. 32, Issue. 1, pp. 4-24, 2020.

자카드 지수를 이용한 한국어 요구사항 유사도 분석 자동화

장우성^o, 문소영, 김영철

홍익대학교 소프트웨어공학연구소

{uriel2001, whit2, bob}@hongik.ac.kr

Automatic Analysis Method using Jaccard Coefficient for Korean Requirements Similarity

Woo Sung Jang^o, So Young Moon, R. Young Chul Kim

SELab., Hongik University

요 약

요구공학의 영역에서 중요한 초점 중 하나는 고객의 비정형 요구사항을 올바르게 파악하는 것이다. 하지만 올바른 요구사항의 식별은 매우 어렵다. 그리고 한국어 요구사항 자동 분석 연구는 매우 드문 실정이다. 우리는 한국어 요구사항 기반의 원인/결과 모델링 및 테스트케이스 자동 생성에 초점을 두고 있다. 이 방법은 요구사항 문장을 단순화가 가장 중요하다. 하지만 요구사항의 원인/결과들의 유사성 식별을 해결하지 못한다. 문제의 해결을 위해 자카드 유사도를 이용한 요구사항 유사성 식별 방법을 제안한다. 이 방법은 유사도를 수학적으로 자동 계산함으로써 유사한 요구사항 내의 원인/결과를 일치화한다.

1. 서 론

많은 소프트웨어 개발 조직은 고품질의 소프트웨어를 개발하기 위해 다양한 테스트 방법을 사용하고, 테스트 비용을 최대한 절감하기 위한 방법을 고려한다[1]. 하지만 한국어 기반의 요구사항으로부터 테스트 케이스를 자동 생성하는 연구는 매우 드물다. 이는 한국어 문장 의미의 자동 분석이 힘들기 때문이다.

기존 어려움의 해결을 위해 한국어 요구사항으로부터 테스트케이스 자동 생성 방법을 연구한다[2]. 방법은 다음과 같다. 1) 한국어 요구사항을 단순화, 2) 단순화 과정을 C3Tree Model[3]로 표현, 3) C3Tree Model로부터 Cause-Effect Graph[4]를 생성, 4) Cause-Effect Graph로부터 Decision Table을 생성, 5) Decision Table로부터 테스트케이스를 생성하는 과정들을 자동화한다. 이 방법은 최소한의 테스트케이스로 100% 커버리지를 만족한다[2]. 하지만 유사한 요구사항 문장을 식별하지 못한다. 이는 중복되는 Cause와 Effect가 증가되고, 이는 테스트케이스의 불필요한 증가를 초래할 수 있다.

이러한 문제의 해결을 위해 자카드 유사도[5]를 사용하여 C3Tree Model에서 유사 노드들의 일치화 후, Cause-Effect Graph로 자동 변환하는 방법을 제안한다. 결과적으로 중복되는 테스트케이스를 최소화할 수 있다.

본 논문의 구성은 다음과 같다. 2장은 관련 연구를 언급한다. 3장은 자카드 유사도 분석을 이용한 한국어 요구사항 자동 분석 방법을 언급한다. 4장은 적용 사례를 언급한다. 마지막으로 결론 및 향후 연구를

언급한다.

2. 관련 연구

2.1 Cause-Effect Graph

Cause-Effect Graph[4]는 시스템에서 가장 작은 기능적 단위를 식별하여 원인(Cause)과 결과(Effect)의 관계를 가지는 기능들을 서로 연결한다. 이 관계에서 왼쪽의 노드가 원인이고, 오른쪽의 노드가 결과이다. Cause-Effect Graph의 연결 관계는 그림 1과 같다. Identity는 단순한 원인-결과 관계이다. NOT 관계는 원인이 참이라면 결과가 거짓이고, 원인이 거짓이라면 결과가 참인 관계이다. OR 관계는 여러 개의 원인 중 하나라도 참이라면 결과가 참인 관계이다. AND 관계는 여러 개의 원인이 모두 참이어야 결과가 참인 관계이다.

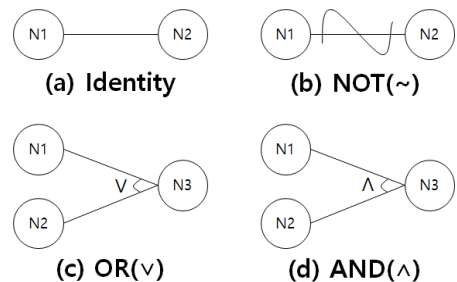


그림 1 Cause-Effect Graph의 연결 관계





2.2 C3Tree Model

C3Tree Model[3]은 복잡한 요구사항 문장의 간소화

과정을 트리 구조로 표현한다. 표 1은 C3Tree Model의 구성요소에 대해 설명한다.

표 1 C3Tree Model의 구성요소

노드	설명
<<Sentence>> Korean Sentence	요구사항 문장
<<Clause>> Korean Sentence	간소화된 문장
<<Complex-Clause>> Korean Sentence	여러 개의 Cause 노드를 포함하는 문장

연결관계	설명
	긍정적 조건 관계 (좌측 자식 노드가 참이라면, 우측 자식 노드가 참)
	부정적 조건 관계 (좌측 자식 노드가 참이라면, 우측 자식 노드가 거짓)
	AND 관계 (모든 자식 노드가 참이라면, 부모 노드가 참)
	OR 관계 (모든 자식 노드 중 하나라도 참이라면 부모 노드가 참)

2.3 자카드 유사도

자카드 유사도[5]는 두 집합 사이의 유사도를 측정하는 방법이다. 자카드 지수(J)는 0~1 사이의 값을 가진다. 두 집합의 유사도가 동일하면 1의 값을 가지고, 두 집합이 전혀 유사하지 않으면 0의 값을 가진다. 집합 A, B에 대해 자카드 유사도를 측정하는 수식은 아래의 그림 2와 같다.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

그림 2 자카드 유사도 계산식

3. 자카드 유사도 분석을 이용한 한국어 요구사항 유사도 자동 분석 방법

한국어 요구사항을 간소화를 위해, 간소화 과정을 C3Tree Model로 표현하고, C3Tree Model에서 유사한 노드를 합친다. 마지막으로 C3Tree Model을 Cause-Effect Graph로 변환하는 과정을 설명한다.

3.1 복잡한 요구사항의 간소화

한국어 요구사항의 간소화 과정은 아래와 같다. 분석하는 문장의 구성에 따라 과정의 순서가 바뀔 수 있다.

- 긍정적 조건절, 부정적 조건절 식별
- AND 연결절, OR 연결절 식별

- 주어 복구
 - 수동문을 능동문으로 변환
- 간소화 과정의 예는 다음과 같다. 표 2는 예제로 사용되는 요구사항 문장이다.

표 2 요구사항 문장의 예제

A가 입력되고 B가 입력되면 C를 출력하다.

표 3은 표 2 요구사항 문장의 형태소 분석 결과이며, 16개의 형태소가 분석된다.

표 3 요구사항 문장의 형태소 분석 결과

형태소	종류(태그)	형태소	종류(태그)
A	외국어(SL)	되	동사파생접미사(XSV)
가	주격조사(JKS)	면	연결어미(EC)
입력	일반명사(NNG)	C	외국어(SL)
되	동사파생접미사(XSV)	를	목적조사(JKO)
고	연결어미(EC)	출력	일반명사(NNG)
B	외국어(SL)	하	동사파생접미사(XSV)
가	주격조사(JKS)	다	종결어미(EF)
입력	일반명사(NNG)	.	마침표(SF)

표 4는 표 3을 사용하여 문장의 절을 식별한 결과이며, 문장의 절은 일반적으로 연결어미(EC)를 기준으로 나누고, 결과절의 경우 수동문이 능동문으로 변환된다.

표 4 간소화된 문장

절	세부 절	단순화된 문장
원인절	선행절	A가 입력되다
	후행절	B가 입력되다
결과절		C가 출력되다

3.2 C3Tree Model의 생성

요구사항 문장의 간소화 과정은 C3Tree Model로 나타낼 수 있다. 완성된 C3Tree Model은 XMI 코드로 표현된다. 표 4의 문장 간소화 과정을 C3Tree Model로 나타낸 결과는 그림 3과 같다. 총 3개의 Clause 노드가 생성된다.

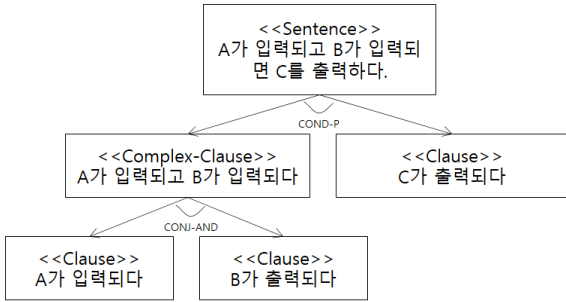


그림 3 C3Tree Model 예제

3.3 자카드 유사도를 이용한 C3Tree Model의 말단 노드 유사도 분석

C3Tree Model이 생성되고, 모델 내의 모든 Clause 노드들 간의 자카드 유사도를 분석한다. 두 문장의 주어, 동사, 목적어가 같고, 수식어 또는 일반 명사에 해당하는 형태소들의 자카드 유사도가 0.6 이상이라면 유사한 문장이라고 판단한다.

표 5는 두 개의 요구사항을 나타낸다. 그림 4는 표 5에 대한 C3Tree Model을 나타내고, 가장 유사도가 높은 두 개의 노드를 표현한다.

표 5 두 개의 요구사항 문장 예제

A가 입력되고 B가 입력되면 C를 출력하다.
D가 입력되고 C가 출력되면 E가 출력되다.

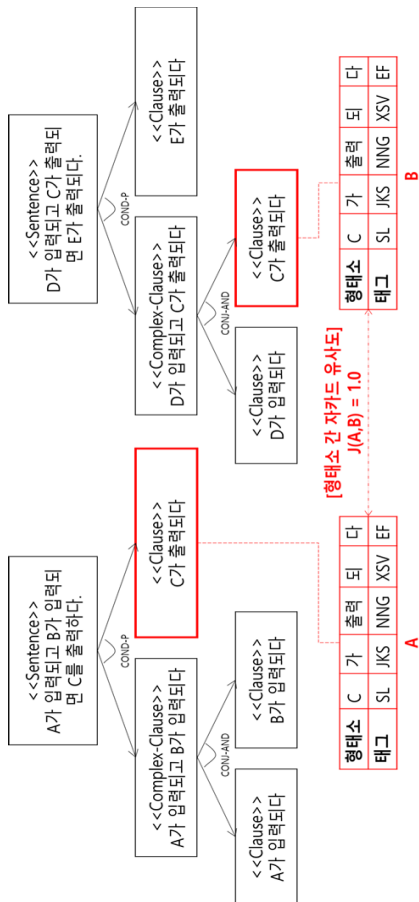


그림 4 자카드 유사도 계산 예제

3.4 C3Tree Model로부터 Cause-Effect Graph 자동 생성

마지막으로 자카드 유사도 지수가 높은 Clause 노드들을 통합 후 Cause-Effect Graph로 변환한다. 완성된 Cause-Effect Graph는 XMI 코드로 표현된다.

그림 5는 그림 4로부터 자동 생성된 Cause-Effect Graph를 나타낸다. C3Tree Model의 Clause 노드 6개가 5개의 Cause-Effect Graph 노드로 표현된다.

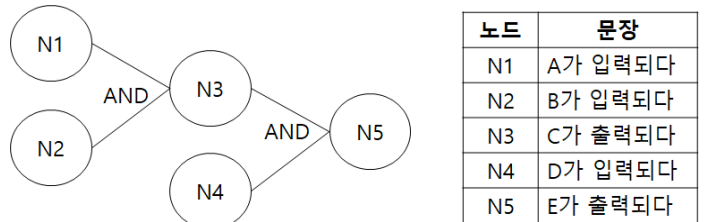


그림 5 Cause-Effect Graph 예제

4. 적용 사례

본 논문에서 제안하는 방법에 대한 적용 사례는 다음과 같다. 표 6은 두 개의 요구사항 예시이다.

표 6 요구사항 1과 요구사항 2

1. 프로그램이 켜지고 사용자가 확인 버튼을 클릭하면 1~10 사이의 난수를 발생시킨다.
2. 1부터 10 사이의 난수가 발생되면, 해당 레벨의 게임을 시작한다.

그림 6과 그림 7은 표 6의 요구사항으로부터 생성된 C3Tree Model이다.

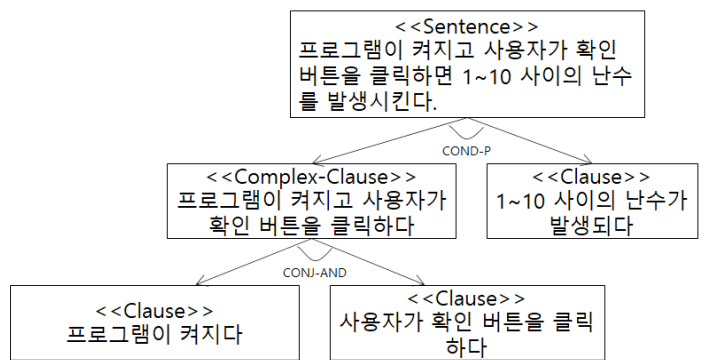


그림 6 요구사항 1의 C3Tree Model

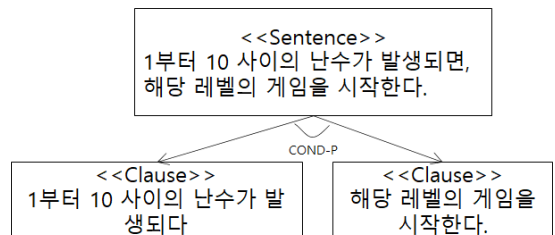


그림 7 요구사항 2의 C3Tree Model

그림 8은 그림 6, 그림 7에서 가장 유사도가 높은 두 개의 노드들의 형태소를 나타낸다. 보조사(JX)와 기호(SY)를 제외하고는 모두 같은 형태소를 가진다.

문장A: 1부터 10 사이의 난수가 발생된다

형태소	1	부터	10	사이	의	난수	가	발생	되	다
태그	SN	JX	SN	NNG	JKG	NNG	JKS	NNG	XSV	EF

문장B: 1~10 사이의 난수가 발생된다

형태소	1	~	10	사이	의	난수	가	발생	되	다
태그	SN	SY	SN	NNG	JKG	NNG	JKS	NNG	XSV	EF

그림 8 두 Clause 노드 간의 형태소 차이

표 7은 그림 8의 두 노드 간 자카드 유사도를 계산한 결과이다. 주어, 동사, 목적어가 같고, 형태소의 자카드 유사도 지수가 0.82이기 때문에 두 문장은 유사하다고 판단한다.

표 7 자카드 유사도 계산 결과

$$J(A,B) = 9 / (10 + 10 - 9) = 0.82$$

C3Tree Model에서 유사도가 높은 노드를 통합하고 Cause-Effect Graph로 변환한 결과는 그림 9와 같다.

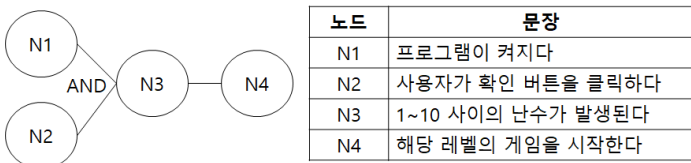


그림 9 생성된 Cause-Effect Graph

5. 결론 및 향후 연구

본 논문은 한글 요구사항으로부터 테스트케이스 생성 과정의 중간 단계이다, 1) 요구사항을 간소화하고, 2) 간소화 과정을 C3Tree Model로 표현하고, 3) C3Tree Model에서 유사도를 측정하여 통합하고, 4) 통합된 결과를 Cause-Effect Graph로 변환하는 방법을 제안한다. 모든 과정은 알고리즘 기반 변환 방법을 사용하고, 모든 모델은 XMI 코드로 저장된다. 그렇기 때문에 요구사항으로부터 Cause-Effect Graph 생성의 자동화가 가능하고, 모델의 XMI 코드 분석을 통해 모델 생성 과정에 대한 자동 추적이 가능하다.

향후 연구로써, 제안한 방법을 프로그램으로 구현하여 자동화하고, Cause-Effect Graph 생성의 정확도를 검증할 것이다.

ACKNOWLEDGMENT

이 논문은 교육부 및 한국연구재단의 4단계 두뇌한국21 사업의 지원(F21YY8102068)과 2021년도 정부(교육부)의 재원으로 한국연구재단의 지원(No.

2021R111A305040711, No. 2021R111A1A01044060)을 받아 수행된 연구임.

참고문헌

[1] 권오승, 홍사능, "로그 기반 효과적 반복 테스트", 한국경영정보학회 2009년 추계학술대회, pp. 685-690, 2009.11.
 [2] 장우성, 김영철, "말뭉치 정규화 의미 규칙 기반 요구사항 정제를 통한 원인-결과 그래프 자동 생성", 한국정보처리학회, Vol. 26, No.2, pp.691-693, 2019.11.
 [3] 장우성, 박보경, 김영철, "비정형 요구사항으로부터 생성된 원인-결과 그래프의 검증을 위한 C3Tree 모델 설계", 국방소프트웨어학회, 2019.12.
 [4] Myers, G. L., "The Art of Software Testing", Wiley-Interscience, 1979.
 [5] J Santisteban, J Tejada-Carcamo, "Unilateral Jaccard Similarity Coefficient.", GSB'15, 2015.

Use case 명세서 기반 자동 가이드 코드 생성

정세준^o 문소영 김영철

홍익대학교 소프트웨어공학연구소

{bvcx79, whit2, bob}@hongik.ac.kr

Automatic Guide Code Generation based on High-level Use Case Specifications

Se Jun Jung^o, So Young Moon, R. Young Chul Kim

SE Lab, Dept. of Software & Communications Engineering, Hongik University

요 약

소프트웨어의 비가시성으로 인해 요구사항과 설계의 기능들이 제대로 구현되는지 검증은 어렵다. 상위 레벨의 요구사항으로부터 설계 그리고 연속적 구현까지는 점점 더 복잡하므로 추적 및 오류 번식 발생한다. 이런 문제를 해결을 위해, 요구사항에서 상위 레벨 유스 케이스 명세서와 설계 정보인 UML 설계를 통합한 ‘설계 메타 모델’을 제안한다. 이를 통해 가이드 코드 생성과 설계 추적성을 가능하게 한다. 단지 상위 수준의 명세서와 모델로부터 가이드 코드 제공함으로써, 개발자들에게 고품질의 개발이 가능하게 될 것이다.

1. 서 론

소프트웨어는 고객의 요구사항으로부터 설계하고 그 기반으로 구현한다. 이때 소프트웨어의 비가시성으로 인해 특정 설계와 구현된 코드가 어느 요구사항에 관련하는지 추적이 어렵다. 이는 결국 초기에 발생한 문제를 인식하지 못하고 잘못된 산출물을 기반으로 단계가 진행됨을 의미한다. 이는 결국 테스트 단계에 되어야 발견이 되며 오류가 번식하는 문제가 발생한다[1]. 이를 해결하기 위해 ID를 기반으로 산출물들을 추적하고 매트릭스 표를 생성하거나 전반적인 소프트웨어 프로세스를 관리하는 연구가 있다[2][3]. 본 연구는 UML과 Usecase 시나리오를 통합한 메타 모델을 제안한다. 각각의 모델들을 Id와 관계를 통해 추적하고 추적된 클래스 다이어그램을 통해 가이드 코드를 생산한다. 가이드 코드는 추적된 모델의 정보를 바탕으로 스켈레톤 코드와 주석 및 JavaDoc을 생성한다. 이를 통해 개발자들이 설계를 기반으로 코드를 작성하고 자동화 코드 생성을 통해 문서화와 코드 작성을 편하게 돕는다.

2. 관련연구

소프트웨어 개발 생명주기에서 요구사항과 발생한 산출물들을 추적하기 위한 방법으로 추적성 매트릭스 기반의 시스템이 존재한다[4]. 이는 프로젝트 관리도구에 저장된 산출물 데이터베이스를 사용하여

각각의 산출물에 대한 id 값을 통해 관련된 추적 표를 생성한다. 본 연구는 이러한 데이터베이스를 사용한 추적관계를 메타 모델을 사용하여 추적한다. 메타 모델을 사용하면 모델의 통합과 모델 변화 기술을 사용할 수 있다. 예를 들어 다양한 소스 코드 언어를 파서를 통해 AST를 생성하면 서로 다른 AST의 구조가 발생한다. 이러한 서로 다른 언어의 AST를 하나의 ASTM으로 메타 모델링하여 소스 코드를 가시화할 수 있다[5]. 즉 서로 다른 AST를 하나의 ASTM 모델로 정의한 것처럼 메타모델을 사용하여 두개의 모델들을 통합할 수 있다. 또한 스마트폰 앱의 UI는 플랫폼에 따라 다른 구조를 가진다. 이를 하나의 UI Model로 통합하고 모델 변형기술을 사용하여 다중 플랫폼의 UI 파일들을 생성한다[6]. 이처럼 Model To Model, Text To Model 기술을 사용하여 메타 모델을 사용할 수 있다. 또한 UML을 확장한 모델로 시스템의 설계로써 SysML 메타 모델이 존재한다. 해당 메타 모델을 사용하여 안전이 중요한 시스템의 설계에 초점을 맞추어 요구사항들을 추적하고 검증한다[7]. 이는 시스템 프로세스에 효율적인 설계 구성과 추적을 가능하게 해준다. 이와 유사하게 본 논문은 UML 설계와 Usecase 시나리오를 통합 및 추적을 위해 메타모델 RDOM(Requirement Design Object Model)을 정의한다. 또한 최종적으로 자동으로 설계를 반영한 가이드 코드를 생성하여 설계의 이해와 구현을 돕는다. 이를 EMF(Eclipse Modeling Framework)를 사용하여 구현한다.

3. Use case 명세서 기반의 가이드 코드 생성

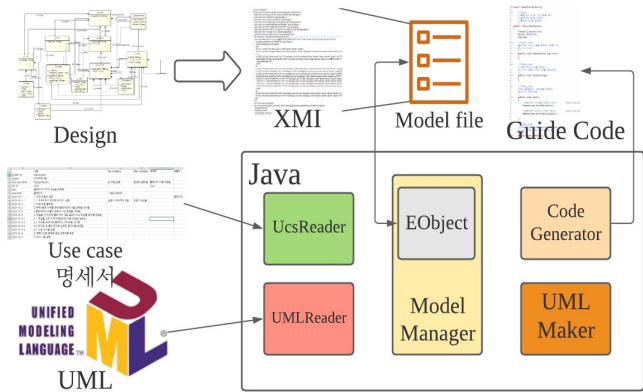


그림 1 시스템 구조

그림 1은 EMF를 통해 구현한 시스템의 구조이다. 입출력인 파일들을 제외하고 모든 과정을 JVM 내에서 처리하여 프로그램의 모듈성을 높이고 간단하게 처리할 수 있다. 먼저 유스 케이스 명세서 파일을 UcsReader 객체가 읽어 파일의 정보를 읽는다. UML의 파일을 입력하면 UMLReader 객체가 파일의 정보를 읽는다. ModelManager 객체는 XMI 형태의 모델 파일을 읽어 EObject 객체들로 변경해준다. 따라서 모델 파일의 내용을 객체로써 변경하고 수정한다. 이로서 자바 프로그램 내에서 해당 객체의 정보를 각각의 Reader가 읽은 정보들로 수정하고 생성하여 하나의 자바 프로세스로서 처리할 수 있다. 그리고 CodeGenerator 객체를 통해 최종적으로 가이드 코드를 생성한다. 또한 UMLMaker 객체를 통해 UML 설계 다이어그램을 생성이 가능하다. 이는 입력을 통해 정상적으로 모델 파일에 저장되었음을 확인하고 해당과정에서 자동으로 생성된 UML 모델들을 검증하기 위함이다. 가이드 코드의 생성은 클래스 다이어그램을 기반으로 생성된다.

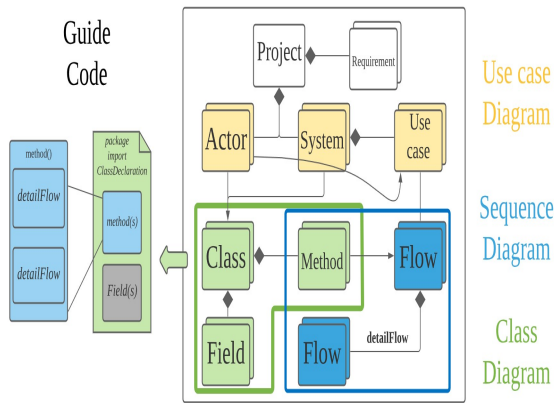


그림 2 추적 및 가이드 코드를 위한 모델 구조

그림 2는 가이드 코드의 생산과 설계와 코드의 추적을 위한 설계 메타모델의 핵심 구조이다. 먼저 Project를 최상위 모델로서 놓고 해당 Project에 대한 정보(남

기 날짜, 규모, 책임자 등등)를 저장한다. 그리고 Requirement에 대한 정보를 Project에 추가한다. 해당 Requirement은 다른 모델들이 참조 및 연결하여 관련된 요구사항을 추적하고 체크할 수 있도록 구조를 정의한다. 그리고 프로젝트에 Actor와 System, Usecase를 생성하여 저장한다. 이들은 추상화 수준이 상대적으로 높은 상위 단계인 Use case 다이어그램의 요소이다. 공통적으로 Id와 이름을 가지며 각각 필요한 세부적인 값들을 가진다. Use case의 경우 System 내부에 정의하며 유스 케이스 명세서의 시나리오의 흐름을 Flow로 정의하며 Use case 내부에 고유한 Id를 주고 정의한다. 이러한 내부 정의를 통해 관계를 정의한다. Flow는 대안이나 예외 혹은 함축적인 흐름의 경우, 상세한 내용을 위해 Flow 내부에 Flow를 정의한다. 또한 pre, post 조건 또한 해당하는 모듈의 내부에 정의한다. Flow의 흐름을 수행 요청하는 client와 수행하는 host와 같은 정보를 입력하여 저장하여 시퀀스 다이어그램을 생성할 수 있게 한다. 이렇게 저장된 상위 단계의 모델들은 최종적으로 클래스 다이어그램의 모듈들과 관계를 갖는다. Flow나 Usecase는 클래스 다이어그램의 메서드로서 구현하며 System과 Actor는 클래스로써 구현한다. 이렇게 단계별로 추적이 가능하도록 하고 상대적으로 추상화가 낮은 클래스 다이어그램을 통해 가이드 코드를 만든다. 클래스 다이어그램을 통해 클래스 파일의 정의 및 구조를 생성하고 각각의 관련된 모델의 정보로 Javadoc을 생성한다. 소스코드의 핵심인 메서드의 내용은 flow를 통해 지원한다.

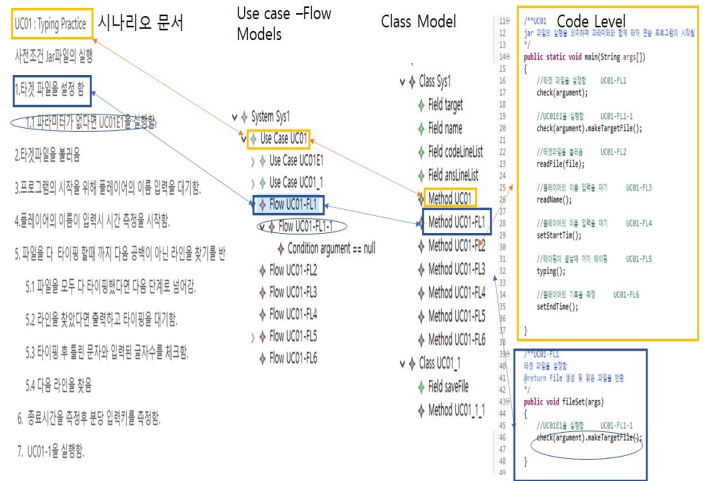


그림 3 Id 기반 추적 사례

그림 3은 Id를 통해 시나리오 문서로부터 가이드 코드가 생성되는 추적을 EMF를 통해 확인한 그림이다. 위쪽의 노란 네모는 Usecase UC01의 추적이며 아래쪽의 파란 네모는 UC01-FL1의 Id를 가진 흐름의 추적이다. 가이드 코드의 기반이 되는 클래스 다이어그램의 모델은 클래스와 필드 메서드 모델로 나뉘며 메서드와 클래스 모델은 Id값을 통해 추적이 가능한 것을 확인할 수 있다. 그림 3의 클래스 모델에서 두 개의 클래스 모델이 Sys1과 Use case의

Id를 가지는 UC01_1의 Id를 가지는 것을 그림에서 확인할 수 있다. 메서드의 경우도 Use case 혹은 Flow의 Id와 동일한 값을 가지고 있다. Usecase의 시나리오 문서는 Usecase와 그 내부의 Flow 모델로 저장되며 파란 원과 같이 1.1과 같은 대체 흐름은 Flow 내부의 Flow로 저장되어 있다. 이러한 내부 구조는 가장 우측의 가이드 코드에서 메서드 내부의 정보로도 활용되며 노란색의 Usecase는 Usecase의 흐름도가 메서드 내부로 활용된다. 이렇게 추적된 설계 정보를 바탕으로 JavaDoc과 주석을 생성하여 가이드 코드에 설계에 대한 정보와 ID 정보를 추가한 것을 확인할 수 있다. 또한 그림 3과 유사하게 각각 최종적으로 생성된 RDOM을 사용하면 동일한 가이드 코드를 생성할 수 있다.

4. 결론

본 연구를 통해 자바를 기반으로 설계 정보를 입력하면 자동으로 추적이 가능한 메타 모델을 생산하고 해당 모델을 기반으로 가이드 코드를 생성하고 UML 다이어그램을 복원하였다. 또한 JavaDoc과 주석의 생성으로 코드의 문서화를 지원한다. 이를 통해 설계를 기반으로 개발자들이 설계에 대한 이해, 구현, 문서화에 대한 시간 소요를 줄이고 설계를 기반으로 구현하도록 도와 보다 더 쉽게 개발이 가능할 것으로 기대한다. 향후 연구로는 설계한 모델을 확장하여 요구사항에 대한 설계 정보를 확장하고, 상태 다이어그램, 객체, 컴포넌트 등 다양한 설계 요소를 확장한 시스템을 만들 예정이다. 그리고 모호하고 정제되지 않은 설계 문서를 정제하여 모델을 생성할 방법과 그를 응용하여 보다 더 구체적인 설계로 확장하여 코드의 자동화를 더욱 더 높이는 연구를 계획하고 있다.

Acknowledgment

이 논문은 교육부 및 한국연구재단의 4단계 두뇌한국21 사업의 지원(F21YY8102068)과 2021년도 정부(교육부)의 재원으로 한국연구재단의 지원(No. 2021R111A305040711, No. 2021R111A1A01044060)을 받아 수행된 연구임.

참고 문헌

[1] 최규식(Che Gyu-Shik). "테스트 및 운영중 소프트웨어의 오류 원인 분석." 한국콘텐츠학회 종합학술대회 논문집 4.1 (2006): 455-458.
 [2] 박보경(Bo Kyung Park),문소영(So-Young Moon),서채연(Chae Yun Seo), 김영철(R Young Chul Kim). "국방 소프트웨어 자원관리시스템의 소스 코드 품질 향상을 위한 소프트웨어 가시화 적용 구축

사례." 한국정보과학회 학술발표논문집 2017.12 (2017): 161-163.
 [3] 권하은, 박보경, 김영철, 김영수, 이상은. "요구사항 스펙과 소스 코드 간 동기화를 위한 자동 프로젝트 문서 도구 개발", 한국정보처리학회 추계학술발표대회 논문집 2015.10: 1035-1037
 [4] 이진협. "소프트웨어 프로세스 자동화 기반의 요구사항 추적성 매트릭스 연구." 국내석사학위논문 홍익대학교 대학원, 2018. 서울
 [5] Hyun Seung Son,R Young Chul Kim. "xCodeParser based on Abstract Syntax Tree Metamodel (ASTM) for SW Visualization" INFORMATION -YAMAGUCHI: 963-968.
 [6] 손현승, 김우열, 김영철. "이종 스마트폰 앱 개발환경 구성을 위한 모델변환 적용 방법." 정보과학회논문지: 컴퓨팅의 실제 및 레터 20.4 (2014): 238-242.
 [7] 김영현(Young-Hyun Kim), 이재천(Jae-Chon Lee). "안전중시 시스템의 모델기반 설계에서 메타모델을 활용한 기능 고장의 탐지 및 안전 요구사항 검증." 한국산학기술학회논문지 17.9 (2016): 308-313.

코드가시화의 고도화 연구

김장환^o, 문소영, 김영철

홍익대학교 소프트웨어공학연구소

lentoconstante@g.hongik.ac.kr, {whit2, bob}@hongik.ac.kr

A Study on Enhancing Code Visualization

Janghwan Kim^o, So Young Moon, R. Young Chul Kim

SELab., Hongik University

요약

미래는 소프트웨어 규모가 점점 커지면서 소프트웨어의 비가시성은 소프트웨어 개발에 큰 장애물이다. 이 문제를 해결하기 위해, 사람들은 소프트웨어 아키텍처 가시화 방법을 제안한다. 이를 통해 비가시성을 극복한다. 즉 코드 내부를 가시화함으로써 소스코드의 취약점을 찾고 수정하여 소프트웨어 품질을 고도화한다. 이는 기존 가시화는 소프트웨어 규모로 인해 소스코드 크기, 복잡도 증가로 소프트웨어 아키텍처 가시화 도구의 성능이 저하되었다. 이러한 문제를 해결 위해, 소프트웨어 아키텍처가시화 고도화를 제안한다. 이를 통해 코드가시화 방법이 개선되어 소프트웨어 품질향상을 기대한다.

1. 서론

앞으로의 4차산업 분야의 소프트웨어는 그 크기가 점점 더 커지고 더 복잡해지고 있다. 이러한 이유로 인해 소프트웨어의 비가시성을 극복하기 위해 고안된 소프트웨어 코드 가시화 방법은 소스코드로부터 코드정보를 추출하여 소프트웨어의 복잡한 정도를 나타내고 이를 개선하여 소프트웨어 품질을 높인다[1]. 하지만, 기존 도구는 과거에 비해 개선이 되지 않기 때문에 방법과 도구의 변경이 불가피하다. 이러한 문제점들을 코드 가시화의 고도화를 제안한다. 본 논문의 구성을 다음과 같다. 2장은 관련 연구로써 코드 가시화를 언급하고, 3장은 기존 방법이 갖는 문제점과 개선방안을 언급한다. 마지막으로, 4장에서는 결론 및 향후 연구를 언급한다.

2. 관련연구

2.1 기존 코드 가시화 방법

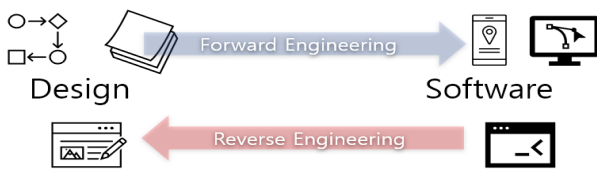


그림 1. 역공학 기반 코드 가시화

소프트웨어 아키텍처 가시화는 그림1처럼 이미 완성된 소스코드로부터 소스코드 내부의 구조를 추출해 소스코드로부터 설계를 추출한다. 구현한 소프트웨어가 기존의 설계와 같게 구현 결과를 검증한다. 또한 해당 소프트웨어를 개발한 개발자 없이도 소프트웨어에 대한 정보를 습득이 가능하다[2]. 다양한 소프트웨어 품질지표를 기반으로 소프트웨어 내부의 복잡도를 추출하여 복잡한 소프트웨어를 간결하게 소프트웨어

자체의 품질을 고도화한다. 이 방식으로 육군 정채단과 리드포인트㈜의 개발 환경 (CI)를 구축한 사례가 있다.

2.2 기존 코드 가시화 방법의 문제점

기존의 코드 가시화 방법은 오픈 소스를 이용해 소스코드로부터 그래프를 통해 가시화하는 과정을 도구들을 사용한 튜체인을 만들어 제공한다[3].

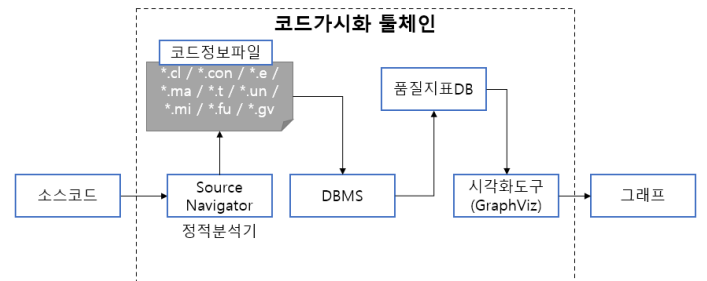


그림 2. 기존 코드 가시화 튜체인

정적분석도구인 소스네비게이터를 이용해 소스코드 정보를 추출하여 SNDB라는 파일 형태로 추출한다. 각 파일은 소스코드에 대한 고유 저장정보를 저장하고 있어 파일별로 테이블이 각각 생성되어 정보가 저장된다. 저장된 정보는 DBMS를 이용해 품질지표에 따라 계산되고 계산된 정보는 다시 품질지표 DB로 저장된다[3]. 저장된 정보는 GraphViz라는 시각화 도구를 사용하여 클래스 다이어그램, 메서드 콜 관계 그래프, 유스케이스 다이어그램 등, 그래프와 Diagram의 형태로 가시화된다.

기존의 소프트웨어 코드 가시화는 데이터베이스를 이용하여 데이터 처리를 한다. 이때, Tool-Chain은 소스네비게이터를 추출한 정보를 dbDump라는 프로그램을 이용해 텍스트 형태로 다시 추출한다. 텍스트로 된 정보들은 다시 쿼리문을 사용해 추출하고자 하는 품질지표들을 계산하기 쉬운 형태로

다시 데이터베이스에 저장한다. 소스네비게이터 추출결과물이 가져오는 단점으로 인해 테이블 크기가 커지게 된다. 이는 서로 다른 테이블을 통해 정보를 저장할 때 정보들의 속성들과 연결관계를 모두 파악하기 위해 Join문을 사용하기 때문이다.

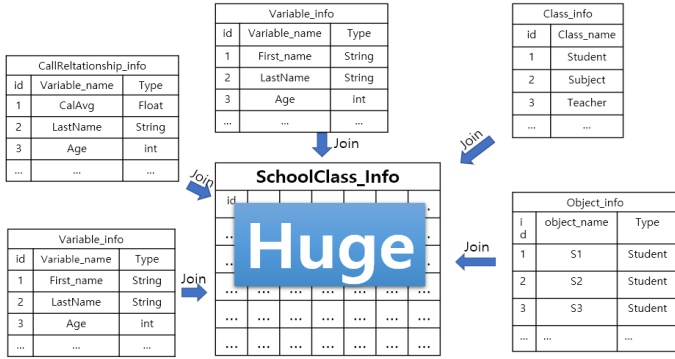


그림 3. Join문 사용으로 인한 테이블 비대화

그림3은 각 파일을 테이블화 하여 정보들을 join문 사용하여 하나의 테이블로 저장을 나타낸다. 이렇게 Join문을 사용하게 되면 소스코드가 커짐에 따라 연산 시간이 기하급수적으로 증가된다[1].

3. 코드정보 객체화를 통한 코드가시화 방안

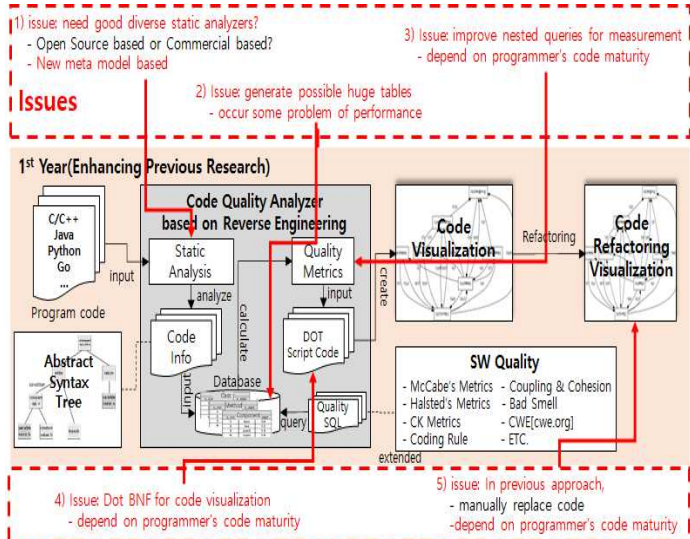


그림 4. 코드가시화 개선방안 순서도

기존 연구방식으로는 코드 규모가 증가하면, Table 크기가 커지는 문제와 그로 인해 발생하는 Query의 복잡도가 증가하는 문제가 있다. 이런 문제들을 개선하는 것은 쉽지 않다. 다시 말하면, 역공학 관점에서의 코드가시화는 이미 완성된 코드를 사용하여 상위설계단계로 추출하는 방법을 말한다. 소스 코드 데이터를 데이터베이스에 저장한 후, 여러 가지 품질지표를 측정해보는 용도라면 데이터베이스가 효과적일 수 있으나, 품질지표 계산을 위해서 다시 데이터베이스에 접근해 데이터를 추출하고 프로그램을 통해 계산한 다음 품질지표 DB에 넣는 것은 매우 비효율적이다. 즉, 소스네비게이터라는 정적분석도구를

사용함으로써 발생하는 문제이었다. 본 논문에서는 문제의 해결을 위해 추상구문트리(AST)를 추출하는 파서를 정적분석도구로 변경하고 추출된 AST정보를 객체형태로 저장하는 방법을 제안한다.

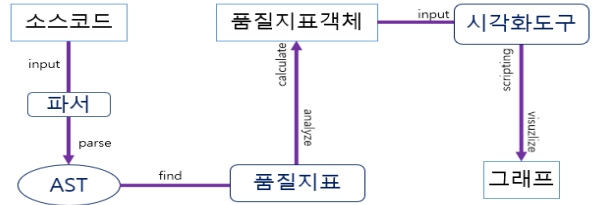


그림 5. 개선된 코드 가시화 구조도

그림 4처럼 중간 데이터베이스를 없애고 코드에서 객체 형태로 데이터를 저장하면 코드에서 데이터베이스를 연동하고 다시 추출하고 계산하는 이중 연산과정을 수행하지 않아도 된다. 위에서 언급한 복잡한 테이블을 탐색하거나 Join으로 인해 테이블 크기가 거대해지는 현상을 피하게 만든다. 또한, 데이터를 객체에 저장한 이유는 객체에 저장된 데이터 정보들을 바탕으로 곧바로 품질지표에 적용하기 편리하기 때문이다. 틀 체인에서 품질지표를 계산할 경우, 마치 프로그래밍 하듯이 품질지표 계산을 사용할 수 있다.

4. 결론

본 논문은 코드정보 객체화를 통한 코드 가시화 방법에 대한 연구이다. 기존 가시화 방법의 문제점을 개선하여 보다 빠르고 효율적으로 코드를 가시화한다. 향후, 소스코드에 대한 다양한 품질지표를 학습하여 패턴을 찾고 자동으로 리팩토링 하는 연구를 통해 소프트웨어 품질 고도화를 기대한다.

Acknowledgement

이 논문은 교육부 및 한국연구재단의 4단계 두뇌한국21 사업의 지원(F21YY8102068)과 2021년도 정부(교육부)의 재원으로 한국연구재단의 지원(No. 2021R111A305040711, No. 2021R111A1A01044060)을 받아 수행된 연구임.

참고문헌

[1] Park, B. K., Kang, G. H., Son, H. S., Jeon, B., & Kim, R. (2020). Code Visualization for Performance Improvement of Java Code for Controlling Smart Traffic System in the Smart City. Applied Sciences, 10(8), 2880.
 [2] 정보통신산업진흥원(NIPA) 부설 SW공학센터, SW개발 품질 관리 매뉴얼(개정판), 2013
 [3] 박보경, 권하은, 손현승, 김영수, 이상은, 김영철. "소프트웨어 가시화를 통한 품질 개선 사례 연구" 정보과학회논문지, 41(11), 935- 942. (2014)

Keypoint-RCNN을 이용한 반려견 대퇴골 탈구인식을 위한 수의영상판독 시스템

김민경[○], 김지인

건국대학교 컴퓨터공학부

minkyung0114@naver.com, jnkm@konkuk.ac.kr

A reading System of Veterinary image for diagnosis femur dislocation of dogs using Keypoint-RCNN

Kim MinKyung[○], Kim Jee-In

Department of Computer Science and Engineering, Konkuk University

요 약

반려가구가 증가하고 반려견의 평균수명이 늘어나며 노령견의 증가비율이 높아졌다. 퇴행성 관절질환인 대퇴골 탈구는 주로 노령견 혹은 품종의 특성으로 대형견에서 나타나는 질환이다. 기존 수의사들이 대퇴골 탈구 진단을 위해 Norberg angle를 측정하여 대퇴골의 탈구 정도의 여부를 판단하는데 직접 좌표를 선정하여 측정하는 방식에서 나아가 keypoint-RCNN을 사용하여 Keypoint 위치를 찾아 좌표선정 오차를 없애고, 각도측정을 자동으로 수행하는 Norberg Angle 측정기반 대퇴골 탈구 판독시스템을 제안한다.

1. 서 론

최근 인공지능망의 모델이 개선되어 다양한 형태로 개발되고, 관련 하드웨어의 비약적인 발전으로 딥러닝 기술이 여러 산업군에 커다란 영향을 준다.

수의학에서도 딥러닝을 이용하여 수의영상을 분석하여 질병을 판독하는 연구를 수행하고 있다.[1,2] 노화증상이 일어나면서 발생하는 퇴행성 질환은 노령견에서 흔히 찾아볼 수 있는 질환이다. 대퇴골 탈구 증상은 퇴행성 변화로 인하여 나타나는 증상 중 하나인데 유전적인 원인으로 인하여 대형견 품종에서도 발병된다. 대퇴골 탈구를 진단하기 위해 “Norberg Angle 측정” 기법을 이용하여 대퇴골 탈구의 여부를 진단한다.

현재 수의사들은 대퇴골 탈구 진단을 위하여 미국 국립보건원과 위스콘신대학교 (LOCI, University of Wisconsin)에서 개발한 imageJ 프로그램[6] 혹은 INFINITT 소프트웨어 프로그램을 이용하여 수동으로 좌표선정을 하여 Norberg angle를 측정한다.

하지만 수의영상 전문의가 아니거나 혹은 숙련자라도 각도 측정의 기반이 되는 좌표 선정에서 발생하는 오차로 인하여 매번 측정하는 값이 미세한 오차가 발생한다. 또한 수의영상 전문 수의사들이 부족하여 수요 대비 판독 속도가 느리다.

본 논문은 Keypoint RCNN을 이용하여 대퇴골 탈구를

진단을 위한 X-ray 이미지를 판독하는 기법을 제안한다. 수의영상이미지를 통해 Norberg Angle 측정을 자동으로 수행하여 대퇴골의 탈구 여부를 판독하는 기법을 제안한다.

2. 관련연구

2.1 Norberg Angle 측정방법[4]

1. 대퇴골두에 맞는 가상의 원을 그린다.
2. 가상의 원의 중심끼리 잇는다.
3. 원의 중심에서 대퇴골두 전연을 지나도록 긋는다.
4. 내부각도를 측정한다.

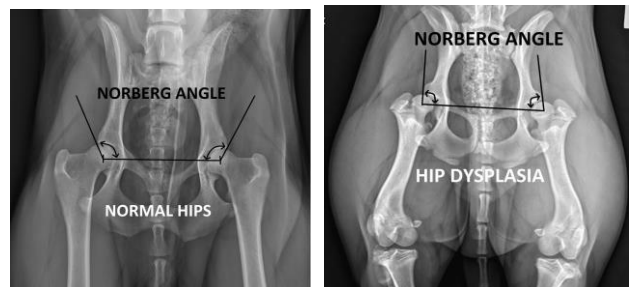


그림 1. Norberg Angle

2.2 대퇴골 탈구 진단 기준

본 논문에서 대퇴골 탈구 여부의 진단 기준으로 The

Federation Cynologique International (FCI)에서 제시한 Norberg Angle을 기준으로 한다. Norberg Angle이 105°이상은 정상, 105°미만과 90°이상은 Border line, 90°미만은 이완도 증가로 대퇴골 탈구를 3 가지 유형의 증상으로 등급을 나눈다. [3,4,5]

3. 대퇴골 탈구 판독시스템

3.1 데이터셋

반려견의 대퇴골 관절 데이터를 수집하기 위해 건국대학교 수의대학에서 반려견의 1547장의 X-ray 이미지를 제공받았다.

Norberg Angle 측정방식을 참고하여 양쪽의 대퇴골두 전연과 대퇴골두 중심을 Keypoint 데이터 형식으로 COCO-annotator[8] 툴을 이용하여 라벨링 하였다

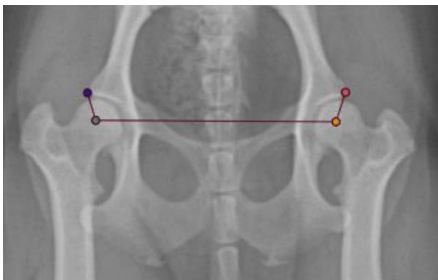


그림 2. 데이터셋 라벨링 이미지

3.2 Norberg angle 측정 시스템

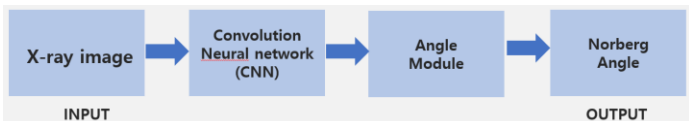


그림3. 대퇴골 탈구판독을 위한 Norberg Angle 측정시스템

본 연구는 X-ray의 이미지를 input값으로 넣어 컨볼루션 뉴럴 네트워크를 통해 Keypoint를 추출한 후 Norberg Angle 모듈을 통하여 Norberg Angle을 측정값을 기반으로 대퇴골 탈구 증상을 판독한다. 세부 과정은 그림 4와 같다.

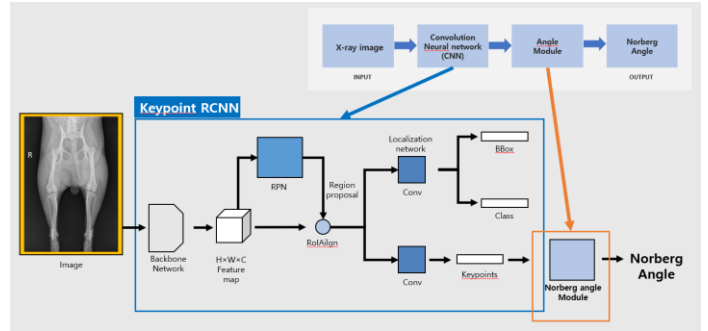


그림 4. 전체 Norberg Angle 측정 시스템 세부 구조도

반려견의 대퇴골 탈구 진단을 위한 딥러닝 모델을 Keypoint-RCNN[9]을 활용하였다. 백본 네트워크로 Resnet50[7]과 Resnet101[7]을 사용하여 피쳐맵을 추출 후 Keypoint의 위치를 찾아내고 Keypoint 값을 딥러닝 네트워크와 결합시킨 Norberg Angle Module의 input값으로 받아 각각 좌우 다리의 Norberg Angle 값이 측정된다. 각도의 값에 따라 The Federation Cynologique International (FCI)[5]기준에 의해 대퇴골 탈구 증상이 정상, Borderline, 이완도 증가 세가지 유형으로 판독된다.

3.3 평가방법

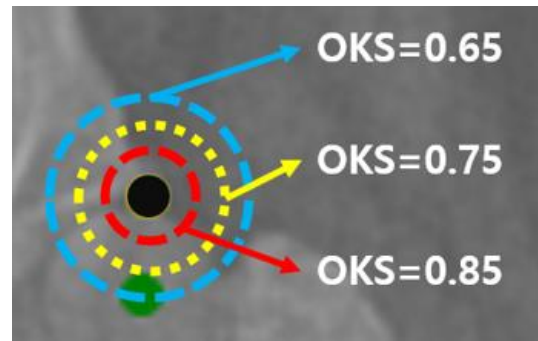


그림 5. OKS(Object Keypoint Similarity)

Object detection에서 평가지표를 위해 사용되는 IOU(Intersection over Union)는 Bound Box 형태의 객체인식 모델 성능 평가를 하는 과정에서 정량적인 평가를 위해 사용된다. 본 실험에서 keypoint 데이터셋 유형에 따라 IOU 대신 OKS(Object Keypoint Similarity)을 사용했다. OKS는 Keypoint를 중심으로 반경이 좁으면 좁을수록 OKS값이 높아지고 반면 멀어질수록 OKS 값이 낮아진다.

3.4 실험환경

AMD Ryzen 7 3700X 8-Core Processor 3.60 GHz, GPU NVIDIA Geforce RTX3080, pytorch 1.7, python 3.8 환경에서 진행되었다. batch 사이즈는 3, epoch

10000, train과 test dataset은 8:2비율로 나눠 학습하였다.

3.5 실험

표 1. 백본에 따른 실험결과

Backbone	AP ^{kp50}	AP ^{kp75}
Resnet50_1x_fpn	99.010	98.970
Resnet50_1x_fpn	99.007	98.982
Resnet101_3x_fpn	99.395	98.969
Resnet101_3x_fpn	99.000	98.991

데이터의 백본으로 Resnet과 FPN을 결합한 형태로 사용하였고 데이터 증강기법을 이용하여 대칭을 이루고 있는 이미지를 절반 나눠 데이터를 2배로 증강시켜서 학습을 시킨 결과가 표 1이다. Resnet50[7]과 Resnet101[7]을 백본으로 사용한 결과 AP^{kp50}(Average Precision Keypoint)과 AP^{kp75}(Average Precision Keypoint)가 90 이상을 얻은 결과를 볼 수 있다. 그림 6은 각 키포인트, 바운드 박스를 시각화한 것이다.

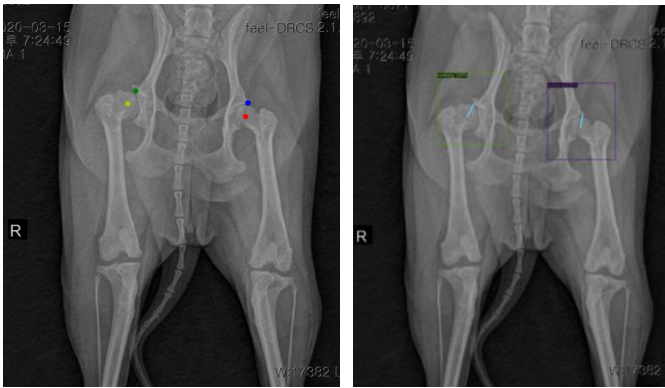


그림 6. Keypoint, bound box시각화

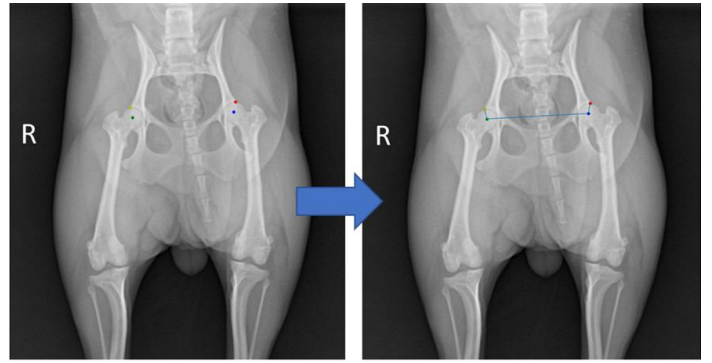
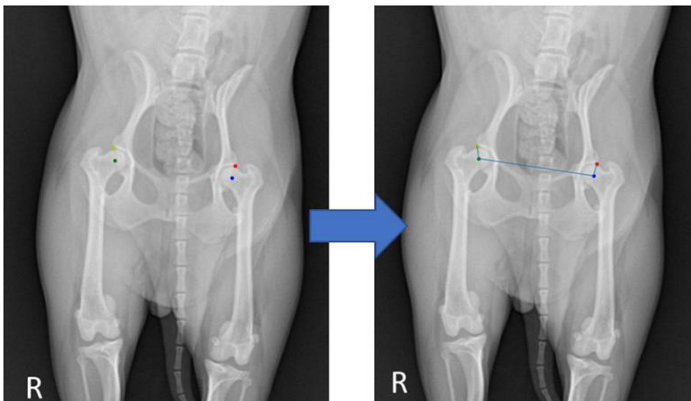


그림 7. 추론된 Keypoint와 Norberg Angle

표 2. 추론 결과

Img_name	Infer _right	Infer _left	Right등급	Left등급
1883.jpg	91.97	90.33	Borderline	Borderline
1884.jpg	88.61	100.5	이완도증가	Borderline
1885.jpg	105.9	108.5	정상	정상
1886.jpg	94.67	86.81	Borderline	이완도증가
1887.jpg	179.6	122.8	정상	정상
1890.jpg	117.8	117.4	정상	정상
1892.jpg	105.6	167.3	정상	정상
1893.jpg	109.2	109.8	정상	정상
1894.jpg	97.83	118.3	Borderline	정상
1895.jpg	96.0	116.4	Borderline	정상

결과가 가장 좋았던 백본 Resnet101[7]을 사용한 Keypoint RCNN의 weight값으로 받아 추론한 결과가 표 2이다.

표 2의 항목인 Infer_right, Infer_left는 최종적으로 추론한 오른쪽과 왼쪽의 Norberg Angle값이다. Right등급과 Left등급은 The Federation Cynologique International (FCI)[5]의 판정기준에 따른 등급으로 표기하였다.

4. 결론 및 향후연구

310장의 X-ray 데이터를 활용하여 Confusion Matrix로 평가하였다. 105°이상의 정상을 True값으로 105°미만인 이완도 증가와 Borderline의 증상을 False로 하여 얻은 결과는 표 3과 같다. 오른쪽 다리 증상일치는 277개 왼쪽 다리증상일치는 281개로 오른쪽 다리는 89%, 왼쪽 다리는 91%로 정확도를 얻었다.

표 3. Confusion Matrix 추론 결과

Right angle	Predictive values		
Actual values		Positive	Negative
	Positive	104	28
	Negative	5	173

Left angle	Predictive values		
Actual values		Positive	Negative
	Positive	93	27
	Negative	2	188

추가적으로 추론된 Norberg Angle 값과 라벨링된 Norberg Angle 값이 차이가 나는 현상이 보였는데 실제 각도를 구할 때 3개의 점을 사용하여 각도를 측정하게 되는데 각각 OKS 범위 안의 드는 좌표가 나오는 것이기 때문에 각도 값이 차이가 나는 현상이 발생된다. 향후 이러한 점을 보완하여 개선해야 할 것이다.

Acknowledgement

이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (2020-0-00872, 인공지능 기반 수의 영상 의학 정보 판독 SaaS 기술 개발)

참고문헌

[1] S. Burti, V. Longhin Osti, A. Zotti, T. Banzato, Use of deeplearning to detect cardiomegaly on thoracic radiographs in dogs, The Veterinary Journal, Volume 262, 105505, ISSN 1090-0233, 2020.

[2] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.

[3] Verhooeven, G. E. C, Coopman, F., Duchateau, L., Bosmans, T., Van Ryssen, B., Van Bree, H., Interobserver agreement on the assessability of standard ventrodorsal hip-extended radiographs and its effect on agreement in the diagnosis of canine hip dysplasia and on routine FCI scoring. Veterinary Radiology, 50, 259-263, 2009.

[4] 1. Comhaire, F. H., Schoonjans, F. A., Canine hip

dysplasia: The significance of the Norberg angle for healthy breeding. Journal of Small Animal Practice, 52, 536-542., 2011.

[5] Kapatkin AS, Fordyce HH, Mayhew PD. Canine hip dysplasia: the disease and its diagnosis. Compend Contin Educ Vet 24: 526-537., 2002.

[6] ImageJ, "https://imagej.nih.gov/"

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," IEEE conference on computer vision and pattern recognition (CVPR), pp. 770-778, 2016.

[8] Jsbroks, "https://github.com/jsbroks/coco-annotator.git"

[9] Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick, Mask R-CNN, International Conference on Computer Vision (ICCV), 2017.

히스토리 기반 타임뱅크 봉사 파트너 추천 알고리즘

선다영¹, 손서락², 양혜란^{2,3}, 유원상¹

¹ 선문대학교 정보통신공학과

² 사단법인 타임뱅크코리아, ³썬시건의 꽃밭

kkbo5222@naver.com, yarnmatch@gmail.com,

timecrowds@gmail.com, wyou@sunmoon.ac.kr

History-based TimeBank Volunteering Partner Recommendation Algorithm

Da-Young Seon¹, Seorak Son², Hyeran Yang^{2,3}, Wonsang You¹

¹Dept. of Information and Communications Engineering, Sun Moon University

²TimeBank Korea, ³TimeGarden Inc.

요 약

타임뱅크(timebank)는 봉사활동을 시간 단위의 화폐로 환산하여 상호 호혜적인 봉사활동을 지향하는 시스템이다. 그러나 많은 가입자 중에서 봉사거래를 위한 파트너를 코디네이터의 도움 없이 스스로 찾기는 쉽지 않다. 본 논문에서는 타임뱅크 플랫폼에서 거래 히스토리에 기반하여 자신이 원하는 봉사 거래를 위한 최적의 파트너를 자동으로 추천하는 알고리즘을 제안하였다. 고정기간 동안 이루어진 봉사거래의 횟수에 따른 sigmoid형 스코어를 정의하고, 스코어가 가장 높게 매겨진 순위로 봉사 파트너를 추천하는 방식이다. 제안된 알고리즘을 덕성여대 타임뱅크 데이터에 적용한 결과, 추천된 봉사 파트너에 대한 Top-5 정확도는 과거 300개의 거래기록을 반영하였을 때 약 70.4%의 정확도를 보였다. 본 연구는 타임뱅크 시스템에서 빅데이터에 기반한 지능화된 봉사거래 코디네이팅의 실현가능성을 보여준다.

1. 서 론

타임뱅크(Time bank)는 봉사활동을 시간 단위의 화폐로 환산하여 상호 호혜적인 봉사활동을 지향하는 시스템이다[1]. 1980년대 후반 미국의 에드가 칸(Edgar Cahn) 박사에 의해 시작되어, 국내에서는 2004년경 경북 구미의 사랑고리를 통해 도입 후 타임뱅크 코리아를 중심으로 확산되고 있다. 타임뱅크에 대한 수요가 증가함에 따라, 국내외적으로 타임뱅크 플랫폼 개발도 활발히 진행되고 있다. 최근 Django 프레임워크 기반의 한국형 타임뱅크 웹 플랫폼 TimeCloud가 소개된 바 있다[2].

타임뱅크 플랫폼에서는 사용자들이 봉사를 주고 싶은 거래와 봉사를 받고 싶은 거래를 자유롭게 게시할 수 있다. 그러나 사용자가 많아지고 거래량이 증가함에 따라, 나에게 적합한 최적의 봉사 파트너를 찾는 작업이 난해한 일로 대두되었다. 봉사 파트너를 연결시켜주는 코디네이터도 있지만 수많은 사용자들을 관리하기에는 역부족이다.

온라인 쇼핑몰이나 영화 사이트 등에서는 개인별 선호도를 분석하여 선호할만한 상품을 고객에게 자동으로 추천하는 기술이 발전하고 있지만[3], 봉사 시스템에서는 관련 기술의 개발이 미진한 실정이다. 개인에게 최적화된 봉사 파트너를 자동으로 추천하는 기술은 더욱 쉽게 봉사를 주고 받을 수 있도록 하여 타임뱅크 플랫폼 사용자의

편의성을 강화할 것으로 기대된다.

본 논문에서는 타임뱅크 플랫폼에서 사용자에게 최적의 봉사 파트너를 자동으로 추천하는 알고리즘을 제안한다. 제안된 알고리즘은 사용자의 거래 히스토리에 기반하여 추천 스코어를 정의하고 스코어가 가장 높은 상위 사용자들을 추천하는 방식이다. 제안된 방식은 덕성여대 타임뱅크 데이터에 적용하여 성능을 검증하였다.

2. 방 법

모든 타임뱅크 구성원에 대하여 파트너를 일대일 매칭시키는 양방향 매칭 문제(Stable matching problem)로 볼 때, 대표적인 매칭 알고리즘으로는 Gale-Shapley 알고리즘이 있다[4]. 그러나 이 방법을 적용하기 위해서는 각 구성원이 선호하는 대상자의 우선순위 정보가 있어야 한다. 본 논문에서 제안된 봉사 파트너 추천 알고리즘은 거래 히스토리 기반 스코어에 기반하여 대상자의 우선순위 정보를 예측하는 방법으로 이해할 수 있다.

타임뱅크 플랫폼에서 사용자 추천 프로세스는 그림 1에 요약되어 있다. 요청자는 타임뱅크 플랫폼 인터페이스를 통해 거래유형(봉사를 주기를 원하는지 또는 받기를 원하는지 여부)과 봉사의 종류(병원동행, 차량이동, 심부름, 청소 등)를 제시한다. 제안된 알고리즘은 두가지 사용자 입

력정보를 바탕으로 최적의 봉사 파트너를 추천하기 위해 데이터베이스에 접속하여 모든 사용자의 거래 내역을 조회한다.

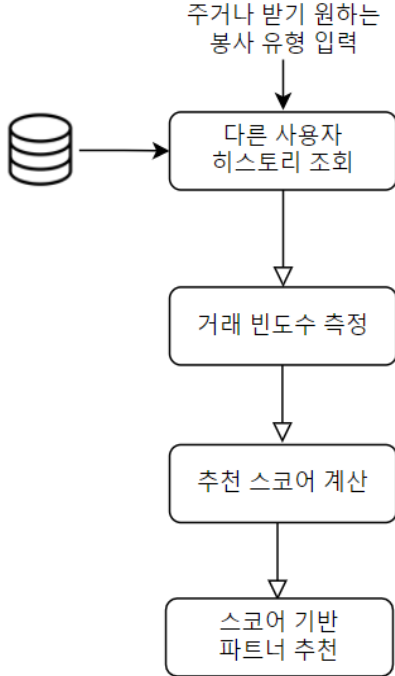


그림 1 사용자 추천 프로세스

거래내역을 바탕으로 최적의 봉사 파트너를 추천하기 위해, 어떤 대상자의 파트너 추천 스코어(partner recommendation score, PRS)를 다음과 같이 정의한다. PRS는 직접거래 기반 추천 스코어 (direct transaction-based PRS, D-PRS)와 자산 거래 기반 추천 스코어 (property transaction-based PRS, P-PRS)의 합산으로 정의된다.

$$S = S_{hd} + S_{hp}$$

D-PRS (S_{hd})와 P-PRS (S_{hp})는 다음과 같이 sigmoid형 함수로 정의된다.

$$S_{hd} = \frac{2}{1 + e^{-f_d}} - 1$$

$$S_{hp} = \frac{2}{1 + e^{-f_p}} - 1$$

모든 구성원의 봉사 거래 기록에 대하여 현재 시점부터 거슬러 올라가 일정 거래량만큼 이루어졌던 기간을 봉사 히스토리 기간(history interval)이라 하고, 거래가 이루어

진 봉사의 종류를 자산(property)이라 정의하자.

이때, f_d 는 봉사 히스토리 기간에 걸쳐서 자산의 종류와 무관하게 요청자와 대상자 사이에 이루어진 직접 거래의 횟수를 의미한다. 직접 거래의 횟수 f_d 가 0이면 D-PRS (S_{hd})는 0이 되고, f_d 가 무한대로 증가함에 따라 D-PRS는 1에 수렴한다.

마찬가지로, f_p 는 (요청자가 의뢰한) 해당 자산에 대하여 봉사 히스토리 기간에 걸쳐서 대상자가 상대와 무관하게 거래한 횟수를 의미한다. 이때, 자산에 대한 거래 횟수 f_p 가 0이면 P-PRS (S_{hp})는 0이 되고, f_p 가 무한대로 증가함에 따라 P-PRS는 1에 수렴한다.

이와 같이 모든 대상자에 대하여 PRS를 계산한 후, 내림차순으로 정렬하고, 상위 Top-1, Top-3, 또는 Top-5 등의 방식으로 추천한다.

3. 결 과

알고리즘의 성능을 평가하기 위한 데이터로 2019년 9~12월 덕성여자대학교의 <시장경제의 이해> 강의에서 57명의 수강생으로부터 수집된 타임뱅크 활동 데이터를 사용하였다. 수집 데이터에는 거래자, 거래유형, 봉사종류, 거래일시, 거래장소, 거래시간 등의 정보가 포함되어 있다.

366개의 타임뱅크 거래 활동을 네트워크로 시각화하면 그림 2와 같다. 사용자 간 거래가 많이 발생할수록 연결선 색상이 진해지도록 하였다.

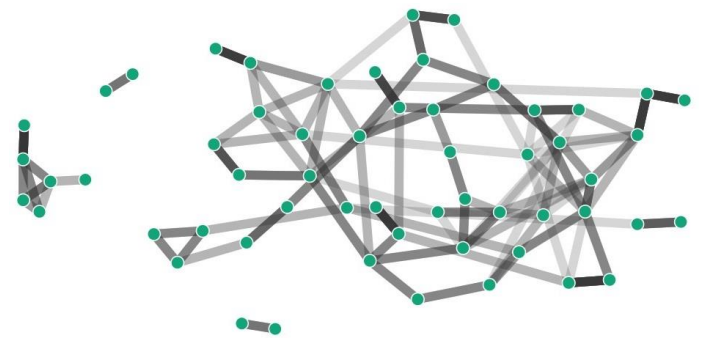


그림 2 덕성여대 타임뱅크 거래활동

추천 알고리즘의 성능을 평가하기 위해, 덕성여대 거래내역의 각 거래에 대하여 제안된 알고리즘을 적용하여 추천 파트너를 예측한 후 실제 거래한 파트너가 추천 파트너 그룹에 포함되는지 여부를 거래에 걸쳐 평균하여 추천 정확도(accuracy)로 정의하였다. 그림 3-4는 Top-1와 Top-5 추천방식에 대한 정확도가 과거 히스토리의 기간에 따라 어떻게 변화하는지 여부를 보여준다.

PRS 또는 D-PRS를 적용하였을 때 히스토리 기간이 길어짐에 따라 추천 정확도는 상승하다가, 히스토리 기간이 300일 때 Top-1의 경우 각각 46.6%와 46.3%, Top-5의 경우 70.4%와 72.3%로 수렴하였다. 반면, P-PRS를 적용하였을 때는 히스토리 기간이 길어짐에 따라

추천 정확도는 Top-1에서는 감소하는 경향이 있었고 Top-5에서는 증가하다가 43.8%로 수렴하였다.

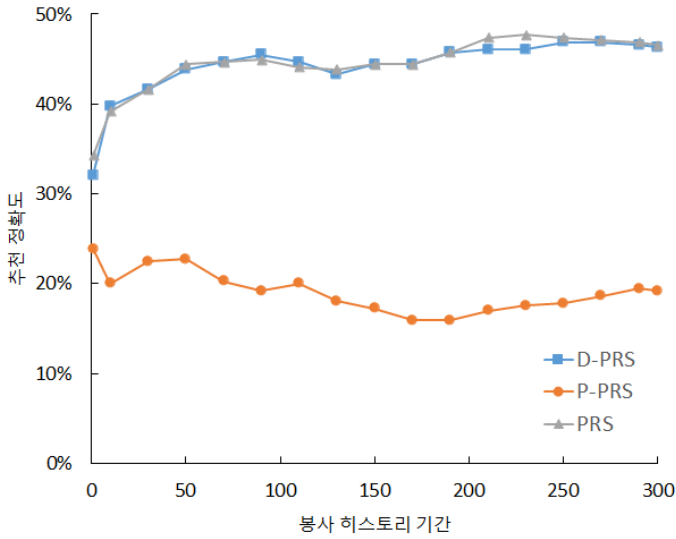


그림 3 Top-1 추천에 대한 추천 정확도

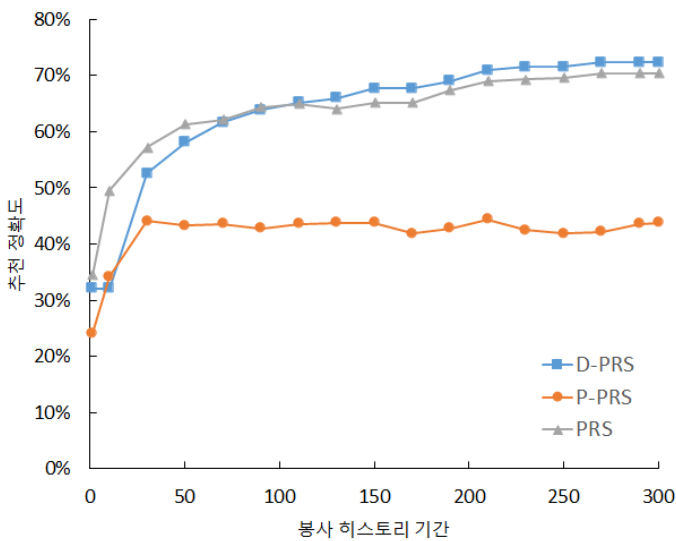


그림 4 Top-5 추천에 대한 추천 정확도

히스토리 기간에 따른 PRS 및 D-PRS의 상승 패턴은 과거 히스토리, 특히 요청자와 봉사 대상자와의 거래 히스토리를 반영할 때, 봉사 파트너 추천의 정확도가 높아짐을 방증한다.

D-PRS에 기반한 추천의 정확도가 P-PRS보다 월등히 높다는 결과가 의미하는 것은 사용자가 봉사 파트너를 선정할 때 상대방이 줄 수 있는 자산이 무엇인지에 따라 결정하기 보다는 자산과 무관하게 이전에 자신과 거래한 사람과 계속해서 재거래를 할 확률이 높다는 것을 의미한다.

5. 결론

본 연구에서는 타임뱅크 플랫폼에서 사용자에게 최적의 봉사 파트너를 자동으로 추천하는 알고리즘을 제안하였다. 제안된 알고리즘은 덕성여대 타임뱅크 데이터에 적용하여 성능을 검증하였다.

실험결과 파트너 추천 알고리즘의 정확도는 70%에 달하였지만, 실용화를 위해서는 만족스러운 결과라고 볼 수는 없다. 선호 봉사의 종류, 거래시간, 거래장소 등 다양한 정보를 활용한 알고리즘의 개선, 기계학습 모델 등을 통하여 추천의 정확도를 높일 수 있을 것으로 기대된다.

덕성여대 타임뱅크 데이터에 적용한 실험결과는 타임뱅크 사용자의 봉사 거래의 경향성을 보여준다. 즉, 봉사 파트너를 선정할 때 상대방이 줄 수 있는 자산에 따라 결정하기 보다는 자신과 거래한 사람과 계속해서 재거래를 하게 될 확률이 높다는 점이다. 즉, 해당 실험군에서는 봉사 파트너의 신뢰성이 파트너를 결정하는데 있어서 더욱 중요한 요소로 작용하였음을 간접적으로 유추할 수 있다.

그러나 실험의 특성상 학교라는 제한된 환경에서 이루어졌고 친한 친구 사이에 주로 거래가 이루어졌을 수 있음을 유추해 볼 때, 이러한 봉사 거래의 경향성을 일반화할 수는 없다. 후속 연구를 통해 다양한 커뮤니티로부터 취득된 빅데이터를 분석하여 봉사 파트너의 신뢰성이 파트너 선정에 미치는 영향을 검증할 필요가 있다.

본 연구는 향후 빅데이터 분석 및 인공지능 기술과 결합하여 타임뱅크 플랫폼과 봉사거래 시스템에서 지능화된 코디네이팅을 이끌어가는 중요한 밑거름이 될 것으로 기대된다.

Acknowledgments

본 논문은 2021년도 (주)비타소프트의 지원을 받아 수행된 산학공동연구과제의 연구결과로 수행되었음 (한국형 타임뱅크 웹 플랫폼 구축).

참고문헌

- [1] 김정훈, 이다경, "타임뱅크(Time Bank)를 활용한 복지서비스의 혁신", 이슈&진단, 310호, pp. 1-25, 2018.
- [2] 선다영, 손서락, 양혜란, 유원상, "Django 기반 한국형 타임뱅크 웹 플랫폼", KICSP, 2021
- [3] 윤호민, 최규완, "사용자 선호기반 개인화 음식메뉴 추천 기법 연구", 호텔경영학연구 제29권 제1호 (통권 제121) pp. 83~100, 2020.01
- [4] D. Gale, L. S. Shapley, "College Admissions and the Stability of Marriage," The American Mathematical Monthly, Vol. 69, No. 1, pp. 9-15, 1962.

멀티태스크 PLC 프로그램의 테스트를 위한 주기 별 테스트 시퀀스 실행 제어 방법*

은형석^o Lingjun Liu 지은경 배두환
한국과학기술원 전산학부
{hseun, riensha, ekjee, bae} @se.kaist.ac.kr

An execution control method of test sequences per cycle for multi-task PLC programs

Hyeongseok Eun^o Lingjun Liu Eunyoung Jee Doo-Hwan Bae
School of Computing, Korea Advanced Institute of Science and Technology (KAIST)

요 약

PLC(Programmable Logic Controller)는 산업용 제어기로, 안전이 중요한 시스템에서 제어 작업을 수행하는데 적용되며 주기적으로 실행된다는 특징을 지닌다. 멀티태스크 PLC 프로그램의 경우 태스크의 우선순위에 따른 상태전환 문제와 지속적으로 실행결과가 바뀌는 특성으로 인해 멀티태스크 PLC 프로그램을 PLC 내에서 주기 별로 테스트하는 것은 어려운 문제였다. 본 연구에서는 멀티태스크 PLC 프로그램의 주기 별 테스트 시퀀스 실행을 제어하기 위해, 태스크 상태 및 Standard Function Block과 User Function Block의 내부 메모리 변수를 조절하는 방법을 제안한다. 제안한 방법을 통해 주기 별 실행 제어가 가능하여 보다 정밀한 테스트 모니터링이 가능함을 보인다.

1. 서론

원자력발전소 보호 시스템과 같은 안전 중요 시스템 구현에 PLC(Programmable Logic Controller)가 사용되고 있고 이에 따라 PLC 테스트의 중요성이 높아져 관련 연구개발이 다수 진행되었다[1,2].

PLC의 고유한 특성은 프로그램이 주기적으로 지속적으로 실행된다는 점으로, 각 루프(loop)에서 입력 데이터를 읽고 프로그램을 실행하고 결과를 출력한다. 주기적 실행은 PLC를 제어 시스템에 적합하게 만들며, 예를 들어 원자로 보호 시스템(Reactor Protection System)에서의 PLC는 센서 값을 영구적으로 모니터링하고 위험 조건이 감지되면 안전 조치를 실행한다. 실제 PLC 프로그램은 복수의 태스크를 지닌 멀티태스크 PLC 프로그램으로 구성되며, 태스크의 상태와 주기(cycle)에 따라 다른 값을 지속적으로 산출하는 특성은 멀티 태스크 PLC 프로그램의 정밀한 테스트를 어렵게 하는 요소이다. 이를 극복하고 주기 별로 PLC 프로그램을 정밀하게 테스트하려면 태스크 시퀀스의 주기 별 입출력 모사가 가능해야 한다.

기존 PLC 테스트 방법으로는 주기 별 입출력 모사가 어렵기 때문에, 단순히 입력 신호를 I/O 시뮬레이터에서 모사하고 사용자가 관측 시점에서의 출력값을 확인하는 방법으로 PLC 테스트를 수행하는 한계가 있었다. 또한 기존 테스트 연구[1]에서는 단일 태스크 환경에서 PLC 테스트 베드를 구축하고 주기 별 모사를 제시하였으나, 멀티태스크 환경을 고려하지 않은 한계가 있었다.

특정 주기를 PLC 프로그램 내에서 모사하려면 태스크 상태를 고려하고 function block들의 내부변수를 모두 통제하여 모사하고 결과를 고정(freezing)해야 한다. 본 연구에서는 PLC 프로그램의 특정 주기 모사를 위해 태스크 상태를 모사하고 모든 function block 들을 통제하여 사용자가 원하는 주기의 상태를 정밀히 모사하는 체계적인 방법을 제안한다.

2. 배경 지식

2.1 Standard Function Block 과 User Function Block

FBD(Function Block Diagram) 프로그램은 내부 상태가 없는 function과 내부 상태를 가지는 function block들로 구성되며, function block은 IEC 61131-3[3]에 따라 제조사가 제작하여 사용자에게 제공하는 Standard Function Block(이하 SFB)과, 사용자가 직접 만들어 사용하는 User Function Block(이하 UFB)으로 구분된다.

* 본 연구는 원자력안전위원회의 재원으로 한국원자력안전재단의 지원을 받아 수행한 원자력안전연구사업의 연구결과이며(No. 2105030), 2019년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2019R1I1A1A01062946).”

2.2 IEC 61131-3

IEC61131-3 표준에는 PLC 표준 프로그래밍 언어로 FBD, IL(Instruction List), LD(Ladder Diagram), ST(Structured Text), SFC(Sequential Function Chart)가 정의되어 있으며 FBD와 ST, SFC 언어는 C코드로의 변환과정을 거쳐 소스코드를 생성한다[4]. 또한 SFB들을 ST언어 논리와 함께 정의하며[3] 이를 통해 SFB들이 사용하는 변수와 논리를 알 수 있다. 아래 표 1은 IEC61131-3 표준에 정의된 SFB의 예이다.

표 1. IEC61131-3의 SFB 예

Function Block 그룹명	Function Blocks	예
Bistable	RS, SR	
Edge detection	F_TRIG, R_TRIG	
Timer	TOF, TON, TP	
Counter	CTD, CTU, CTUD	

2.3 멀티태스크 및 태스크 우선순위에 따른 상태전환

태스크는 프로그램 구성 단위이며 사용자가 임의로 주기를 설정할 수 있다. PLC 프로그램 특성에 따라 복수의 태스크가 존재할 수 있으며, 이 경우 사용자가 지정한 우선순위 순서에 따라 실행된다. 예를 들어 3개의 태스크 T1, T2, T3의 우선순위가 T1>T2>T3이고 실행주기는 T2가 T1의 2배, T3가 T1의 4배라면 그림 1과 같이 멀티태스크 PLC 프로그램이 실행된다.

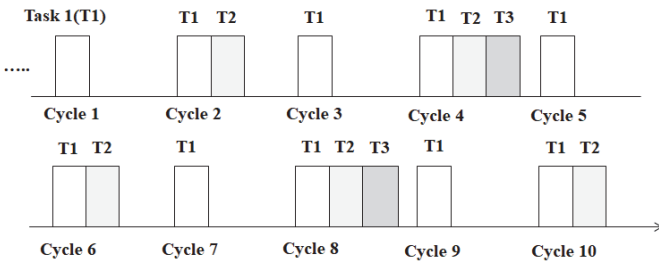


그림 1. 멀티태스크 PLC 프로그램

낮은 우선순위의 태스크는 높은 우선순위 태스크가 끝나는 시점에서 태스크가 시작된다. 만약 그림 2와 같이 T2가 끝나기 전에 우선순위가 높은 T1의 실행이 도래한다면 Context Switching 이 일어나 아직 완료하지 못한 T2는 Suspended 상태가 되고 T1이 Run 상태가

된다. 그 뒤 T2는 다시 Run 상태가 되지만 완료되지 못하여 다시 한번 Suspended 상태가 되고 4주기에서 T2가 완료됨을 볼 수 있다.

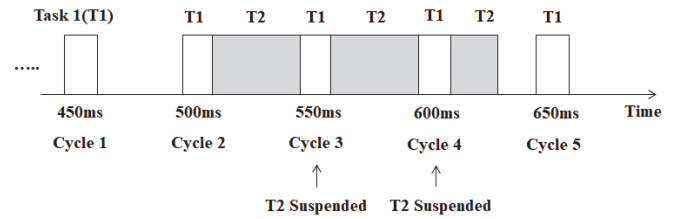


그림 2. 태스크 우선순위에 따른 상태 개념도

태스크의 Suspended 상태의 유무가 중요한 이유는, T1은 T2의 Suspended 상태를 확인하여 T2가 제공하는 값들의 사용을 일시적으로 유예하는 기능이 원자로 보호 시스템에 존재하기 때문이다. 이는 T2가 정상적으로 완료되기 전까지는 T2이 마지막으로 정상 갱신한 유효한 값들을 보수적으로 신뢰하기 때문이다.

2.4 주기 모사에 필요한 상태정보

주기 별 태스크 상태에 따른 모사가 가능하려면 태스크 상태, SFB의 상태 그리고 UFB의 상태 3가지를 모두 모사해야한다. 예를 들어 그림 2의 3주기의 상태를 정확히 모사하려면 T2가 Suspended 상태라는 정보와 해당 주기의 SFB와 UFB가 지니는 내부 메모리 변수 들을 모두 모사해주어야 한다. 다만 PLC 제조사가 제공하는 개발도구의 제한사항에 따라 3가지를 모두 모사하는 것이 불가능할 수 있다. 하지만 원자로 보호 시스템에 사용되는 POSAFE-Q PLC는 3가지를 모두 모사하는 것이 가능하여 본 연구에서 제안하는 방법을 사용할 수 있다. 기존 연구[1]에서의 테스트 드라이버와 테스트 스템을 이용하면 전 주기의 결과가 후속 주기 실행으로 인해 갱신되지 않도록 내부 메모리에 따로 저장할 수 있어 지속적인 PLC 실행환경에서도 원하는 주기의 모사결과를 확인할 수 있다. 따라서 본 연구에서는 기존 테스트 베드를 사용하여 결과를 제시한다.

3. 멀티태스크 PLC 프로그램의 주기 별 모사 방법론

3.1 테스트 상태 모사

태스크의 Suspended 상태 판정을 위해서는 그림 3과 같이, 태스크 실행상태를 지시해주는 PLC 레지스터를 사용하는 방법과 별도의 PLC 레지스터가 없는 경우 Suspended 판정 변수를 사용하는 방법이 있다.

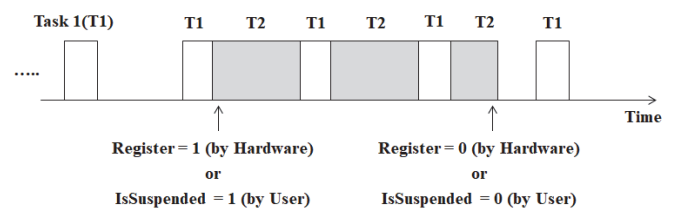


그림 3. 태스크 Suspended 상태 판정

기존 연구[1]의 테스트 베드의 특정 PLC 레지스터는 태스크가 시작되면 레지스터가 1로 설정되고 태스크 종료되면 0이 된다. 따라서 T1이 T2의 레지스터를 관측했을 때 레지스터가 1이면 T2는 Suspended 상태임을 알 수 있다.

만약 PLC가 별도의 레지스터를 제공하지 않는 경우는 T2의 가장 처음 로직에서 Suspended 판정 변수에 1을 대입하고 최종 로직에서 0을 대입하면, 이 변수를 T1이 관측했을 때 1이면 T2는 Suspended 상태임을 알 수 있다. 따라서 원하는 주기의 T1, T2 상태에 따라 레지스터 혹은 Suspended 판정 변수를 대치해주면 해당 태스크들의 상태를 모사할 수 있다.

3.2. UFB의 내부 메모리 변수 모사

UFB의 내부 메모리 변수 모사를 위해서는 그림 4와 같이 테스트 드라이버가 function block 내부 구조체에 접근할 수 있도록 extern 코드를 삽입해야 한다.

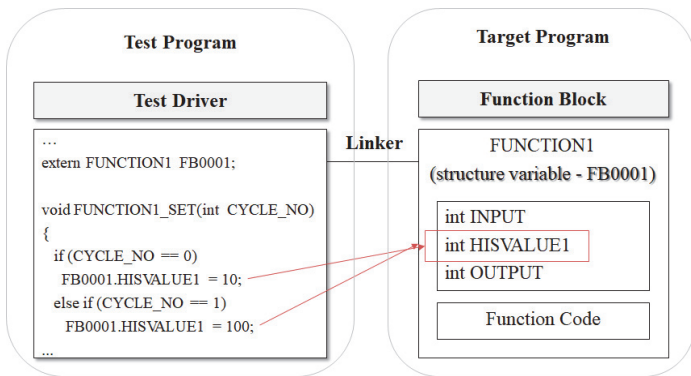


그림 4. UFB의 내부 메모리 변수 모사

Linker 가 extern 코드로 인해 Test Program 과 Target Program의 구조체 변수를 연결하므로, Test program 은 Target program의 구조체 변수에 접근할 수 있다. 이를 이용하여 Test Program에서 내부 Target Program의 상태를 조절할 수 있게 되며, function block의 입력을 모사하고 내부 메모리 변수 또한 주기에 맞게 모사하면 Test Program 상태를 주기 별로 모사할 수 있다.

3.3. SFB의 내부 메모리 변수 모사

기존 연구[1]에서는 전체 원자로 보호 시스템 프로그램 중 일부 프로그램만을 대상으로 하였으므로 이에 필요한 일부 SFB만을 모사하여 주기를 조절하였다. 하지만 실제 산업에서의 시스템들은 다양한 SFB들을 사용하므로 전체 SFB들을 모사하여 주기 별 조절이 가능하도록 해야 한다.

또한 일반적으로 PLC 제조사는 SFB의 코드를 제공하지 않으며 제조사 별로 SFB의 구현이 차이가 있을 수 있으므로, 제조사의 SFB와 같은 기능을 하는 SFB를 IEC61131-3을 참고하여 구현한 뒤 예상되는 내부 메모리 변수 변화에 따라 실제 SFB와 같이 동작하는지

확인해야 한다. 편의상 IEC61131-3을 참고하여 모사된 SFB는 simulated SFB(sSFB)로, 제조사의 실제 SFB는 read SFB(rSFB)로 축약기술한다.

J.Sim 외[4]의 접근에서처럼 sSFB의 구현은 C언어를 통해 구현되며, IEC61131-3에 정의된 ST언어를 참고하여 작성되었다. 내부 메모리 변수를 지니지 않는 ADD, OR 등의 function은 주기 별 입력 모사만 필요한 반면, 내부 메모리 변수를 지니는 CTUD(Counter Up/Down), R_TRIG, TON 등의 function block은 내부 메모리 변수까지 모사해야 주기 별 모사가 가능하다.

각 SFB의 내부 메모리 변수는 조건문에 포함된 변수 중 입력변수가 아닌 경우에 해당된다. 예를 들면 그림 5와 같이 CTUD는 CU, CD, R, LD, PV 변수가 입력되고 각 입력변수는 조건문에 쓰이게 되나, CV 변수는 입력 변수가 아니나 조건문에 사용된다. 이 경우 CV값은 내부 메모리 변수 값이며 다음 주기에서 분기에 영향을 준다.

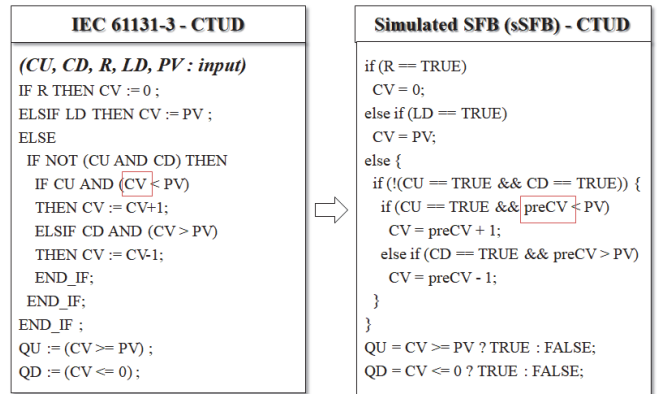


그림 5. sSFB 구현 및 내부 메모리 변수

sSFB와 rSFB에 같은 입력변수들과 내부 메모리 변수를 입력하였을 때 같은 출력을 보이게 되면, 구현한 sSFB의 동작은 rSFB와 같음을 보장할 수 있고 확인된 내부 메모리 변수를 모사함으로써 특정 주기 별 rSFB를 모사할 수 있게 된다. 아래 표 2는 sSFB와 rSFB의 일치 여부를 PLC 테스트 베드에서 확인한 결과이다.

표 2. sSFB의 내부 메모리 변수 수 및 rSFB와 일치 여부

Function Block 그룹명	Function Blocks	내부 메모리 변수 수*	sSFB와 rSFB 출력 일치여부
Bistable	RS, SR	1	일치
Timer	TOF, TON, TP	4	불일치
Edge detection	F_TRIG, R_TRIG	1	일치
Counter	CTD, CTU	2	일치
	CTUD	3	일치

* PLC 제조사의 구현에 따라 차이가 있을 수 있음.

타이머 관련 SFB인 TOF, TON, TP를 제외하면 sSFB와 rSFB는 출력이 일치하여 얻어진 내부 메모리 변수로 출력을 정확히 모사할 수 있다. 타이머 관련

sSFB가 rSFB의 결과와 일치하지 않는 이유는 rSFB는 내부 타이머 변수로 고정 전역 변수를 사용하였기 때문이다. 타이머 관련 SFB의 출력결과인 ET(Elapsed Time)는 내부 타이머가 증가함에 따라 증가하므로 시간이 지나면 항상 PT(Preset Time)와 같은 값을 출력하게 되어 ET를 고정할 수 없게 된다. 따라서 예외적으로 타이머 관련 rSFB는 전역 변수를 통제하거나 내부 변수로 통제할 수 있도록 기존 연구[1]처럼 sSFB로 실제 테스트 시 대체해야 한다.

4. 멀티태스크 PLC 프로그램 테스트 사례 연구

4.1 사례 테스트 시퀀스 정의

모든 UFB, SFB의 내부 메모리 변수를 모사할 수 있게 되었으므로 사용자는 원하는 주기를 선정하여 PLC 내에서 테스트 시퀀스를 입력하고 결과를 확인할 수 있다. 본 연구에서는 주기 별 시뮬레이션을 위해 아래와 같은 시퀀스를 작성하였으며 그림 6과 같이 도식화 하였다.

변수 정의 :

- 이상상태1(Q1) : 태스크1에서 변경
- 이상상태2(Q2) : 태스크2에서 변경
- 이상여부(R) : 태스크1에서 변경(function block)
- (R = Q1 AND Q2. 단, 태스크2가 Suspended 상태면 태스크1은 결과 R를 갱신하지 않음)

테스트 시퀀스 :

- 주기1 : 이상상태1,2 발생 (Q1=TRUE, Q2=TRUE)
- 주기2 : 태스크2 시작
- 주기3 : 이상상태1 해소 (Q1=FALSE)
- 주기4 : 태스크2 종료

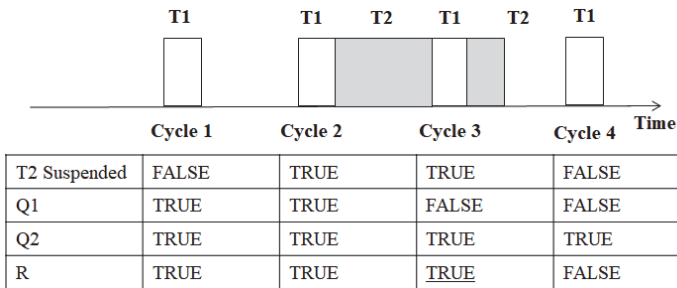


그림 6. 테스트 시퀀스 도식화

4.2 사례 테스트 시퀀스의 주기 별 실행 확인

기존 테스트 방법으로는 중간 값을 관찰할 수 없어 Q1, Q2를 입력하면 관측 시점의 R의 마지막 값만 확인할 수 있으므로, R의 중간 값을 예측하는 것이 어렵다. 또한 Suspended 상태를 모사할 수 없으므로 이를 입력으로 받는 실행결과를 모사할 수 없다. 하지만 본 연구의 멀티태스크 PLC 프로그램 테스트 방법을 이용하면 PLC 테스트 베드에서 주기 별로 테스트 시퀀스를 모사하고 확인할 수 있으며, 상세한 테스트 수행절차는 다음과 같다.

- 1) 테스트 드라이버의 주기 선택
- 2) 첫 주기의 function block의 초기값(R) 강제 지정
- 3) 태스크 Suspended 상태 및 Q1, Q2를 주기 별 대입
- 4) function block 실행 후 실행결과(R)를 출력
- 5) function block 내부 메모리 변수는 후속 주기로 복사
- 6) 주기 별 반복 및 PLC 내 결과출력(R) 확인

테스트 시퀀스는 PLC 테스트 베드에서 성공적으로 모사되었고 주기 별 입출력 결과는 그림 7과 같다.

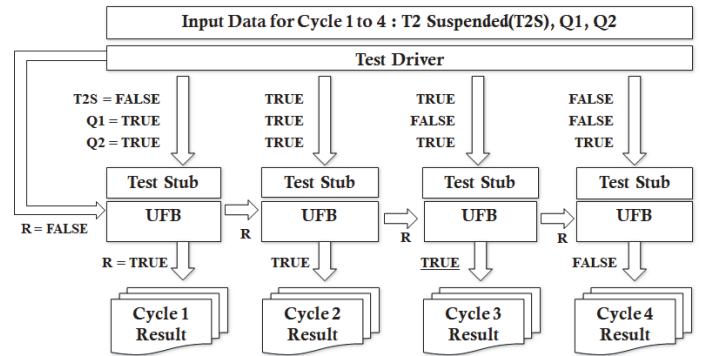


그림 7. PLC 테스트 베드 내 테스트 시퀀스 결과확인

5. 결론 및 향후연구

본 연구에서는 주기적으로 실행되는 멀티태스크 PLC 환경에서 테스트 시퀀스 실행을 제어하기 위하여, 태스크 상태, SFB 및 UFB의 내부 메모리 변수를 모사하는 방법을 제안하였다. 이를 통해 특정 주기를 정밀하게 모사하고 실행 결과를 PLC 테스트 베드를 통해 확인하는 것이 가능함을 보였으며, 기존 테스트 방법보다 정밀한 테스트 실행 가능성을 확인하였다. 향후 연구에서는 제안한 방법을 다양한 시스템 테스트 사례에 적용하여 방법의 효과성을 검증하고자 한다.

참고문헌

[1] 은형석, Lingjun Liu, 지은경, 배두환, "PLC 프로그램 테스트 시퀀스의 체계적 실행 및 출력 방법," 한국소프트웨어종합학술대회 논문집, pp. 245-247, 2021.
 [2] 지은경, 송지영, Lingjun Liu, 배두환, "안전 필수 시스템을 구현한 FBD 프로그램 대상 테스트 기법의 진화 사례 분석," 정보과학회지, 제39권, 제10호, pp.30-40, 2021.
 [3] IEC61131-3: "International Standard for Programmable Controllers -Part 3: Programming Languages," 2013.
 [4] J. Sim, C, Kwon, K Han, "Model based PLC instruction Verification Method." In Proceedings of 2011 Conference on Information and Control Systems (CICS 2011), 2011.

Concolic 테스팅과 Fuzzing을 결합한 유닛 테스팅 자동화 기술

김윤삼

브이플러스랩㈜

yunsam.kim@vpluslab.kr김문주^o

브이플러스랩㈜/KAIST

moonzoo.kim@vpluslab.kr

Automated Unit Testing Technique to Utilize Concolic Testing and Fuzzing Together

Yunsam Kim

VPlusLab Inc.

<https://vpluslab.kr>Moonzoo Kim^o

VPlusLab Inc./KAIST

<https://vpluslab.kr>

요 약

Concolic 테스팅 기술과 Fuzzing 기술은 높은 테스트 커버리지를 달성하는 테스트 입력값들을 자동으로 생성하기 때문에, 소프트웨어 오류 검출에 효과적이다. 두 기술은 서로 상이한 방식으로 동작하기 때문에 각각의 장단점이 다른 만큼, 두 기술을 함께 적용하여 테스팅 효과를 향상시킬 수 있다. 본 연구에서는 Concolic 테스팅과 Fuzzing을 조합한 새로운 기술을 개발하여, 실험을 통해 그 효과를 살펴보고자 한다.

1. 서 론

4차 산업 혁명의 중심인 소프트웨어의 복잡도가 날로 증가하고 있으며, 이로 인한 소프트웨어 오류로 인한 수많은 사건사고가 발생하고 있다 (예. 90명이 사망한 토요타 자동차 급발진 사건, 300여명이 사망한 보잉 737 MAX 추락 사건). 따라서, 소프트웨어 테스팅의 중요성이 날로 높아 지고 있으며, 소프트웨어 개발자가 수작업으로 진행되는 테스팅보다 테스트 커버리지가 높고, 예외상황 오류 검출 효과가 뛰어난 소프트웨어 자동 테스팅 기술에 대한 필요성이 부각되고 있다.

Concolic 테스팅과 Fuzzing 기술은 소프트웨어 자동 테스팅 기술 중 각광받고 있는 기술들이다.

- Concolic 테스팅은[1], 소스 코드 분석과 동적 테스팅, AI 기반의 SMT Solver를 동시에 이용하는 자동 테스팅 기법이다. 이 기술은, 소스 코드 분석 및 동적 테스팅에서 얻은 정보를 사용하여, 특정 코드 라인을 실행하기 위한 입력값의 조건을 구한 뒤, SMT solver로 조건을 만족하는 입력을 구하는 작업을, 모든 가능한 수행 경로를 다 실행할 때까지, 반복한다.
- Fuzzing 기술은[2], 타겟 소스 코드를 분석하지 않고, 테스트 입력값들을 랜덤하게 그리고 매우 빠르게 생성한다. 이 때, 해당 테스트 실행으로부터 얻어진 테스트 커버리지 정보를 바탕으로, 테스트 커버리지를 높일 수 있도록 휴리스틱을 적용하여 기존 테스트 입력값을 변이하여 새로운 테스트 입력값을 생성하는 과정을 주어진 시간동안 반복한다.

Concolic 테스팅 기술은 타겟 프로그램의 소스코드 로직을 정확하게 분석하기 때문에 높은 테스트 커버리지를 달성할 수 있는 장점이 있는 반면, 동적 실행에서 심볼릭 경로 조건식을 추출하고 추출된 복잡한 조건식을 SMT solver로 풀어내는 매우 복잡한 과정을 수없이 반복하기 때문에, 테스팅 속도가 느리다. Fuzzing은 타겟 프로그램의 소스코드 로직 분석을 하지 않고, 기본적으로 테스트 입력값을 byte 단위로 mutate해서 새로운 테스트 입력값을 만들기 때문에, 테스팅 속도가 매우 빠른 장점이 있다. 하지만, 타겟 소스코드 로직 분석을 안하기 때문에, 높은 테스트 커버리지를 달성하지 못하는 단점이 있다.

2. CONUZZ 프레임워크

본 논문에서는, 서로 다른 두 기술인 Concolic 테스팅과 Fuzzing의 장점을 합쳐 보다 높은 테스팅 효과를 얻을 수 있는 새로운 자동 유닛 테스팅 기술인 CONUZZ (CONcolic + fUZZing)를 소개한다.

2.1 CONUZZ의 장점

CONUZZ는 Fuzzing의 장점인 빠르고 다양한 테스트 입력값 생성과 Concolic 테스팅의 장점인 정확한 SW 동적 분석 능력을 결합하여, 보다 효과적인 테스팅 효과 달성을 목표로 한다. 또한, Deep path를 탐색하는 능력이 제한적인 Fuzzing의 약점 및 Concolic Testing의 핵심이지만 속도가 느린 symbolic execution 엔진을 상호 보완하고자 한다.

2.2 CONUZZ 동작 방식

그림1에서 보듯이 CONUZZ는 먼저 concolic 테스트의 depth first search (DFS) strategy를 적용하여 테스트 입력값들을 생성하고, 생성된 테스트 입력값들을 시드 테스트 풀로 사용하여 fuzzing을 적용하여 추가적으로 테스트 커버리지를 향상하는 방식으로 동작 한다.

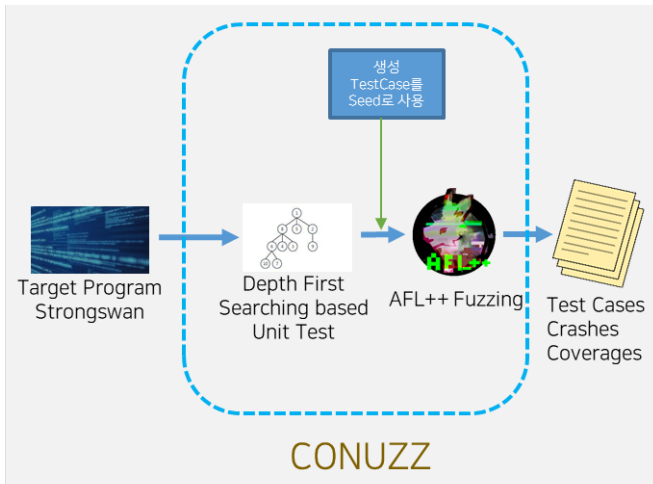


그림1. CONUZZ 프레임워크

이 때, concolic 테스트가 생성한 첫번째 테스트 입력값이 crash인 경우, 해당 실행에 해당하는 심볼릭 경로 조건식 생성이 안되기 때문에 추가적인 테스트 입력값 생성이 불가능하다. 이런 경우, 그림 2에서 보이듯이, Fuzzing에서 처음 테스트 입력값들을 생성하고, 생성된 각각의 테스트 입력값을 concolic 테스트의 초기 테스트 입력값으로 활용하여 다양한 테스트 입력값들을 생성한 후 함께 모아서 최종적으로 시드 테스트 풀로 활용하여 Fuzzing을 적용한다.

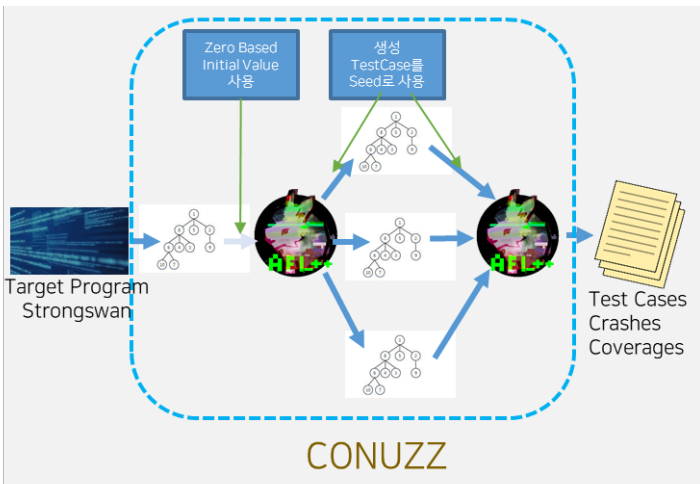


그림2. Concolic 테스트의 첫번째 TC가 crash인 경우

CONUZZ는 concolic 테스트 엔진 및 유닛 드라이버/스텝 생성 도구로 CROWN 2.0 [3]을 사용하며, AFL++[4]를 fuzzing 도구로 사용하도록 개발되었다.

3. CONUZZ 적용 실험 결과

본 연구에서는 함수 1427개/6만여줄로 구성된 보안관련 프로젝트인 S프로젝트에 CONUZZ를 적용한 결과를 분석하도록 한다. 본 실험은 AMD Ryzen 7 3800X (Ubuntu 18 64bit)에서 함수 하나당 Concolic 테스트에 최대 60초, Fuzzing에 최대 10분을 적용하였다.

3.1 Crash 오류 검출 결과

CONUZZ 실험을 통해 인증서 관련 crash 오류 등 여러 신규 오류가 검출되었다. 예를 들어, 데이터를 포함한 자료구조 T가 해당 데이터의 크기를 별도의 field인 len로 관리하는 경우에, 악의적으로 len에 실제 데이터 크기 보다 큰 값을 저장하는 경우, buffer overflow 오류가 발생함을 발견하였다 (참고.CWE-131 Incorrect calculation of Buffer Size [5]).

3.2 CONUZZ vs. Concolic 테스트

1427개의 타겟 함수 중 concolic 테스트에서 첫번째 테스트 케이스가 crash하여 유의미한 테스트를 생성하지 못했으나, CONUZZ는 유의미한 테스트 케이스를 생성하는 경우가 57개 함수, concolic 테스트에서는 crash가 발견되지 않았으나, CONUZZ는 crash 발생하는 테스트 입력값을 생성한 함수가 28개 함수, concolic 테스트 보다 CONUZZ가 더 많은 테스트 입력값을 생성한 경우가 468개였다. 따라서, CONUZZ를 통한 테스트 성능 향상을 확인할 수 있었다.

4. 결론 및 향후 연구

본 연구에서는 Concolic 테스트와 Fuzzing을 결합한 CONUZZ를 개발하여 테스트 효과를 향상시켰다. 실험을 통해, CONUZZ는 crash오류 검출에 효과적이며, concolic 테스트 단독 사용에 비해 장점이 많은 것을 알 수 있었다. 향후에는 Concolic 테스트와 Fuzzing을 좀 더 fine-granularity로 결합하여 테스트 효과를 보다 향상시키는 연구를 수행할 계획이다.

참고 문헌

[1] Baldoni et al., A Survey of Symbolic Execution Techniques, ACM Computing Surveys, Volume 51, Issue 3, May 2019
 [2] Manes et al., The art science and engineering of fuzzing : A survey. IEEE Transactions on Software Engineering, vol. 47, Nov. 2021,
 [3] Automated Concolic Unit Testing Tool CROWN 2.0, <https://www.vpluslab.kr/crown2>
 [4] AFL++ (American Fuzzy Lop plus plus), <https://github.com/AFLplusplus/AFLplusplus>
 [5] CWE-131: Incorrect Calculation of Buffer Size, <https://cwe.mitre.org/data/definitions/131.html>

소프트웨어 수출을 위한 시장주도 요구공학 적용 사례

이종길[○], 길문주, 오정섭

주식회사 엔에스이

jklee@nsetec.com, mjgil@nsetec.com, jsoh@nsetec.com

Case Study to apply Market-driven Requirement Engineering for exporting Software

Jong Kil Lee[○], Moon Joo Gil, Jung Sup Oh

NSE Technology

요 약

본 적용사례는 기존의 특정 고객의 요구사항을 기반으로 개발된 소프트웨어를 수출하기 위하여 시장주도 요구공학 (MDRE; Market-driven Requirement Engineering)을 적용한 사례이다. 우선적으로 새로운 시장의 요구사항을 도출하고, 기존 특정 고객주도 요구공학(CDRE; Customer-driven Requirement Engineering)과 비교하여 차별점을 식별하고, 해당 부분에서 공통부분과 가변부분을 분리하여, 새로운 시장에서의 고객별 요구사항에 대한 대응이 가능하도록 하였다. 이를 통하여 기존 특정고객을 위해 개발된 소프트웨어의 재활용성을 극대화할 수 있었으며, 유지보수성도 증가할 수 있었다.

1. 서 론

시장주도 요구공학(MDRE)은 고객주도 요구공학(CDRE)의 상대되는 용어이다. 이는 소프트웨어를 개발함에 있어서, 특정 고객의 요구사항을 통해서 제품을 개발해 나가는 것이 아니라, 전체적인 시장 분석을 기반으로 요구사항을 도출하여 개발하는 방식을 일컫는다.[1] 시장분석을 기반으로 소프트웨어를 개발하기 때문에 보다 많은 고객에게 소프트웨어를 판매할 수 있고, 이렇게 개발된 소프트웨어를 패키지 소프트웨어, 혹은 상용 소프트웨어라고 부르기도 한다.

우리가 개발한 가동원전 형상관리 시스템(CMIS; Configuration Management Information System)은 하나의 고객을 위해서 개발된 소프트웨어이다. 개발 단계에서부터 고객의 요구사항을 주도적으로 반영하였으며, 하나의 고객이 사용 중에 있다. 물론, 한 명의 고객이 사용하는 것은 아니므로, 사용자에 따른 다양한 요구사항이 반영되기는 하였으나, 동일한 업무 프로세스를 보유하고 있는 한개의 기관에서만 사용되고 있기 때문에 다른 고객에게 판매하기에 쉽지 않은 형태로 유지보수가 지속되고 있다.

원자력발전소는 국가별로 상이한 형태의 원자로를 개발하여 건설하고 있는데 이를 크게 분류하면, 미국/프랑스/한국 등의 PWR, 캐나다의 CANDU, 러시아의 VVER이 가장 특징이 다른 노형이라고 할 수 있다. 이중 PWR과 CANDU는 국내에 있는 형태이고,

VVER은 국내에 없는 형태의 노형이다.

우리나라의 모든 원자력발전소는 한국수력원자력(이하 한수원)에서 운영하고 있으므로, CMIS를 활용하는 기관은 한수원 1개의 기관만 독립적으로 운영할 수밖에 없다. 그러나 다행히도 한수원은 PWR과 CANDU 등 다양한 노형을 운전하고 있으므로, 고객의 요구사항을 기반으로 개발된 CMIS는 다양한 노형에 대한 대응방안이 구현이 되어 있는 상태이다.

더이상 국내에서는 구매할 수 있는 기관이 없기 때문에 해당 소프트웨어를 판매해야 하는 상황에서, CMIS의 가장 큰 해외 시장은 VVER 노형을 가지고 있는 동구권 국가들이라고 할 수 있다. VVER은 현재 가장 많이 신규원전 건설을 하고 있는 노형이며, 아직 러시아의 가동원전 형상관리 시스템이 운영되고 있지 않아서, VVER에 대한 형상관리 시스템이 없는 상황이다. 신규건설 뿐만이 아니라 수명 연장도 진행해야 하는 상황이 도래 됨에 따라 형상관리의 중요성이 부각되고 있는 실정이다.

기존에 운영되고 있는 CMIS를 VVER 노형을 가지고 있는 동구권 국가들에게 판매하기 위한 CMIS-V² 개발이 필요하였고, 이를 위하여 시장주도 요구사항 공학의 개념을 적용하여 CMIS-V² 개발을 진행하였으며, 이에 대한 사례를 통하여 시장주도 요구사항 공학의 장점과 단점에 대해서 알아보하고자 한다.

본 논문은 다음과 같은 순서로 구성되었다. 2장에서는 시장주도 요구사항 개발에 대한 개념에 대해서

살펴보고, 3장에서는 기존의 CMIS의 고객주도 요구사항에 대해서 살펴보고, 4장에서는 VVER 시장 진입을 위한 시장주도 요구사항에 대한 반영 전략을 돌아본다. 끝으로 5장에서는 결론을 기술한다.

2. 시장주도 요구공학(MDRE) 개념

이번 장에서는 Regnell[2]이 소개하고 있는 MDRE에 대한 정의를 기술한다. 첫째, 다양한 유형의 MDRE에 대한 기본 용어를 정의하기 위해 여러가지 기본 개념을 우선 정의한다. 둘째, 고객기반 요구공학(CDRE)과 MDRE의 차이점에 대한 특징을 구별합니다. 마지막으로 MDRE의 여러 중요한 이슈에 대해 소개한다.

2.1 MDRE 기본 개념

MDRE는 소프트웨어 생산자가 많은 고객이 존재하는 공개된 시장에 제공하는 제품을, 개발하는 시장에 맞게 요구사항을 도출하고, 사양서 작성 및 검증과 같은 전통적인 요구사항 공학 활동이다. MDRE는 또한 릴리스 관리 및 시장 분석과 같은 시장 주도적 맥락에서 필요한 특정 활동을 다룬다. MDRE는 종종 다른 생산자와의 경쟁 압력 하에서 수행되며, 시장과 제품이 진화함에 따라 특정 소프트웨어 개발 조직에서 제정하는 MDRE 프로세스도 경쟁에서 앞서기 위해 진화해야 한다.

물론 소프트웨어 제품의 구매자는 구매자의 특정 요구 사항에 맞는 올바른 제품을 선택하기 위해 몇 가지 신중한 요구사항 공학 활동을 수행해야 한다. 이러한 구매자의 요구사항 공학 활동은 논외로 하더라도 생산자는 잠재적 구매자가 선택 과정에서 어떻게 생각할 수 있는지를 이해하는 것이 중요하다. 고객 우선 순위에 관한 이러한 유형의 정보는 섹션에 설명된 대로 시장 분석의 대상이 된다.

표 1. 하드웨어 및 소프트웨어 제품의 변형 예제

	하드웨어	임베디드	소프트웨어
일반	포스트-잇	핸드폰	방화벽
커스터마이징	사무실 가구	튜닝 카	ERP
고객 특정	초상화	군용차량	홈페이지

소프트웨어 제품에는 여러 가지 변형이 있다. 표 1은 (1) 맞춤화 정도와 (2) 하드웨어/소프트웨어 콘텐츠라는 두 가지 차원을 기반으로 한 소프트웨어 제품의 분류 및 몇 가지 예를 제공한다. 사용자 정의의 정도는 세 가지 수준으로 나눈다. 최종 사용자가 수행할 수 있는 사소한 구성으로 제품을 그대로 사용하려는 경우 일반 제품이라고 한다. 제품이 특정 고객의 요구에 맞게 조정된 후 유용하도록 의도된 경우 해당 제품은 커스터마이징 되었다고 한다. 개방형 애플리케이션

인터페이스를 통해 모듈을 추가함으로써. 전체 제품이 한 특정 고객의 바램을 영두에 두고 개발된 경우 해당 제품을 고객별 제품이라고 한다.

2.2 MDRE의 특징

여러 사례 연구에 따르면 MDRE는 CDRE와 여러 면에서 다르다[3]. 시장주도 개발의 주요 목표는 적시에 적절한 제품을 제공하는 것이지만 고객주도로 이루어지는 맞춤형 개발은 종종 계약 이행 및 요구 사항 사양 준수에 중점을 둔다. MDRE의 경우 매출, 시장 점유율, 상품평 등으로 성공 여부가 결정되는 반면, CDRE의 경우 고객 만족도와 사용자 수용도가 프로젝트의 실패 여부를 직접적으로 결정짓는 특징이 있다.

MDRE에서 요구 사항 도출은 종종 시장 분석과 결합된 새로운 요구 사항의 혁신에 전념하는 반면, CDRE에서의 도출은 예를 들어 알려진 사용자와의 인터뷰를 통해 한 조직이 원하는 정보를 수집하는 데 중점을 둔다. MDRE에서 릴리스될 기능 중 일부는 기밀일 수 있고 최종 사용자는 알려지지 않을 수 있으므로 추출이 정보의 주요 소스로서 고객 및 최종 사용자와의 인터뷰에 항상 의존할 수는 없다.

CDRE의 경우 협상과 갈등 해결에 많은 노력을 기울이고 있지만, MDRE 사례는 우선 순위 지정, 비용 평가 및 릴리스 계획에 더 중점을 두며 이러한 활동은 모두 개발 조직에서 수행한다[3].

CDRE의 경우 고객과 개발자 간의 접촉을 통해 지속적으로 검증이 이루어질 수 있지만, 시장 주도의 경우 검증이 개발 후반부(예: 개발 후기)까지 지연되는 경우가 많이 발생하며, 박람회 기간 중 또는 선택된 주요 고객과의 베타 테스트 기간 동안에 요구사항을 수집하거나 검증을 하기도 한다.

일반적인 MDRE의 중요한 특성은 아래와 같다.

- 1) 요구사항 개발과 관련되어 개발조직에서 모든 결정을 내리지만, 개발 중인 조직은 모든 결정을 내리지만 모든 위험을 감수한다.
- 2) 제품 수명 내내 요구 사항의 지속적인 흐름이 있다.
- 3) 요구 사항 볼륨은 잠재적으로 매우 크고 지속적으로 증가하고 있다.
- 4) 대부분의 요구 사항이 비공식적으로 설명된다.
- 5) 제품은 지속적으로 발전하고 있으며 여러 릴리스로 제공된다.
- 6) 출시 계획은 시장 출시 시간과 투자 수익에 중점을 둔다.

2.3 MDRE의 이슈사항

MDRE에 대한 설문조사를 시행한 기존의 Karlsson의

연구[4]를 참고하면 몇 가지 문제가 확인된다. Karlsson의 연구 결과는 규모와 성숙도가 다른 5개 회사의 직원과의 인터뷰를 기반으로 하는데, 이 연구의 목적은 MDRE에서 특별한 요구공학의 도전과제에 대한 인사이트를 제공하기 위한 것이다. 자세한 내용은 Karlsson의 연구[4]를 참고해야 하지만, 여기서 몇 가지 요약된 내용을 제공한다.

- 1) 시장과 기술의 균형 유지: 사용자가 필요로 하는 요구사항과 획기적인 기술을 기반으로 경쟁우위를 점할 수 있는 새로운 요구사항 간의 적절한 절충방안을 찾는 것이 필요하다.
- 2) 마케팅과 개발 사이의 틈: 마케팅과 개발자 사이에서 요구사항을 바라보는 관점의 차이가 발생할 수 있으므로, 이들 그룹 간의 더 나은 소통과 협업은 요구사항의 품질을 높일 뿐만 아니라 최종 제품의 품질을 향상시킬 수 있다.
- 3) 조직의 불안정성과 시장의 혼란: 정의된 프로세스가 없는 회사는 필요한 문서화와 구조화가 부족하기 때문에 핵심 인력이 조직을 떠날 경우 상당한 위험을 감수해야 한다. 조직이 소형화 되거나 급속히 확장되는 경우, 반복 가능한 프로세스를 수립하는 것이 매우 어려운 일이다.
- 4) 기본적인 필요를 위한 간단한 도구: 일부 기업은 기본적인 활동을 위해 간단하고 사용하기 쉬운 기술을 요구하는데, 이러한 솔루션을 찾는 것 또한 하나의 도전적인 과제이다.
- 5) 요구 사항 종속성: 요구 사항 간의 종속성은 릴리스 계획을 어렵게 만든다. 일부 회사는 관련 요구 사항을 번들로 묶어 기본적으로 종속성을 처리하지만 최소한 가장 중요한 종속성을 효율적으로 관리할 수 있는 방법이 필요하다.
- 6) 비용-가치-추정 및 출시 계획: 출시 계획은 정확한 추정에 의존한다. 비용을 과소평가하면 마감일이 초과될 수 있으며, 비용을 과대평가하면 중요한 요구 사항이 제외될 수 있다. 가치를 과대평가하거나 과소평가하면 제품이 실제 시장 요구와 맞지 않게 되어 개발 투자가 손실을 입을 수 있다.
- 7) 과부하된 요구사항 관리: 개발자와 고객의 요구 사항 제안은 필수적이다. 요구 사항 저장소가 요구 사항으로 넘쳐나는 것을 방지하고 도달 요구 사항 수가 피크일 때 처리량을 유지하는 방법은 과제다. 위에 언급된 도전 과제는 본질적으로 어려운 문제를 드러내며 단일한 간단한 솔루션으로 과제를 해결할 수 있을 것 같지 않다. 시장 주도형 기업의 핵심 문제는 경쟁사보다 앞서 나갈 수 있도록 이러한 문제를 관리하는 데 있어 지속적으로 개선하는 것이다.

3. CMIS의 고객주도 요구사항

CMIS는 2014년부터 2016년까지 정부지원사업으로 NSE와 한수원이 함께 참여하여 개발되었다. 정부지원사업으로 플랫폼의 기본 기능을 개발한 이후에 2016년 상반기부터 한수원의 요구사항을 적극적으로 수용하여 제품의 최종 기능을 구현하였다. 개발 단계의 처음부터 고객의 요구사항을 받아서 개발한 것이 아니라, 기본적인 플랫폼 설계와 구현을 마치고, 실제로 서비스를 구현하는 단계에서 고객주도 요구사항을 기반으로 개발하였다.

3.1 CMIS 도입을 위한 고객요구사항 (1차)

서비스 구현단계에서의 고객주도 요구사항 목록은 표2와 같다. 설계요건관리, 설비형상정보관리, 통합변경관리의 3가지가 주요항목이다. 이 중에서도 통합변경관리 항목이 가장 중요한 항목이며, 해당부분의 세부개발 내용이 가장 많은 비중을 차지하였다.

표 2. CMIS 고객주도 요구사항 목록 (1차)

항목	세부 기능
설계요건관리	<ul style="list-style-type: none"> • H-DBD 템플릿 및 DBD 생성 • 설계요건 조회 및 검색 • 설계요건 추적관리
설비형상정보관리	<ul style="list-style-type: none"> • 설비형상정보파일 변환 • SSC관련 형상정보 검색 • 기능위치/설비마스터 자료 조회 • OCR 프로그램 탑재 및 자료 변환 결과
통합변경관리	<ul style="list-style-type: none"> • 설계변경 관리 • 엔지니어링 문서변경 관리 • 임시변경 관리 • 소프트웨어 형상관리 관리 • 대쉬보드 (진행현황 조회 및 통계)
프로젝트 셋팅 / Workflow 셋팅	<ul style="list-style-type: none"> • 메뉴 및 사용자 권한 관리 • Workflow 생성/수정/삭제, 결재 상태 조회
문서검증 / 설계변경절차서	<ul style="list-style-type: none"> • 문서 저장정보 조회 • 설계변경절차서 조회
To-do list	<ul style="list-style-type: none"> • 개인별 to-do list 관리

통합변경관리는 발전소에서 발생하는 설계변경과 관련된 절차를 담당하는 기능으로, 이미 한수원에서 설계변경절차서에 정의된 절차대로 시스템이 동작하도록 구현하였다. 이를 구현하기 위하여 고객의 기존 설계변경절차에 대한 분석과 사용자 인터뷰를 진행하였으며, 사용자들의 기존 업무절차를 개발하려는 시스템을 활용하여 계속 진행할 수 있도록 개발하였다.

3.2 CMIS 전사확대를 위한 고객요구사항 (2차)

한수원은 총 13개의 발전소를 운영하고 있다. 개발된 시스템은 우선적으로 1개의 발전소에 시범운영을 6개월간 진행하였으며, 시스템의 효과성이 증명되어 전체 발전소로 확대 적용하기로 결정되었다.

따라서, 시스템이 한수원의 전 발전소로 확대 적용되기 위하여, 기존 시범적용 기간의 고객 피드백을 중심으로 고객의 2차 요구사항을 수령하였으며, 2차 요구사항의 핵심 내용은 다음의 표3과 같다.

표 3. CMIS 고객주도 요구사항 목록 (2차)

항목	세부 기능
통합변경관리 대쉬보드 개선	<ul style="list-style-type: none"> 본사 및 본부별 통합현황 대쉬보드 설계단계별 진행시간 및 검색 세부 조건별 통계표 추가 발전소별 설계변경 성과지표 단계별 대기목록 통계자료 지연업무 통계자료
통합변경관리 편의성 개선	<ul style="list-style-type: none"> 절차서 개정에 따른 개정양식 적용 결재라인 자동 설정 결재단계 변경시 알림 발송 검토서에 대한 관련 부서 연계 To Do List 개선 필수값 체크 및 엑셀 출력
설비형상정보 기능위치 개선	<ul style="list-style-type: none"> 기기명 검색 계통도 조회
기타	<ul style="list-style-type: none"> 전 발전소에 대한 자료 OCR 및 이관 노형별 설비형상정보 구축을 위한 커스터마이징 수행 교육 및 홍보영상 작성 이력관리 및 권한관리 기능 개선

2차 고객주도 요구사항에서는 표 3과 같이 주로 통합변경관리에 대한 개선 내용이 주로 발생하였다. 앞서 설명한바와 같이 통합변경관리는 고객의 변경관리 절차에 기반하였으며, 해당 절차에 대한 통계를 바탕으로 한 대쉬보드 기능과 자동화 기능을 통한 편의성 개선이 주된 요구사항이었다.

3.3 CMIS 기능 강화를 위한 고객요구사항 (3차)

2차 고객주도 요구사항에 의한 개발이 완료되고 한수원의 모든 발전소에서 CMIS가 사용되기 시작하면서, 사용자의 다양한 의견이 수집되었다. 약 1년정도 운영이 된 이후에, 현장 사용자들로부터 수집된 피드백을 기반으로 고객요구사항이 다시 정리되었다.

표 4. CMIS 고객주도 요구사항 목록 (3차)

항목	세부 기능
통합변경관리 기능 개선	<ul style="list-style-type: none"> 설계변경 절차 개정 반영 유효성 평가 설계변경 후속조치 CAP 발행 화재 안정성 평가 설계기술 검증위원회 양식 개선 임시변경 개선 시공정보 입력 및 계약대상 산정
통합변경관리 대쉬보드 개선	<ul style="list-style-type: none"> 설계변경종합 현황 제안서 및 변경서 통계 인허가 현황 및 자료 관리
기타	<ul style="list-style-type: none"> 본사페이지 개선 디자인 개선 업무효율성 지원 관리자 기능 개선 연계시스템 관리

3차 요구사항은 좀 더 고객에게 특화된 요구사항이 더 많이 수집되었다. 1년간의 시스템 사용에 대한 피드백을 기반으로 정리된 요구사항이기 때문에 고객의 불편사항뿐만 아니라 CMIS와 연계하여 활용할 수 있는 추가 요구사항까지 수집이 되었다.

3차 요구사항에 대한 개발이 완료되었고, 현재까지 유지보수 차원에서 몇 가지 수정사항이 발생하였지만, 다년간 지속적으로 사용하고 있는 시스템으로 발전되었다.

결과적으로 CMIS는 다음과 같은 특징을 갖고 있는 시스템으로 운용되고 있다.

- 1) 순수 국내 기술로 개발되어 5년 이상 사용되고 있음(일평균 500명이상 사용 중).
- 2) 국내 모든 가동원전 23기에 적용하여 운용되고 있음 (웨스팅하우스, 프라타툼, CANDU, 한국표준형 노형에 적용됨).
- 3) 통합변경관리 기능이 핵심기능이며, 원전 설계변경과 관련된 인허가에 필요한 모든 내용이 저장/관리되고 있음.
- 4) 지속적인 유지보수를 통하여 한수원의 업무 프로세스에 최적화되어 있음.

4. CMIS-V²의 시장주도 요구공학 적용 사례

대한민국 정부의 에너지 전환 정책으로 더 이상 신규원전 건설이 이뤄지지 않아 소프트웨어 개발업체로서 소프트웨어 신규 사업 발굴이 어려워지고, 지속적인 유지보수로 인하여 개발자 인력은 지속적으로 유지해야 하는 수익성 악화 상황에 놓이게 되었다.

그래서, 우리 원전기술의 우수성을 알리고 신규 고객 발굴이 가능하도록 해외 시장에 수출전략을 수립하였다.

미국 및 서유럽 등에 가동 중인 원전은 수명연장 시형상관리가 규제요건으로 대부분의 가동원전은 형상관리 시스템이 설치되어 운영되고 있는 상황이다. 또한, 러시아가 개발한 가압경수로¹ VVER-440, VVER-1000 노형의 원전은 형상관리 초기단계에 있으며, 러시아는 2020년 형상관리 관련 규제요건인 GOST R ISO 10007-2019, Quality management. Guidelines for Configuration Management을 발간하고, 형상관리를 요구하고 있는 시장 상황에 있었다. 러시아 VVER 원전은 설계수명이 25년, 30년임으로 비교적 짧은 기간 운영되어 형상관리를 할 필요가 없다고 생각되었지만 10년, 20년의 단위로 수명연장을 함에 따라 형상관리 필요성이 대두되고 있었다. 한국에서의 경험을 바탕으로 형상관리를 시작한 동유럽 및 러시아의 60여기의 VVER 노형의 원전을 대상으로 하는 글로벌 원전 형상관리 신서비스를 개발하여 수출을 추진하기 가장 좋은 기회를 파악하게 되었고, CDRE로 개발된 CMIS를 MDRE를 적용하여 CMIS-V²로 개발하기로 하였다.

시장주도 요구사항기반 개발 절차는 그림1과 같이 표준화 측면과 요구사항 측면으로 나누어 고려해 볼 수 있다. 표준은 해당 분야의 표준으로 제정된 것과 업계 표준으로 통용되는 것을 모두 고려하여 적합성을 검토하고, 시장 요구사항은 주요 고객들로 구성된 위원회를 통하여 요구사항을 도출한다.

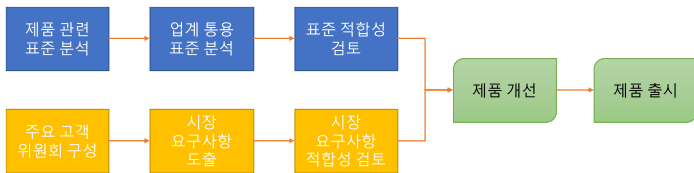


그림 1. 시장주도 요구사항기반 개발 절차

4.1 표준 적합성

고객주도 요구사항으로 개발된 CMIS를 새로운 시장에 적용하기에 앞서, 시장에서 통용되고 있는 표준에 대한 사전 조사가 필요하다. 형상관리와 관련된 표준은 다음 표 5와 같다.

표 5. CMIS관련 국제 표준 문서

문서번호	문서 제목
ANSI/NIRMA CM 1.0	“Configuration Management for Nuclear Facilities”, 2007
IAEA Safety Report 65	“Application of Configuration Management in Nuclear Power

¹ VVER(Vodo-Vodyanoi Energetichesky Reactor)혹은 러시아형 가압수형 원자로로 러시아에서 개발한 원자로로 아르메니아, 불가리아, 중국, 체코, 핀란드, 동독, 헝가리, 인도, 이란, 슬로바키아, 우크라이나, 러시아 연방 등지에서 사용 중임.

	Plants” December 2010
IAEA TECDOC-1651	“Information Technology for Nuclear Plant Configuration Management” Dec. 2010
INPO AP-929	“Configuration Management Process Description”, Rev.1, June 2005
NEI 97-04	“Design Basis Program Guidelines”, Rev.1, Feb. 2001
EPRI 3002003126	“The Progressive Benefits of moving to a Data Centric Configuration Management System for Operating Plants and New Builds”

CMIS는 정부과제를 통하여 플랫폼을 개발한 이후에 고객 요구사항을 기반으로 서비스를 기획하였기 때문에, 플랫폼 설계 시에 국제표준을 참고하여 많은 부분의 설계가 이뤄졌기 때문에 큰 차이점은 발생하지 않았다. 그러나, 통합변경관리에 대한 고객 요구사항을 기반으로 한 수정이 많이 발생하였으므로, 통합변경관리와 관련된 INPO AP-929와의 적합성 분석이 가장 중요한 항목이었다.

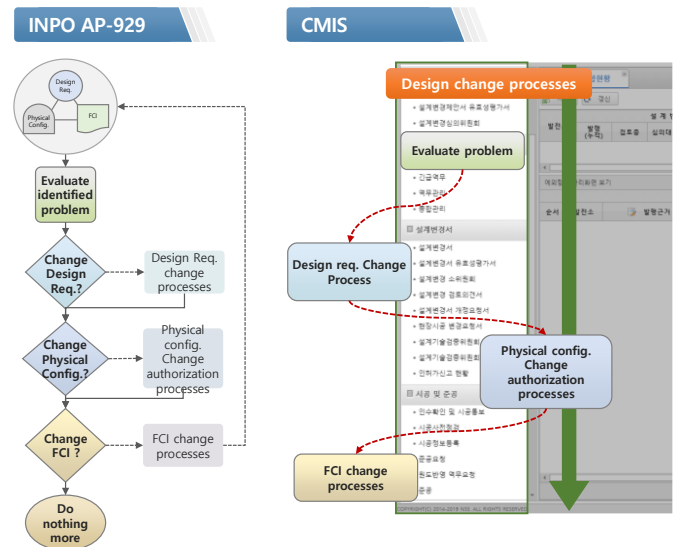


그림 2. INPO AP-929와 CMIS 적합성 분석

그림1과 같이 INPO AP-929의 적합성을 분석한 결과, 한수원의 변경절차가 처음 만들어질 때부터 AP-929를 참고하여 만들어진 절차였기 때문에 적합성이 일치하는 것을 알 수 있게 되었다.

4.2 신규 시장 요구사항 분석

전 세계적으로 449개 호기의 원전이 운영 중으로 세계 원전시장은 향후 20여년간 약 1,000달러 규모로 예상된다. 이 중 러시아, 유럽, 아시아 등 14개

국가에서 건설 및 운영 중인 러시아형 가압경수로 (VVER)는 80여기로 이들을 운영하고 있는 운영사들이 CMIS 시스템의 잠재 고객이 될 수 있지만 신규시장의 요구사항을 분석하는 업무는 매우 어렵다. 만일 그 시장이 우리가 잘 알지 못하는 시장이라면 더욱 어려울 수밖에 없다. 우리는 현재 한국 시장을 경험하면서 살고 있으나, 새로운 VVER 시장을 목표로 신규 시장 요구사항을 분석하기 위해서는 해당 시장을 잘 알고 있는 전문가를 통하여 시장의 요구사항을 수립해야 한다.

우리는 다행히 신규시장 전문가와 협업할 수 있는 기회를 찾았고, 브라질, 루마니아, 슬로바키아, 파키스탄 등과 같은 국가의 원전운영 주체와 업무협약, 인터뷰 및 자문을 통하여 신규 시장의 요구사항을 표5와 같이 도출하였으며, CMIS와의 적합성에 대한 분석을 수행하였다. 단, 표5의 신규 시장의 요구사항은 상세히 공개하기 어려운 내용이기 때문에, 본 논문에서는 키워드를 위주로 비교한 표를 공개한다.

표 5. CMIS관련 시장 요구사항 대비 적합성 분석

Req. for VVER Market	CMIS 적합성
CM for construction and operation	만족
Cover entire life	만족
Responsibility and Procedures	만족
Small sub-assemblies	만족
SSC-based hierarchical DBD	만족
V&V Documentation	만족
Baseline management	만족
Pre-determined procedure	만족
Single source of truth	만족
Provide for all related-organization	만족
Equivalent change	불만족
CM Planning	불만족
Version management	만족
Code management	만족

총 14개의 상위레벨 요구사항을 수집하였고, 해당 요구사항과 CMIS를 비교한 결과 2개의 요구사항이 만족하지 못함을 발견하게 되었다. 해당 요구사항에 대한 추가적인 구현을 고민하였으며, 기존 시스템에 2개의 기능을 쉽게 추가적으로 구현할 수 있었다.

5. 결론

본 논문에서는 CDRE로 개발된 CMIS를 MDRE를 통하여 신규 시장에 적합한 소프트웨어로 상품화하는 과정에 대한 사례를 보여주었다.

우선 CMIS는 CDRE로 개발될 때부터, 표준을 검토하여 프레임워크를 설계한 것이 큰 도움이 되었다. 즉, 서비스 영역에서는 고객이 원하는 요구사항을 적극적으로 반영하여 고객 만족을 극대화하더라도, 시스템을 설계하고, 기본기능을 구축할 때는 표준과의 적합성을 고려하여 설계 및 기본기능 개발을 진행해야 한다. 그렇게 하였을 때, 새로운 시장으로 진출한다고 하더라도 MDRE를 하여, 시장에 대한 요구사항 적용을 다시 검토할 때, 큰 차이점이 없음을 발견할 수 있게 될 것임을 확신하게 되었다.

6. 사사

본 연구는 과학기술정보통신부가 주관하는 2021년 연구산업혁신성장지원-글로벌신서비스개발 사업의 수출선도형 원전 형상관리 시스템(SILKROAD CMIS-V²) 개발 과제(과제번호: 2021글로벌-005) 지원으로 수행되었다.

7. 참고문헌

[1] Carmel, E. and Sawyer, S. "Packaged Software Teams: What makes them Different?", Information Technology & People, 11(1), 6-17, 1998

[2] Regnell B., Brinkkemper S. "Market-Driven Requirements Engineering for Software Products." In: Aurum A., Wohlin C. (eds) Engineering and Managing Software Requirements. pp 287-308, Springer, Berlin, Heidelberg, 2005

[3] Carlshamre P, "A usability perspective on requirements engineering - From methodology to product development." Dissertation No. 726, Linköping University, Sweden, 2002

[4] Karlsson L, Dahlstedt ÅG, Natt och Dag J, Regnell B, Persson A, "Challenges in market-driven requirements engineering - An industrial interview study," Proceedings of 8th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02), September 09-10th, Essen, Germany, pp.37~49, 2002



그림 3. 신규시장 원전 운영주체 업무협약 및 인터뷰

반도체 공정 유지 보수 비용 절감을 위한 공정 레시피 데이터 학습 및 장비 결함 검출

최민호^{1,2}, 유동연³, 이정원³

(주)피에스케이¹, 아주대학교 IT 융합대학원², 아주대학교 전자공학과³

e-mail: minfirst@psk-inc.com, dongso125@ajou.ac.kr,

jungwony@ajou.ac.kr

Process Recipe Data Learning and Equipment Fault Detection for Maintenance Cost Reduction in Semiconductor Process

Min-Ho Choi^{1,2}, Dong-Yeon Yoo³, Jung-Won Lee³

PSK¹, Graduate School of IT Convergence, Ajou University²

Department of Electronic and Computer Engineering, Ajou University³

요 약

본 논문은 반도체 제조 설비로부터 나오는 데이터를 이용하여 이상성 검출 모델을 생성하고, 공정이 진행되기 전에 이상성 데이터를 사전 검출하여 유지 보수 비용을 줄이는 방안을 제안한다. 반도체 공정 기술이 고도화되면서, 이전보다 센서 데이터의 종류가 많아지고 짧은 주기로 빠르게 생성되어 체계적인 데이터 수집과 분석이 필요한 실정이다. 뿐만 아니라 반도체 설비 엔지니어가 수많은 센서 알람과 복잡한 제어 로직으로부터 설비 결함의 근본 원인을 빠르게 찾아내거나, RUL(Remaining Useful Life)을 예측하는 등 유지 보수 비용을 줄이는 요구 사항이 커지고 있다. 본 연구에서는 빠르고 안정적으로 반도체 제조 설비로부터 수집된 데이터를 이용하여 CNN(Convolution Neural Network) 학습을 진행하였다. 공정 진행 중 품질에 영향을 주는 플라즈마를 생성하기 전 단계인 공정 레시피 내의 안정화 단계의 데이터를 CNN 모델로 학습하여, 이상성 데이터의 탐지하고 동시에 결함의 원인을 제시한다. 반도체 설비로부터 획득한 실제 데이터를 가지고 실험한 결과, 정확도 99.9%, 정밀도 99.6%, 재현도 99.7%, 그리고 분류 클래스 간 데이터의 불균형으로 인한 성능을 나타내는 F1 score는 99.7%로 높은 성능을 보였다.

1. 서 론¹

현재 반도체 산업은 제조 공정의 미세화, 복잡화 등이 진행되고 있다. 이와 더불어 제조 원가를 낮추고 생산성을 향상을 위해 웨이퍼의 크기가 10년 주기로 1.5배씩 커지고 있다. 전 세계의 반도체 제조업들은 웨이퍼에 들어가는 칩의 크기 축소를 위해 웨이퍼 내 회로의 선 폭을 점점 더 좁아지게 하고 있다. 반도체 공정 기술의 고도화로 웨이퍼 품질에 영향을 주는 반도체 공정상의 결함이 이전보다 훨씬 많아지게 되었다.

반도체 설비 제조 업체에서는 반도체 공정 고도화로 인한 공정 품질 결함을 방지하기 위해 공정이 진행되는 챔버(Chamber) 내에 많은 센서를 설치하고 데이터를 수집하게 된다. 수집된 데이터는 챔버 내에 다양한 센서들(Variety)로부터 많은 양의 데이터(Volume)를 빠른 주기(Velocity)로 처리하기 때문에 빅데이터의 특성을 가지게 된다. 3V의 특성을 가지는 설비 데이터

사이에서 결함이 발생하게 되면 경험이 많은 반도체 설비 엔지니어도 빠른 시간안에 이상성 데이터를 정확히 찾아내는 일은 어렵다. 또한, 엔지니어가 결함을 찾아 부품을 교체, 조정 또는 수리를 하는 등의 유지 보수를 하는 시간이 길어지게 된다. 반도체 기술이 고도화될수록 유지 보수로 인한 경제적 손실은 이전보다 더 발생하게 된다[1]. 결함으로 인한 유지 보수 비용을 줄이기 위해서는 네 가지 요소가 필요하다.

첫째, 공정 진행 시 웨이퍼 품질에 영향을 미치는 부분을 감지할 수 있는 센서 데이터가 필요하다. 둘째, 챔버 내부의 센서와 제어 기기의 정상/비정상 동작 시 발생하는 데이터의 특징을 파악하고 감지하는 모델이 필요하다. 셋째, 센서들과 제어 기기의 동작 특성을 정교하게 파악하기 위해서는 데이터를 짧은 주기로 수집하고 학습에 적합하도록 전처리하는 과정이 필요하다. 넷째, 검출된 이상성 데이터를 가지고 반도체 설비 엔지니어가 어떠한 조치를 하였는지 확인하고 결함에 따른 유지 보수 종류를 분류해야 한다.

반도체 산업계에서는 사후 유지 보수와 예방 유지 보수 방식을 사용하여 운용하였으나, 반도체 제조 설비의 가용성과 공정의 안정성이라는 높은 요구

¹ 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2020R1A2C1007400).

사항에 완벽하게 대처할 수 없었다. 그래서 설비의 센서 데이터를 기반으로 고장이 발생하기 전에 시행했던 불필요한 예방 보전 활동을 최소화하고 유지 보수 간격을 최적화하기 위해 잔여 수명(RUL, Remaining useful life) 모델링을 통해 이를 예측하였다. 예측 결과에 따라 정상적인 설비의 운용이 지속될 수 있도록 데이터를 기반으로 수행하는 CBM(Condition-Based Maintenance) 전략 방식으로 산업 현장에서 변화가 되고 있으며 관련 연구가 진행되고 있다[2-7].

[8]에서는 공정 중 발생하는 물질들이 챔버 벽면에 침전되어 누적된 물질을 PLA(Piecewise Linear Approximation)를 사용해 가상 계측 방법을 제안하여 챔버의 잔여 수명을 예측하는 것이 연구되었다.

본 논문에서 제안 방법은 다음과 같다.

우선, CNN 학습에 입력값으로 결함 검출의 정확도를 높이기 위해 공정 품질에 영향성이 있는 데이터를 선별하고 수집한다. 공정 품질 결함을 사전에 검출하기 위해 수집한 데이터 중 웨이퍼를 식각하기 전의 데이터만을 추출한다. 공정을 진행하면서 센서들마다 서로 다른 데이터의 특성을 보이기 때문에 공정 결과 데이터나 작업자의 유지 보수 보고서를 기준으로 정상 데이터와 이상성 데이터의 특성을 파악하고 분류한다. 이러한 방법으로 분류된 데이터셋을 가지고 CNN을 이용한 학습 및 검증을 통해 이상성 데이터 검출 모델을 구축하였다. 모델에 대한 평가는 반도체 설비로부터 데이터를 수집하여 모델 평가와 유효성 검증을 진행하였다. 정상 데이터와 143종류의 결함에 대해 센서 데이터를 수집하고 분석하여 데이터셋을 구성하였다. CNN 학습 결과 정확도(Accuracy) 99.9%, 정밀도(Precision) 99.6%, 재현도(Recall) 99.7%, 그리고 F1-Score는 99.7%를 보였다.

2. 학습을 위한 공정 제어 레시피 데이터 특징 추출

일반적으로 반도체 제조 설비에서는 공정을 진행하기 위해서는 챔버 내부에 있는 제어 기기들을 이용하여 어떠한 순서로 제어할 것인지를 기술한 공정 제어 레시피가 있다. 공정 제어 레시피 내의 센서 데이터에서 학습 데이터를 추출하기 위해 센서 데이터 중 플라즈마를 형성하기 위한 주요 요소인 가스 유량, 챔버의 압력을 기준으로 센서 데이터를 선정하였으며, 추가로 웨이퍼 베벨 식각시 식각율(Etch rate)에 영향을 주는 웨이퍼와 챔버 천장 간의 거리를 포함하여 총 6개의 센서 데이터를 선정하였다. 아래는 선정된 6개의 센서 데이터에 대한 설명이다.

- O₂, SF₆, CF₄ MFC(Mass Flow Control): 챔버 내부에 유입되는 가스의 유량 값
- 압력 제어 밸브(Pendulum Valve): 챔버의 내부 압력을 제어하기 위한 밸브의 위치 값
- Chuck vacuum pressure: 척 위에 있는 웨이퍼를

진공으로 고정하기 위한 압력 값

- Gap sensor: 웨이퍼와 챔버 천장과의 거리 값

웨이퍼가 챔버 내에서 공정을 진행하면서 공정 제어 레시피에 기술되어 있는 스텝 순서에 따라 제어 기기들의 제어를 수행한다. 공정 제어 레시피는 기능적으로 아래와 같이 크게 3단계로 나누어진다.

- 1) Stabilizing step: 웨이퍼를 식각하기 위해 플라즈마의 생성을 위한 환경 조건을 안정하게 하는 단계
- 2) Etching step: 플라즈마를 이용하여 웨이퍼 위의 필름을 식각하는 단계
- 3) Cleaning step: 식각을 완료 후 웨이퍼의 이동 환경 조성 및 부산물 제거 등의 공정을 마무리하는 단계

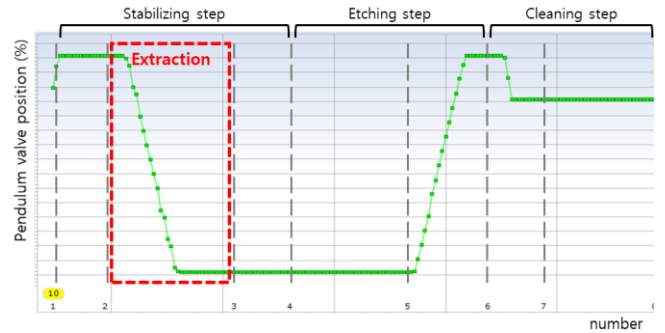


그림 1. 압력 제어 밸브 위치 센서 데이터별 스텝 구간 및 추출 영역

공정 제어 레시피에 의해 웨이퍼가 챔버에 들어가서 공정을 시작하고 끝날 때까지의 전체 데이터를 그림 1처럼 3단계(Stabilizing → Etching → Cleaning)로 구분한다. 그리고 공정비용 손실을 줄이기 위해, 공정 품질에 영향을 주는 식각(Etching) 단계에서 플라즈마의 생성을 위한 환경 조건을 만들어 주는 안정화 단계 중, 센서 데이터의 변동이 크게 일어나는 구간만을 추출한다. 그림 1의 경우 선정된 센서 데이터 중 압력 제어 밸브 위치 데이터를 스텝별로 구분한 전체 그래프이다. 이 중 Stabilizing step이며 데이터의 변동이 크게 일어나는 스텝 번호 2번의 데이터를 추출하였다.

3. 설비 데이터 이상성 사전 검출을 위한 학습

3.1 데이터 전처리

CNN으로 학습을 잘 이루어질 수 있도록 추출된 안정화 스텝 구간의 데이터를 정규화된 데이터로 변환하는 전처리단계를 거쳤다.

아래 그림 2에서 보는 바와 같이 각 센서 데이터의 값들이 표현을 할 수 있는 범위가 서로 다르므로 이를 그대로 적용하면 값의 크기가 절대적인 가중치로 작용하여 학습 성능이 떨어질 수 있다. 이를 해결하기 위해서 최솟값 0, 최댓값 1로 하여 값의 범위를 0부터

1까지로 표현하도록 최소-최대 정규화(min-max normalization)를 사용하였다.

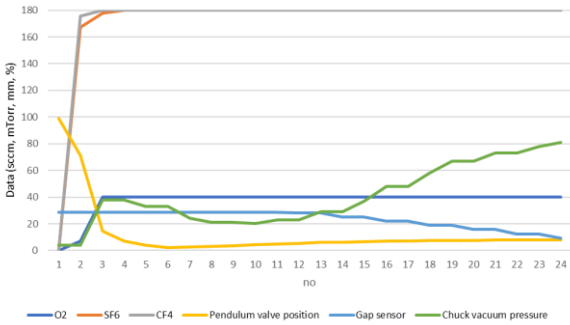


그림 2. 정규화 이전의 센서 데이터값

3.2 CNN 학습 모델 구축

CNN으로 학습하기 위해 베벨 식각 설비에서 진행한 센서 데이터를 입력 데이터로 사용하여, 정상 데이터 대비 결함을 가지는 이상성 데이터 비율이 낮았기 때문에, 정상 데이터에 편향된 학습이 되어 모델의 정확도가 낮아진다.

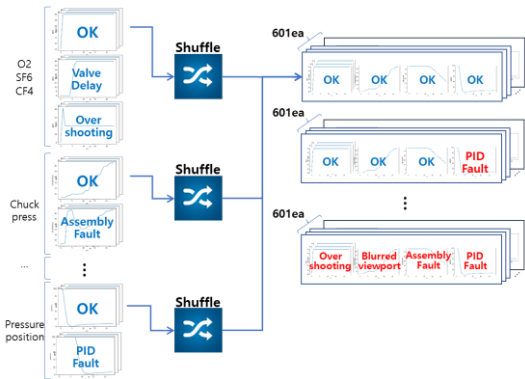


그림 3. 균형적인 데이터셋 생성 방법

이러한 문제점을 해결하기 위해서 다음과 같은 방법을 사용하였다. 각 센서 데이터별로 정상 데이터와 이상성 데이터로 분류된 데이터 중 무작위로 하나의 데이터를 선택하여 데이터셋을 만드는 방식을 사용하였다. 예를 들어 그림 3에서와 같이 센서들이 모두 정상인 데이터를 만들기 위해서는 각 센서 데이터별로 정상 데이터를 무작위로 하나씩 선별해서 조합하여 만든다.

한 개의 정상 데이터와 143개의 결함별로 분류된 이상성 데이터를 학습 모델의 성능을 고려하여 각각 601개씩 생성한다. 생성된 데이터는 총 86,544개로 144개의 결함 원인이나 정상 여부를 분류할 수 있게 되었다. 이에 따라 정상 데이터와 이상성 데이터 비율에 균형이 잡힌 데이터를 다량으로 생성하여 학습의 정확도를 높일 수 있었다.

전체 데이터 86,544개 중 훈련 데이터 69,235개를 가지고 훈련하였다. CNN 구조 설계는 3차원 합성곱 신경망 (3 dimension convolution neural network)을 사용하였다. 각 센서 데이터를 재배치시켜 3차원 구조로 만들었다. 3차원 구조의 가로축(depth)은 시계열, 세로축(height)은 센서 데이터가 되고 폭(width)은 센서 데이터 개수가 된다. 그림 4는 CNN 레이어 구조이다.

표 1. 3차원 CNN 학습 과정

Layer (type)	Output Shape	Param
Conv2D	(None, 1, 24, 64)	1216
MaxPooling2D	(None, 1, 12, 64)	0
Conv2D	(None, 1, 12, 128)	24704
MaxPooling2D	(None, 1, 6, 128)	0
Dropout	(None, 1, 6, 128)	0
Flatten	(None, 768)	0
Dense_1	(None, 512)	393728
Dense_2	(None, 256)	131328
Dense_3	(None, 144)	37008

표 1은 CNN 학습 과정을 보여주고 있다. 입력 데이터로 1*24*6 크기의 3차원 데이터가 입력되고 입력 데이터의 특징점을 추출하기 위해서 커널 사이즈 64, 스트라이드(Stride) 1, 그리고 세임 패딩(same padding)을 사용하는 3차원 CNN 학습 진행하였다. 이후 특징점의 크기를 줄이기 위해 맥스 풀링을 사용하였다. 이에 따라 3차원 데이터가 64개의 특징점을 가지고 1*12 크기를 가지는 2차원 데이터로 변경이 되었다. 그리고 한 번 더 커널 사이즈가 128인 2차원 CNN 학습과 맥스 풀링을 진행하였다. 결함을 발견하고 분류하기 위해서 3번의 완전 연결 층을 사용하였다. 그리고 훈련 데이터를 이용하여 학습 시 계산 비용과 시간을 절약하기 위해 훈련 데이터의 검증

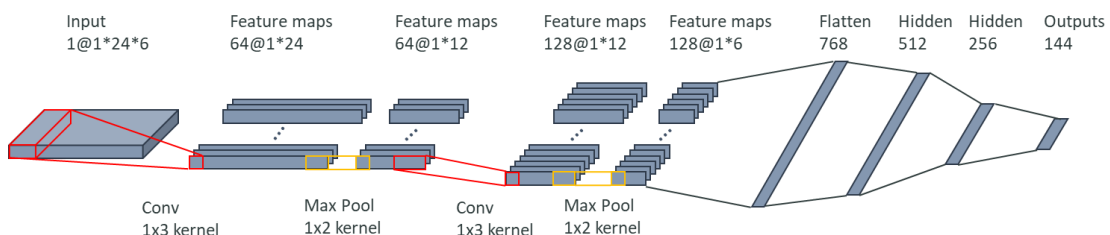


그림 4. CNN 레이어 구조

점수가 더 이상 감소하지 않고 상승하여 과대 적합이 일어난다고 판단하면 그 훈련을 계속 진행하지 않고 멈추는 조기 종료(Early stopping) 기법을 사용하였다.

4. 평가

실제 반도체 제조 설비인 베벨 식각 설비의 챔버에서 공정을 진행한 센서 데이터를 이용하여, 학습한 모델이 실효성이 있는지 확인하기 위해 정상 데이터와 이상성 데이터의 검출과 분류를 통하여 검증한다. 수집된 데이터를 기반으로 생성한 143개의 결함 클래스로 학습한 모델을 실제로 발생한 결함 클래스 24개로 검증을 진행하였다.

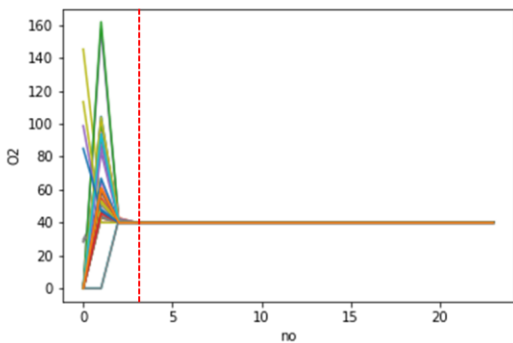


그림 5. O₂ 가스 유량 이상성 데이터 중 Overshoot 데이터 누적 그래프

실제 데이터로 인해 결함 클래스 143개 중 24개를 제외한 119개의 결함 클래스를 검증할 수 없어, 센서 데이터별로 정상 데이터와 이상성 데이터의 검출과 분류를 통해 검증한다. 그림 5는 O₂ 가스 유량 데이터 중 Overshoot 이상성 데이터로 분류된 데이터를 누적한 그래프이다.

O₂ 가스 유량 데이터는 총 1,001개의 데이터 중 OK 914개, 제어 Valve delay 42개, 그리고 Overshoot 45개로 분류되었다. 검출 결과 정상과 이상성 데이터로 분류된 데이터는 모두 정상 판정되었다. CF₄ 가스 유량 데이터는 총 1,001개의 데이터 중 OK 961개와 제어 Valve delay 40개로 분류되었다. 검출 결과 정상과 이상성 데이터로 분류된 데이터는 모두 정상 판정되었다. SF₆ 가스 유량 데이터는 총 1,001개의 데이터 중 OK 963개와 제어 Valve delay 38개로 분류된 데이터를 판정한 결과이다. 실제 정상 데이터 963개 중 정상 데이터로 판단한 데이터 수는 962개이고 1개가 제어 Valve delay로 잘못 판정하였다.

압력 제어 밸브 위치 데이터는 총 1,001개의 데이터 중 OK 962개와 모터의 PID fault 39개로 분류되었다. 검출 결과 정상과 이상성 데이터로 분류된 데이터는 모두 정상 판정되었다.

웨이퍼 고정 압력 데이터는 총 1,001개의 데이터 중 OK 939개와 Assembly fault 62개로 분류되었다. 실제

정상 데이터 939개 중 정상 데이터로 판단한 데이터 수는 938개이고 1개가 Assembly fault 이상성 데이터로 잘못 판정되었다. 그리고 실제 Assembly fault 62개 중 61개가 이상성 데이터로 판정이 되었고 1개가 정상 데이터로 잘못 판정되었다.

표 2. 센서 데이터별 정확도, 정밀도, 재현도, F1 score 결과

	Accuracy	Precision	Recall	F1 score
O ₂ flow rate	100.0%	100.0%	100.0%	100.0%
SF ₆ flow rate	99.9%	98.7%	99.9%	99.3%
CF ₄ flow rate	100.0%	100.0%	100.0%	100.0%
Pendulum control valve position	100.0%	100.0%	100.0%	100.0%
Chuck vacuum pressure	99.8%	99.1%	99.1%	99.1%
Gap sensor	99.9%	99.9%	99.3%	99.6%

웨이퍼와 챔버 상단 간 거리의 데이터는 총 1,001개의 데이터 중 OK 937개와 공정 recipe fault 12개, 그리고 Blurred viewport 52개로 분류되었다. Blurred viewport 이상성 데이터의 경우 실제 데이터 52개 중 Blurred viewport 이상성 데이터라고 판단한 개수는 51개이며 그중 1개는 정상 데이터로 잘못 판정하였다.

표 2는 센서 데이터별 정상 데이터 및 이상성 데이터 검출 정확도, 정밀도, 재현도 그리고 F1 score를 보여주고 있다. 모두 99% 이상 높은 수치로 산출되었다.

5. 결론

본 논문은 공정을 진행 중 센서들의 결함으로 인한 공정비용 손실이 발생하기 전의 데이터를 사용하여 학습시키고, 학습된 모델을 통해 정상 데이터와 이상성 데이터를 사전에 검출하는 방법을 제안하여 높은 성능을 보였다. 높은 성능의 주요 원인은 다량의 균형적인 데이터셋 생성과 공정 결과 및 유지 보수 보고서를 기준으로 정상과 이상성 데이터 분류로 분석된다.

본 연구에서 제안한 내용을 이용하면 웨이퍼 식각 단계로 넘어가기 전에 결함 여부를 알 수가 있어 공정비용 손실이 감소한다. 또한 이상성 데이터 분류 시 결함의 근본 원인별로 분류를 하였기 때문에 반도체 설비 엔지니어가 반도체 제조 설비를 유지 보수하는 시간이 줄어들게 된다. 이러한 유지 보수 시간을 효율적으로 줄이기 위해서 반도체 설비 엔지니어는 주기적으로 이상성 데이터별 조치 사항에 대해 분류할 수 있도록 학습 데이터셋을 업데이트할 필요가 있다.

참 고 문 헌

- [1] 박유진, 박영수, 허선, 이현, “차세대 반도체 공정을 위한 실시간 수율관리 시스템 아키텍처 구축에 대한 연구,” 한국산학기술학회, pp.555-558, May 2010.
- [2] Jay Lee, David Siegel and Edzel Racsa Lapira, “Development of a predictive maintenance demonstration system for a semiconductor etching tool,” ECS transactions, vol. 52, no.1, pp.913-927, Mar. 2013.
- [3] Jimmy Iskandar, James Moyne, Kommisetti Subrahmanyam, Parris Hawkins, and Mike Armacost, “Predictive maintenance in semiconductor manufacturing,” in 2015 26th Annual SEMI Advanced Semiconductor Manufacturing conference (ASMC), May 2015.
- [4] Anahid Naghibzadeh-Jalali, Clemens Heistracher, Alexander Schindler, Bernhard Haslhofer, Tanja Nemeth, Robert Glawar, Wilfried Sihn, and Peter De Boer, “Predicting time-to-failure of plasma etching equipment using machine learning,” in 2019 IEEE International Conference on Prognostics and Health Management (ICPHM), Jun. 2019.
- [5] Pin Li, Xiaodong Jia, Masahiro Sumiya, Yoshito Kamaji, Masaki Ishiguro, Laura Pahren, and Jay Lee, “A novel method for deposit accumulation assessment in dry etching chamber,” IEEE Transactions on Semiconductor Manufacturing, vol. 32, May 2019.
- [6] Shota Umeda, Kenji Tamaki, Masahiro Sumiya, and Yoshito Kamaji, “Planned Maintenance Schedule Update Method for Predictive Maintenance of Semiconductor Plasma Etcher,” IEEE Transactions on Semiconductor Manufacturing, vol. 34, pp.296-300, Aug. 2021.
- [7] Sathyan Munirathinam, and Balakrishnan Ramadoss, “Big data predictive analytics for proactive semiconductor equipment maintenance,” 2014 IEEE International Conference on Big Data (Big Data), pp.893-902, Oct. 2014.
- [8] Pin Li, Xiaodong Jia, Masahiro Sumiya, Yoshito Kamaji, Masaki Ishiguro, Laura Pahren, and Jay Lee, “A Novel Method for Deposit Accumulation Assessment in Dry Etching Chamber,” IEEE Transactions on Semiconductor Manufacturing, vol. 32, pp.183-189, May 2019.

개인정보보호를 위한 SGX 기반 학습데이터 비식별화 인공지능 학습 프레임워크

김남령[○], 옥지원[○], 김성민

성신여자대학교 융합보안공학과

{20190885, 20190924, sm.kim}@sungshin.ac.kr

Secure de-identification for data privacy with SGX-based Artificial intelligence learning framework

Namryeong Kim[○], Jiwon Ock[○], Seongmin Kim

Department of Convergence Security Engineering, Sungshin Women's University

요 약

인공지능 기술의 발전에 따라 다양한 서비스 분야에서 빅데이터 기반의 학습데이터가 이용되고 있다. 관련 서비스들은 방대한 양의 학습데이터 저장 및 관리를 위해 필요한 컴퓨팅 자원 및 대용량 스토리지를 클라우드에 의존함으로써, 데이터 이용에 대한 효율성을 높였다. 그러나 클라우드는 인공지능 학습 알고리즘에 대한 보안과 저장 및 관리 환경에 대한 신뢰를 보장할 수 없고, 학습데이터 유출로 인한 지속적인 프라이버시 문제가 제기되고 있다. 본 논문에서는 신뢰 실행 환경을 제공해주는 보안 프로세서 아키텍처인 SGX를 인공지능 서비스를 위한 클라우드 플랫폼에 적용함으로써, 안전한 인공지능 학습 방법을 제시한다. 개인정보가 포함된 학습데이터를 분리하고 비식별 조치 및 저장할 수 있는 방안과 더불어, 기존 블록체인 및 암호화를 기반으로 한 기존 기술 대비 실효성을 갖추면서도 학습 과정 및 데이터의 안전성 또한 보장하는 프레임워크를 제안한다.

1. 서 론

인공지능(Artificial Intelligence, AI)은 인간의 지능을 인공적으로 구현한 컴퓨터 시스템으로, 모바일, PC, IoT(Internet of Things), 스마트팩토리, 헬스케어 등 사회 전반을 걸쳐 활발하게 사용된다. 인공지능 서비스는 대량의 학습데이터를 기반으로 예측 및 제공되는데, 학습데이터에 대한 저장과 관리에 어려움이 있다. 이는 시·공간에 구애받지 않고 대용량 데이터 스토리지를 제공하는 클라우드로 해결할 수 있고, 현재 다양한 클라우드 환경 기반의 인공지능 서비스가 등장하여 활발히 이용되고 있다[1].

하지만 인공지능의 학습데이터 일부는 민감정보를 비롯한 개인정보를 포함하고 있어 수집 및 이용, 저장, 관리, 파기 전 과정에 걸쳐 보호 대책이 필요하다. 그러나, 투명성을 강조하는 프라이버시와는 달리 클라우드는 사용자가 내부 구조를 확인할 수 없는 불투명성을 기반으로 하고 있어 개념적 상충이 발생한다. 이러한 부조화는 클라우드 내부에 저장된 개인정보 유출로 이어졌고, 이를 법적으로 제재하자는 목소리가 커졌다. 따라서, 클라우드에서 활용되는 개인정보를 포함한 학습데이터들을 보호하기 위한 기술 확보가 시급해졌다.

현재 클라우드에 저장되는 데이터들을 안전하게

보관하고자 데이터 분리나 암호화, 블록체인을 도입하며 보안성을 제고한다 [2, 3, 4]. 이러한 방법은 종단(end-host)의 접근을 통제하여 데이터의 보안성을 높일 수 있지만, 클라우드를 관리하는 플랫폼 제공자 등의 상위 특권을 갖는 계층으로부터의 접근을 통제할 수 없다. 때문에, 악의적인 상위 특권 계층이 데이터를 도·감청하여 유출할 수 있는 문제가 발생할 수 있고, 이는 곧 클라우드 플랫폼 제공자에 대한 신뢰도 저하로 이어진다. 또한, 개인정보를 포함한 민감한 데이터가 유출될 경우 막대한 피해를 초래할 수 있어 민감 데이터를 안전하게 저장 및 관리하면서 성능적으로 실효성이 있는 시스템에 대한 재고가 필요하다.

이에, 본 논문에서는 SGX 기술을 이용하여 개인정보가 포함된 인공지능 학습데이터의 처리 및 학습을 비롯해 클라우드에 저장된 데이터의 안전성 또한 보장하는 방안을 제안한다. 데이터 제공자로부터 민감한 개인정보와 일반데이터에 대한 명세를 구분하여 받은 뒤, 개인정보에 대해서는 SGX로 보호되는 영역 내에서 비식별화 조치를 수행한다. SGX의 또 다른 기능 중 하나인 sealing 및 unsealing을 통해 학습 시에만 데이터를 사용하고 미사용 시 암호화하여 클라우드 내 스토리지에 저장함으로써 클라우드 플랫폼이 근본적으로 갖는 불투명성에도 학습데이터를 보호할 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 개인정보가 포함된 인공지능 학습데이터의 이용 및 보관 현황과 관련 연구를 분석한다. 3장에서는 SGX 기술 관련 배경지식을 설명하고, 4장에서는 인공지능 학습 프레임워크에 대해 제안하며 5장에서 정성적 평가를 기술한다.

2. 관련 연구 및 기술 현황

2.1. 개인정보가 포함된 인공지능 학습 데이터의 이용 및 보관 현황

인공지능에 활용되는 학습데이터를 보호하기 위해 국내외를 불문하고 법적, 제도적 장치들을 마련하는 것은 하나의 큰 흐름이 되었다. 유럽의 경우 인공지능 데이터 및 개인정보보호를 위한 가이드라인을 발간하여 인공지능의 프라이버시 이슈를 극복하고자 하였다[1]. 국내는 개인정보위원회를 중심으로 인공지능 학습 데이터로 이용되는 개인정보를 비식별 조치 또는 암호화를 통해 보호하고 있다[5].

비식별이란 이름, 전화번호, 주소 등 개인을 식별할 수 있는 정보가 노출되어도 식별할 수 없도록 조치하는 것을 말한다. 예를 들면, 영상이나 이미지의 경우 필터링, 일부분의 암호화, 얼굴 합성 등을 이용해 개인정보를 보호할 수 있다[6]. 국내에서는 정부에서 운영하는 'AI 허브'가 대표적인 예시로 볼 수 있는데, 전 국민을 대상으로 학습용 데이터를 제공하는 서비스로 제공 대상 범위가 넓은 만큼 4단계의 검수 체계를 거쳐 비식별화 여부를 점검하고 있다[7].

클라우드에 저장되는 데이터들을 법적으로 암호화하여 관리하고 인공지능 학습데이터에 대한 저장 및 보관 관련 보호 대책을 수립하고 있지만, 기술적, 관리적 한계로 개인정보 유출 사례는 끊임없이 발생하고 있다. 지난 2019년 5월부터 개방한 AI 허브는 무려 21개월간 비식별화되지 않은 개인정보를 전 국민이 열람할 수 있었다. 개인정보는 비식별 조치 이후에도 적정성 평가와 사후관리를 통해 지속해서 점검하는 과정이 필요하나, AI 허브는 4단계의 검수 체계가 있음에도 초기 점검 과정에서 제대로 된 검수를 하지 않았음이 드러났다. 이후 무결성 점검 과정을 통해 식별 정보로 남아 있는 데이터들을 비식별화하며 보완하였지만, 이는 3개월 뒤인 2021년 5월과 9월에 거쳐 재발하였고 지속적인 개인정보 유출에 우려를 사고 있다[7]. 2021년 상반기에는 '이루다' 사건이 세상을 떠들썩하게 만들었다. 이 또한, 민감정보를 비롯한 개인정보의 비식별 조치 및 접근 제어, 암호화 등의 미흡이 원인이었고, 이루다에 데이터베이스를 제공하였던 연애 관련 앱 서비스 이용자의 카카오톡 대화 내용이 사용자에게 노출되는 사태로 이어져 결국 이루다 서비스는 3주 만에 폐지되었다[8, 9].

2.2. 클라우드 환경에서의 개인정보보호 기술 현황

개인정보를 포함한 인공지능 학습데이터의 지속된 유출 사례를 해소하기 위해 연합 학습과 동형암호에 관한 연구가 활발히 진행되고 있다[6]. 연합 학습이란 인공지능이 컴퓨팅 자원을 분산 처리하여 개인의 데이터를 개인의 장치에서 학습하며 데이터 자체를 보호하는 방법을 말한다[6]. 동형암호는 평문을 연산한 후 암호화한 값과 암호화 후 연산한 값이 같아 개인정보를 노출하지 않으면서 데이터를 공유할 수 있는 암호 방법을 말한다[6]. 이러한 데이터 보호 연구가 지속됨에도 불구하고 개인정보를 분리·저장하지 않아 역으로 인공지능에 질의를 통해 개인정보를 취득할 수 있는 문제가 발생할 수 있다.

이러한 문제를 방지하고자 학계에서는 인공지능 학습데이터에 대한 분리, 암호화 및 블록체인 기술을 통해 해결하고자 하는 연구가 존재한다. Han Qiu 외 연구진은 단편화 및 분산화를 이용하여 클라우드에 데이터를 안전하게 저장할 수 있는 모델을 제시했다[2]. 이는 데이터를 두 조각으로 단편화하여 선택적 암호화를 진행하게 되는데, 각각 원본 데이터와 저장 공간을 많이 차지하는 데이터를 저장한다. 단편화된 두 데이터 조각은 데이터 접근성을 개선하였으나, 기밀성이 중요한 데이터는 단편화가 되어도 신뢰할 수 있는 스토리지에 넣어야만 안전하다고 할 수 있다. 이때 이 신뢰할 수 있는 스토리지를 찾아야 하는 추가 오버헤드가 발생하여 편의성이 저하된다.

또한 신뢰할 수 없는 클라우드 스토리지에 대한 검증에 대해 블록체인 기반 보안 아키텍처를 제안한 연구가 수행되었다[3]. 분산 클라우드 스토리지와 블록체인을 통합 후, 유전자 알고리즘으로 데이터를 학습하는 과정에서 발생하는 네트워크 전송 지연을 해결하였다. 이는 블록체인의 결합으로 신뢰성을 보장하며 학습된 데이터를 보호하였다. 그러나 대규모 데이터 통신 시에는 네트워크 전송 지연이 발생하며 이는 데이터 유실로 이어질 수 있다. 이에 원본 데이터에 대한 복제본을 만들어 데이터 유실을 방지하고자 했지만, 공간 오버헤드 초래한다는 한계점을 갖는다.

암호화에 기반하여 클라우드에 저장된 데이터에 대해 임의로 접근하는 것을 막는 방법을 제안한 연구가 수행되었다[4]. 블록 단위로 나눈 데이터를 공개키 암호로 저장해 사용자 개인의 데이터에 대한 기밀성과 무결성을 보장한다. 이 또한 키 관리에서 공간 및 성능 오버헤드가 발생해 인공지능의 학습 및 학습데이터 저장·관리에 적합하지 않다. 이렇듯 클라우드 내 데이터를 안전하게 저장하기 위한 연구가 활발히 진행되고 있지만, 보안성을 고려하지 않아 개인정보 저장에 구분되지 않으며 성능 및 공간 측면의 오버헤드를 유발한다. 이는 곧 비용 문제로 직결되며, 개인정보를 합리적인 방안으로 보호할 수 없다. 또한,

클라우드 내에서 인공지능이 데이터 학습 중에 발생할 수 있는 데이터 유출을 막을 수 없다. 따라서 인공지능이 개인정보가 포함된 데이터 학습 시, 학습데이터 처리 및 저장을 보호하고 안전한 학습 환경을 제공할 수 있는 방안이 필요하다.

3. 배경 지식

3.1 SGX (Software Guard Extensions)

SGX는 민감하고 가치 있는 정보들을 보유한 플랫폼에서 신뢰를 보장하기 위한 보안 프로세서 아키텍처이다[10]. SGX는 enclave라는 격리된 보호 영역을 가지며, 사용자의 enclave 설정으로 메모리 일부를 암호화하고 enclave 내부에서 실행되는 프로그램 이외에는 코드 및 데이터에 접근할 수 없도록 한다. 즉, enclave 영역에 속해 있는 데이터는 enclave 영역에서 실행되고 있는 코드만이 접근할 수 있다. 따라서 enclave는 승인된 위치에서만 실행되고, 외부 소프트웨어로부터의 변조에 보호되며 정보가 누출되지 않는다는 특성을 갖는다.

enclave의 내부 데이터 및 코드를 enclave의 외부로 보내기 위해서는 sealing 과정이 필요하다. sealing은 데이터를 암호화하여 내용을 공개하지 않고, 신뢰할 수 없는 메모리에 데이터를 쓸 수 있는 프로세스이다. enclave가 생성될 때 MRENCLAVE 값으로 다음에 사용하는 키에 바인딩하며, 이는 EGETKEY 명령어를 통해 하드웨어에서 수행된다[11]. EGETKEY는 enclave의 보안 작업을 위한 고유 키를 생성하는 키 생성 명령어이다[10]. 키를 요청하면 EGETKEY를 호출하여 현재 enclave에서 sealing을 수행한다. 이렇게 sealing한 데이터는 enclave 외부로 보낼 수 있게 된다. 반대로 enclave 외부에서 sealing 된 데이터를 enclave 내부로 다시 가져와 저장하려면 unsealing 과정이 필요하다. unsealing 할 데이터를 enclave 메모리에 할당하고, unsealing API로 EGETKEY를 호출하여 unsealing 작업을 수행한다[12]. 이후 unsealing 된 데이터는 다시 enclave 내에서 이용할 수 있다.

SGX의 EGETKEY를 바탕으로 한 sealing, unsealing으로 학습데이터는 enclave에 의해 보호된다. 하지만 인공지능이 개인정보를 포함한 데이터 학습하는 과정은 SGX만으로 완벽히 보호할 수 없다. SGX 내 코드가 취약할 경우 부채널 공격으로 개인정보가 누출되면 식별 가능하다는 문제가 발생할 수 있다. 구체적으로, 부채널 공격을 기반으로 한 DNN fingerprinting attack이 SGX에서도 여전히 유효하다[13].

3.2 비식별화 처리

개인정보는 데이터 삭제, 가명 처리, 집계처리, 범주화, 마스킹 기법을 단독 또는 복합적으로 활용하여

더는 개인정보로 간주하지 않도록 비식별 처리한다[14].

데이터 삭제(Data Reduction)는 개인을 식별할 수 있는 데이터 일부 또는 전체 정보를 삭제하는 것을 말한다. 구체적으로 원본 데이터에서 단순히 식별자 일부나 전체를 삭제하거나, 오차범위를 벗어나는 자료를 삭제할 수 있는 레코드 삭제, 식별자와 함께 잠재적으로 식별 가능성이 있는 속성값을 모두 제거하며 개인정보 침해 요소를 줄인다[14].

가명 처리(Pseudonymisation)는 개인을 분별할 수 있는 일부 또는 전체 정보를 대체하여 특정한 추가 정보가 없다면 개인을 식별할 수 없도록 처리하는 기법을 말한다. 미리 정의한 일정 규칙에 따라 식별자를 대체 및 가공화하는 휴리스틱 가명화를 적용해볼 수 있다[14, 15].

총계처리(Aggregation)는 식별자에 전체 혹은 부분적인 데이터에 통계치를 적용해 특정 개인을 식별할 수 없도록 하는 것이다. 전체 합이나 평균값을 집계하여 나타내는 총계처리, 집계 처리된 값을 올림, 내림, 반올림하여 최종 집계 처리하는 라운딩, 단순히 데이터 배열을 뒤섞는 재배열 방법이 해당한다[14].

범주화(Suppression)는 데이터를 대푯값이나 구간 값으로 범주화하여 특정 값을 알지 못하도록 하는 것이다. 데이터의 평균이나 범주 값으로 치환하거나 임의의 수를 기준으로 올림이나 내림을 적용하는 무작위 라운딩으로 명확한 값을 숨긴다[14].

마스킹(Masking)은 개인을 식별할 확률이 높은 데이터를 일정 대체 값으로 변환하는 것을 뜻한다. 이는 식별자에 임의 숫자를 더하거나 곱하는 등의 노이즈를 추가하는 임의 잡음 추가, 공백과 대체 기법을 이용한다[14].

비식별화는 2가지 이상의 정보 결합으로 개인을 특정해볼 수 있는 성명과 주소의 조합, 이메일 주소, 사용자 ID나, 수치적인 범위를 나타낼 수 있는 카드 지출액, 위치정보, 우편번호를 비롯하여 의료기록, 생체정보, 범죄 기록, 종교 등의 민감정보, 유추 및 재조합으로 식별 가능성이 발생해서는 안 되는 고유식별정보나 카드 및 계좌정보, 전화번호 등 개인정보의 범위에 대한 한계 없이 전사적으로 보호할 수 있다.

4. 인공지능 프레임워크

본 논문에서 제안한 프레임워크는 SGX 기술을 이용하여 enclave 내부에서 인공지능 학습데이터를 분류 후 비식별 처리한다. 비식별화된 데이터를 학습 과정에서 활용할 경우 학습 모델의 유용성이 손상될 수 있는 문제에 대해, 데이터 제공자들과 합의하여 threshold를 설정한 뒤에 모델의 정확도가 일정 수준까지 달성하는 수준으로 비식별화를 수행할 수 있다[16]. 이때, 비식별화로 인한 모델의 유용성이 손상될 수 있기 때문에 [16], 일정 수준 이상의 추론

정확도를 보장하기 위해 필요한 모델의 성능에 대한 threshold를 데이터 제공자들과 합의하여 데이터에 대한 비식별화 수준을 결정하도록 한다. 인공지능의 학습 또한 enclave 내부에서 수행되는데, 비식별 처리 후 sealing 하여 클라우드에 저장한다. 저장된 데이터는 인공지능 학습 시에만 enclave로 가져와 unsealing 후 사용할 수 있다. 이는 학습 과정이 기본 OS 환경에서도 기밀성과 무결성을 유지하며 실행될 수 있음을 나타낸다.

이 과정은 멀티스레딩(multi-threading)을 지원하는 SGX 특성에 따라 가상 머신 간 통신 오버헤드 등을 최소화하기 위해 하나의 가상 머신에서 수행하며, 데이터 제공자로부터 학습데이터를 받아 구체적인 목적의 학습 용도로도 활용할 수 있도록 지원한다. 이때, 데이터 제공자에게 보호 대상 데이터를 필수 제공하도록 요청하여 수합한 데이터를 enclave 내부로 가져온다. 학습데이터는 다음의 기준이 적용된 enclave 내 데이터 구분 코드를 통해 분류된다.

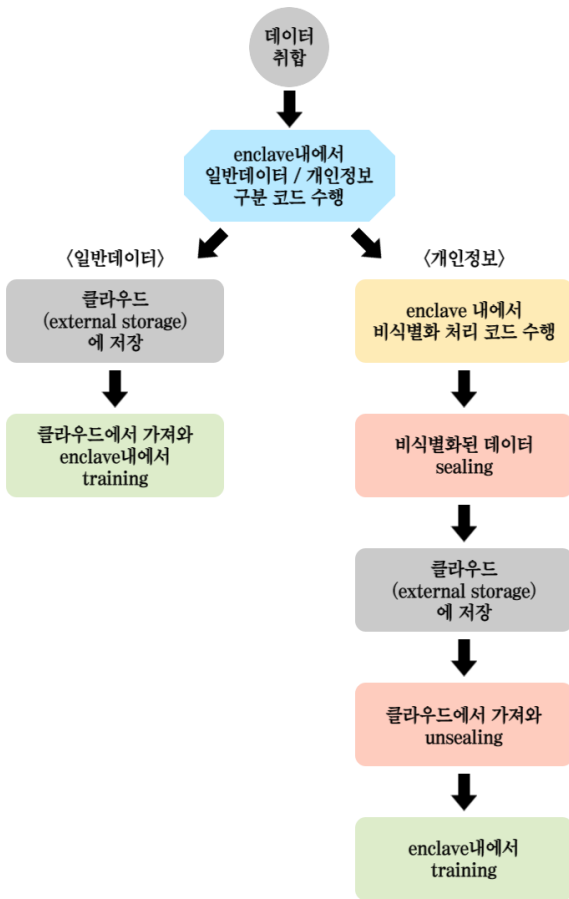


그림 1. enclave 기반 일반데이터 및 개인정보에 대한 인공지능 학습 과정

인공지능의 학습데이터는 정보의 민감도에 따라 일반정보와 개인정보로 분류한다. 개인정보는 개인을 식별할 수 있는 정보로, 특정 개인의 사생활을 현저히 침해할 우려가 있는 민감정보와 고유식별정보를

비롯하여 해당 정보만으로 특정 개인을 식별할 수 없더라도 다수의 정보와 결합 시 식별할 수 있는 간접식별 정보를 포함한다[5]. 일반정보는 개인정보를 제외한 모든 정보로 정보가 유출되더라도 해당 정보 자체와 이들의 조합으로 개인을 식별할 수 없는 정보로 정의한다.

구분된 학습데이터는 일반데이터와 개인정보 여부에 따라 그림 1의 과정으로 저장 및 학습을 수행한다. 그림 1에서 볼 수 있듯이 개인정보에 대해서는 비식별화 처리 후 원본 데이터를 과기한다. 이 과정에서 SGX sandbox인 Ryoan를 이용함으로써 SGX 부채널 공격 및 enclave 내 코드의 취약성으로 인한 원본 데이터 유출에 대응할 수 있도록 조치한다[17].

4.1 개인정보

구분된 개인정보는 enclave 내부에 저장된 비식별화 처리 코드를 통해 외부에 정보가 공개되더라도 개인을 식별할 수 없는 형태로 변경된다. 분류된 개인정보는 데이터 삭제, 가명 처리, 총계처리, 범주화, 마스킹의 기법을 적용하여 비식별화된다[5].

비식별화된 데이터는 sealing을 통해 암호화할 수 있다. 해당 데이터를 클라우드에 저장하기 위한 고유키가 필요하며, 이 Seal Key를 생성하기 위해 SGX EGETKEY feature를 사용한다. enclave는 EGETKEY명령어를 사용하여 Seal Key를 생성하고 비식별화된 데이터를 암호화하기 위해 ecall_seal 함수를 호출한다[12]. 하드웨어 기반 Seal Key는 요청 시 내부적으로 파생되며 enclave에 노출되지 않는다. 이러한 기능을 바탕으로 비식별화된 데이터는 Seal Key로 암호화한다. Seal Key는 enclave의 Position과 내용 등을 해시(Hash)로 구한 값인 MRENCLAVE 또는 Enclave Author의 Public-Key를 해시로 구한 값인 MRSIGNER를 근거로 얻게 된다[12]. 이렇게 sealing 된 데이터는 외부 스토리지인 클라우드로 보내 저장한다.

인공지능이 학습데이터를 습득하기 위해서는 unsealing 과정이 필요하다. sealing 되어 저장된 데이터는 SGX 내에서만 복호화할 수 있고, 인공지능 학습 코드는 enclave 내에서만 수행할 수 있으므로 클라우드에 저장된 데이터를 enclave 내부로 가져온다. 같은 MRENCLAVE값을 가진 enclave만 비식별화된 데이터를 unsealing 할 수 있으므로 sealing 된 데이터는 enclave 메모리에 할당된다. 그 후 unseal API를 호출하여 unsealing을 수행한다. unsealing 중 생성된 해시값과 sealing 중 생성된 해시값이 일치하는지 확인해야 하여 일치하면 unseal 함수를 수행하여 sealing을 해제하고, 비식별화된 데이터에 대해 인공지능 학습 코드를 통해 학습시킬 수 있게 된다[11]. 해당 과정을 통해 개인정보를 특정한 목적의

서비스 제공을 위한 학습 용도로 이용할 수 있으며, 이는 개인정보를 안전하고 효율적으로 활용할 수 있는 방안이 된다.

4.2 일반데이터

enclave 내에서 구분된 일반데이터는 클라우드 외부 스토리지에 저장한다. 이때, 일반데이터는 비식별 조치가 필요하지 않은 데이터로, 누구에게나 공개되어도 개인정보보호에 대한 문제가 발생하지 않는 정보이다. 이는 다른 목적의 인공지능 학습에도 활용될 수 있도록 공공 학습데이터로 클라우드에 제공한다. 즉, 해당 클라우드에는 누구나 접근하여 인공지능 학습데이터로 이용할 수 있다. 하지만 인공지능 학습에 있어 학습 알고리즘은 보호가 필요하다.

이에 인공지능 학습 코드를 enclave 내 저장 후, 클라우드 내부에 있는 학습데이터를 각 이용자의 enclave로 가져와 학습을 수행한다. 이는 데이터 이용자들의 인공지능 학습 알고리즘과 모델 파라미터들을 노출 없이 자유롭게 학습데이터를 이용할 수 있어, 학습데이터에 대한 접근성과 활용도를 높임과 동시에 보안성까지 제고한다.

5. 정성적 평가

5장에서는 제안한 인공지능 학습 프레임워크의 기대효과를 선행 연구와 함께 비교 분석하여 표 1에 나타낼 수 있다. 평가지표는 비용 효율성, 성능 효율성, 보안성 측면에서 프레임워크의 기대효과를 분석하였으며, 이를 5.1장에서 정의한다.

표 1. 정성적 평가표

평가지표 논문	비용 효율성	성능 효율성	보안성
학습데이터에 사용되는 개인정보보호를 위한 SGX 기반 인공지능 학습 프레임워크	-클라우드 기반 다량의 학습데이터 저장으로 관리 비용 절감 -SGX 기반 sealing 암호화로 기존 암호화 대비 비용 절감 -동일한 환경에서의 학습 데이터 비식별 조치 및 인공지능 학습으로 비용 절감 효과	-일반데이터와 개인정보의 분리 저장 및 관리 가능 -성능 오버헤드 감소 -공간 오버헤드 감소	-Enclave 의 접근통제로 데이터 유출 방지 -Enclave 에서 코드를 수행함으로써 데이터 처리 및 학습 과정의 개인정보 보호 가능 -Seal Key 를 이용한 개인정보 sealing 으로 안전하게 저장 가능
A user-centric data protection method for cloud storage based on invertible DWT[2]	-클라우드에 저장함에 따라 이용량에 따른 비용 부과 -단편화된 데이터에 대해 효율적인 배치 알고리즘 적용으로 인한 추가 비용 발생	-개인정보 분리 가능 -선택적 암호화에 개인정보가 포함 기준 필요 -다량의 데이터를 보관하는데 있어서 지연성이 높음 -공간 오버헤드 큼	-선택적 암호화로 인한 전체 데이터 안전성 보장 불가
Block-secure: Blockchain based scheme for secure P2P cloud storage[3]	-블록체인 구축 비용 발생 -데이터 기록을 위한 거래 수수료 발생	-데이터 자체를 분리하여 암호화할 수 있지만, 분리된 데이터를 일반데이터와 개인정보로 분류하는 것은 불가함 -공간 오버헤드 큼 -대규모 네트워크에서 통신 속도 저하	-ECDSA 암호화 기반 데이터 보안성 제고 -P2P 통신으로 데이터 무결성 보장
A Study on the Protection of User Data in the Cloud System[4]	-클라우드에 저장함에 따라 이용량에 따른 비용 부과 -암.복호화에 사용되는 개인키와 공개키 관리 비용이 높음	-데이터 저장 시 개인정보 분류 불가 -공간 오버헤드 큼 -모든 데이터에 대해 암호화하므로 성능 오버헤드가 큼	-공개키 암호방식으로 개인키 소유자만 데이터 접근 가능 -데이터 저장 시 개인정보에 대한 차등 보안 수단 적용 불가 -블록의 일부만을 암호화하기 때문에 복구 시스템의 암.복호화에 사용되는 개인키와 공개키의 효율적인 관리 메커니즘이 필요

5.1 평가지표

첫 번째는 비용 효율성이다. 비용 효율성은 자원, 시간, 비용 등의 적은 양으로 목적을 합리적으로 추구한 정도이다. 본 연구에서는 비용적 측면에서 클라우드 내 스토리지 구축에 대한 비용을 얼마나 절감하였는가를 나타낸다. 두 번째는 성능 효율성으로, 선택한 수단이 클라우드 내 저장할 데이터 중 개인정보를 분리하는 방안으로서 성능을 개선한 정도이다. 클라우드 스토리지에 저장된 개인정보는 어떠한 정보가 유출되더라도 개인을 식별할 수 있는 요소가 최소화되어야 한다. 동시에 일반데이터와 개인정보를 분리 저장하면서도 성능적, 공간적 오버헤드를 감소할 수 있어야 한다. 마지막으로 보안성은 특정한 정보가 유출되었을 때도 안전성을 보장할 수 있는 정도를 나타낸다. 학습데이터에 이용되는 개인정보는 다양한 원인으로 유출될 수 있다. 이에 대해 비인가 된 접근이나 외부 공격으로부터 일정 수준의 안전을 도모할 수 있어야 한다.

5.2 평가 결과

다음은 5.1장에서 설정한 지표에 따라 평가한 표 1에 대한 결과이다.

1) 비용 효율성

선행 연구의 블록체인 도입이나 암호화 키 관리, 단편화된 데이터의 효율적인 관리를 위한 특정한 알고리즘은 비용 발생으로 이어진다[2, 3, 4]. 특히, 선택적 암호화를 사용한 연구는 데이터를 단편화하여 배치 알고리즘을 적용함에 따라 추가적인 비용이 발생하였다[2]. 블록체인을 기반으로 데이터를 안전하게 보호하고자 했던 연구는 블록체인의 트릴레마 중 하나인 전송 지연 문제를 해결하였으나, 블록체인을 구축하고 관리하는 비용과 함께 확장성에 대한 공간 오버헤드가 발생하였다[3]. 본 논문에서 제안한 방안은 SGX를 기반으로 하여 enclave 내부에서 Seal Key를 보관하고 있어 sealing 암호화에 대한 추가적인 키 관리 공간이 필요하지 않기 때문에 공간 효율성을 증대할 수 있다. 또한, 학습데이터를 비식별화하고 인공지능이 데이터를 학습하는 과정 모두 enclave에서 수행되어 비용을 절감하는 효과를 가져올 수 있다.

2) 성능 효율성

암호화키 관리, 블록체인을 위한 공간 할당은 공간적·성능적 오버헤드를 불러일으키고, 결과적으로 비용 발생으로 이어진다[2, 3, 4]. 블록체인을 적용한 연구는 데이터를 분리하여 암호화할 수 있지만, 분리된 데이터를 한 번에 저장해야 했다[3]. 개인정보를 분리할 수 있더라도, 대용량의 데이터가 한 번에 저장되어 있어 지연 시간이 높아 개인정보를 안전하고 효율적인 방법으로 적합하지 않았다[2]. 제안한 방법은 일반데이터와 개인정보를 분류한 상태로 각각 다른

동작 과정을 거쳐 결괏값을 도출한다. 개인정보의 경우 enclave 내부에서 비식별화 과정을 거쳐 클라우드에 저장되고, 추가적인 암호화가 필요하지 않은 일반데이터는 클라우드에 저장한다. 이분화된 방법은 데이터 접근에 대한 효율성을 높이는 동시에 공간적 오버헤드와, 지연 시간을 줄일 수 있는 효과적인 방법으로 적용될 수 있다.

3) 보안성

선행 연구에서는 모든 데이터에 대해 공개키 암호화를 수행하였을 때 키를 가진 사용자만 접근할 수 있도록 하였다[3, 4]. 그러나 선택적 암호화를 이용하였던 연구[2]는 암호화를 수행하는 부분에 반드시 개인정보가 위치할 때만 안전하다고 할 수 있어, 개인정보가 포함된 대용량의 학습데이터 저장 및 관리에는 비효율적이다. 또한, 일반데이터와 개인정보에 대해 같은 수준의 암호화 방법을 적용해 민감정보 유출에 대한 우려가 크다. 제안한 인공지능 학습 프레임워크는 SGX 기술로 enclave 내에서 일반데이터와 개인정보 분류하고, 개인정보에 비식별 처리를 수행한다. 때문에, enclave 영역은 비인가자에 대한 접근을 통제할 수 있어 데이터 유출을 방지한다. enclave 내부에서 진행되는 인공지능의 학습은 학습 과정에서 발생하는 데이터 유출 문제를 예방할 수 있고, Seal Key를 이용해 개인정보를 암호화하며 하드웨어 자원 보호를 보장할 수 있다.

6. 결 론

본 논문에서는 학습데이터에 사용되는 개인정보 유출 및 악용을 방지하기 위해 SGX 기반 인공지능 학습 프레임워크를 제안하였다. Intel SGX 기술을 중심으로 클라우드와 결합하여 다량의 학습데이터를 저장할 뿐만 아니라 개인정보를 분리 및 비식별 처리하여 저장함으로써 효과적으로 안전한 인공지능 학습 과정이 될 수 있다. 이후 비용 효율성, 성능 효율성, 보안성의 기준 지표를 설정하여 클라우드 환경의 개인정보보호 기술에 대한 선행 연구와 비교·분석하였다. enclave 내부에서 이뤄지는 sealing이나 인공지능 학습으로 공간 오버헤드를 줄일 수 있고, 비용이 절감될 것으로 판단된다. 또한 일반데이터와 개인정보 분리를 통해 일반데이터는 빠른 속도로 통신을 지원하고, 안전성 조치가 필요한 개인정보는 비식별 조치를 수행하며 효과적인 처리가 가능하다. 더불어 키 저장과 비식별화 조치, 인공지능 학습을 수행하는 enclave 영역은 인가된 자만 접근할 수 있어 개인정보를 비롯한 소프트웨어 및 하드웨어 자원을 안전하게 보호할 수 있다. 향후 인공지능 학습 프레임워크의 구현 후 벤치마크를 사용하여 학습을 수행하고, 오버헤드를 정확히 평가할 수 있는 후속 연구를 진행할 예정이다. 이를 통해 정성적 평가뿐만 아니라 정량적으로 평가할 수 있는 객관적인 분석을 수행하여 제안

한 프레임워크의 효율성을 확인할 수 있다. 또한 학습 과정에서 비식별화 기술의 종류에 따라 학습 모델 성능을 분석할 예정이다.

Acknowledgement

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원(NRF-2021R1G1A100632611)과 2021년도 정부(산업통상자원부)의 재원으로 한국산업기술진흥원의 지원(P0008703, 2021년 산업혁신인재성장지원사업)을 받아 수행된 연구임.

참고문헌

- [1] 최정란, 송영미, 김철홍, 김선자, “인공지능을 위한 클라우드 컴퓨팅 산업 동향”, 한국전자통신연구원, 2017.
- [2] Han Qiu·Hassan Noura·Meikang Qiu, "A User-Centric Data Protection Method for Cloud Storage Based on Invertible DWT", IEEE Transactions On Cloud Computing, vol.9, 1293-1304p, 2019.
- [3] Li, Jiaying, Jigang Wu, and Long Chen, "Block-secure: Blockchain based scheme for secure P2P cloud storage", Information Sciences 465, 219-231p, 2018.
- [4] 이애리, "A Study on the Protection of User Data in the Cloud System", 디지털정책연구, 제10권, 제11호, 2012.
- [5] 개인정보보호법.
- [6] 김현지·서화정, "Trends in deep learning technology to protect the privacy of training data", 한국정보처리학회 춘계학술대회 논문집, 제28권, 제1호, 451-453p, 2021.
- [7] 김윤진, “데이터담 ‘AI허브’서 21개월 간 개인정보 유출, 뉴스로드, <http://www.newsroad.co.kr/news/articleView.html?idxno=15439> Accessed: 2021.11.16. [Online]
- [8] 강태욱, "이루다 서비스와 AI모델 학습에서의 개인정보의 처리", DAIG2021, 제2호, 125-159p, 2021.
- [9] 전승재·고명석, "Possibility of Using Personal Information as Machine Learning Data Seen Through the Iruda Case", KCI저널, 정보법학 제25권, 제2호, 2021.
- [10] F. McKeen, I. Alexandrovich, A. Berenzon, and C.V. Rozas, "Innovative Instructions and Software Model for Isolated Execution," Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, 2013.
- [11] Ittai Anati, Shay Gueron, Simon P Johnson, Vincent R Scarlata, "Innovative Technology for CPU Based Attestation and Sealing", Intel Corporation, 2013.
- [12] Intel, "Intel(r) Software Guard Extensions Developer Guide", 2019.
- [13] S. Hong, M. Davinroy, Y. Kaya, S. Locke, I. Rackow, D. Dachman, T. Dumitras, "Security Analysis of Deep Neural Networks Operating in the Presence of Cache Side-Channel Attacks", arXiv'18, 2018.
- [14] 개인정보보호위원회, "가명정보 처리 가이드라인", 2021.
- [15] 김혜련, "통계적 목적의 개인정보보호와 비식별화", 통계청, 통계연구 특별호, 35-51p, 2016.
- [16] 정홍주, 이나영, 설수진, 한경석, “개인정보의 비식별화에 따른 기계학습의 예측 정확도 분석 연구”, 정보과학회논문지, 제47호, 906-901p, 2020.
- [17] T. Hunt, Z. Zhu, Y. Xu, S. Peter, E. Witchel, "Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data", 12th USENIX Symposium on OSDI '16, 2016.

CFG 생성 타당성 검사를 위한 테스트 자동화 연구

정소희^o, 김동우, 최윤자
 경북대학교 IT대학 컴퓨터학부

정소희^o cwc06129@gmail.com, 김동우 kdw9242@gmail.com, 최윤자 yuchoi76@knu.ac.kr

Test Automation for CFG Generation Validity Check

Sohee Jung^o, Dongwoo Kim, Yunja Choi

School of Computer Science and Engineering, College of IT engineering
 Kyungpook National University

요 약

C 프로그램의 제어흐름분석 및 AST(Abstract Syntax Tree) 분석은 프로그램의 분석 및 검증에서 필수 요소이다. 그러나 현존하는 도구를 사이즈가 큰 프로그램에 적용하다 보면 제어흐름분석 부분에서 Loop 구문, 중첩문을 탐색할 수 없는 등의 오류들이 발생하였다. 이를 해결하기 위해 추가적인 수정 작업이 이루어졌지만 제어흐름분석을 통해 도출되는 제어흐름그래프(CFG)가 제대로 생성되었는지에 대한 확신은 할 수 없었다. 이에 본 연구에서는 도출되는 CFG에 대한 타당성 검사를 위해 테스트 자동화 도구를 개발하였다. 개발된 도구를 Erika IoT 임베디드 운영체제, OSEK/VDX 기반의 개방형 차량용 운영체제인 Trampoline을 포함한 총 95개의 코드에 적용한 결과, 4분 25초만에 총 7184개의 함수 중 7183개의 함수가 테스트에 통과함을 보임으로써 CFG 생성의 타당성을 확인하였다.

1. 서 론

C 프로그램의 제어흐름분석 및 AST(Abstract Syntax Tree) 분석은 프로그램 분석 및 검증에서 필수 요소이다[1,2,3,4]. 이를 수행하기 위한 선행 연구가 있었다[5]. 해당 도구를 사이즈가 큰 프로그램에 적용해보니 오류가 발생하였고 이를 해결하기 위해 많은 수정 작업이 수행되었다. 하지만 수정 작업 이후에도 제어흐름그래프(CFG)가 제대로 생성되었는지를 확인할 수 없었다. 따라서 CFG 생성의 타당성을 검사하는 작업이 별도로 필요하였고 타당성 검사를 위한 테스트를 진행하였다. 다양한 사이즈의 C 프로그램 코드들에 대해 일일이 수작업으로 테스트를 진행하다 보니 어려움이 있었다. 그래서 CFG 생성의 타당성 검사를 위한 테스트 자동화 도구를 개발하게 되었다. 해당 도구는 기존 C코드와 CFG를 통해 재생성된 코드를 기반으로 테스트를 진행한다.

본 연구에서 개발한 도구를 Erika IoT 임베디드 운영체제[6], OSEK/VDX 기반의 개방형 차량용 운영체제인 Trampoline[7]을 포함한 총 95개의 코드에 적용해보았다. 테스트 입력으로 사용된 95개의 C코드들은 다양한 크기로 이루어져 있으며 총 7184개의 함수를 포함하고 있다. 테스트 결과 7184개의 함수 중 어셈블리 코드를 포함하고 있는 1개의 함수를 제외한 7183개의 함수에서 테스트가 통과함을 보임으로써 CFG 생성의 타당성을 확인하였다.

2. 연구 배경

2.1 제어흐름그래프(CFG)

제어흐름그래프(CFG)는 프로그램이 실행 중에 거칠

수 있는 모든 경로를 그래프로 표현한 것이다. CFG는 프로그램 분석과 검증에 있어서 필수적인 요소이다. 제어흐름에서 제어구조가 바뀌는 부분을 결정 포인트(Decision point)라고 하는데 이 결정 포인트에서 유효한 테스트 케이스를 도출해 낼 수 있기 때문에 소프트웨어 테스팅에서 제어 흐름 분석이 사용된다[2]. CFG는 자료 흐름 분석(data flow analysis), 제어 종속 분석(control dependence analysis), 예외 분석(exception analysis), 리그레션 테스팅(regression testing), 프로그램 슬라이싱, 테스트 데이터 생성 등을 위해 쓰일 수 있다[3].

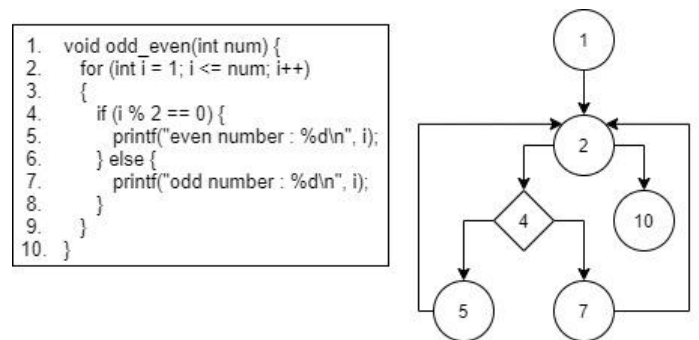


그림 1 예제 코드와 생성된 CFG

그림 1은 예제 C코드와 해당 코드를 CFG로 나타낸 것이다. 각 CFG 노드에 표시된 번호는 각 구문이 위치한 코드의 라인수이다. 노드를 연결하는 간선과 화살표의 방향을 통해 후행 노드를 알 수 있다. 여기서 후행 노드는 다음에 실행될 수 있는 구문을 이야기한다.

2.2 AST와 CFG를 이용한 연구 사례

제어흐름분석 및 AST 분석을 수행하는 도구[5]가 선행 연구되었다. 그림 2의 점선으로 된 박스에 표시된 절차는 선행 연구된 도구에서의 제어흐름분석 부분이다. 해당 도구는 Understand[8]와 Perl API를 이용하여 CFG 정보를 받아온다. 이 때 CFG 정보에는 각 구문의 종류를 구분하기 위한 node kind, 구문이 시작되는 라인 정보를 가지는 start line, 구문이 끝나는 라인 정보를 가지는 end line, 해당 구문이 Block 구문인지를 파악할 수 있게 해주는 end structure node, 그리고 단방향성 제어 흐름 정보를 가지고 있는 child node가 포함된다. 그림 2에서 볼 수 있듯이 이 도구는 정보 추출 단계와 CFG 변환 단계, CFG 테스트 단계로 구성된다.

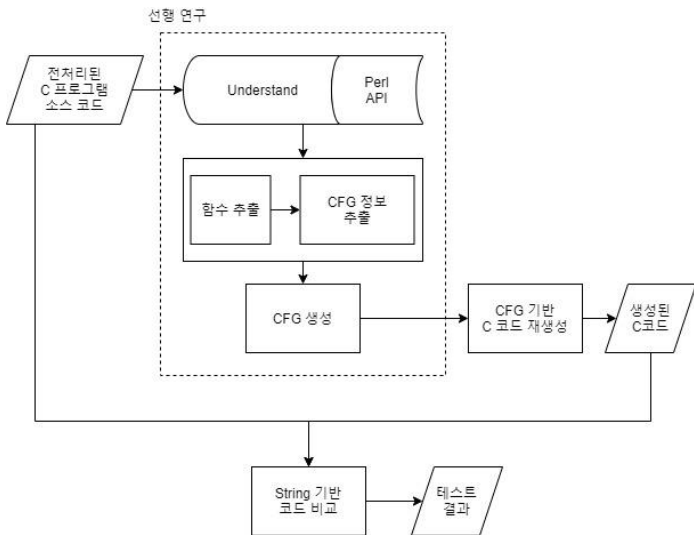


그림 2 Overview of tool for Test Automation

해당 도구를 다양한 사이즈의 프로그램에 적용해본 결과 사이즈가 큰 프로그램에서 오류가 발생하였다. 대표적인 오류는 다음과 같다. 첫번째, Loop 구문에 대한 탐색이 진행되지 않았다. 두번째, 중첩문에 대한 탐색이 제대로 진행되지 않았다. 첫번째 오류의 경우 CFG 탐색 방법이 스택을 이용한 상향식 접근 방식이었기에 발생하였다. 이는 CFG 탐색 방법을 함수 몸체의 상위 구문부터 하향식으로 접근하도록 수정함으로써 해결할 수 있었다. 두번째 오류의 경우 if 구문을 만나면 else if 구문과 else 구문을 무시하고 탐색하는 형식으로 오류가 나타났다. 이는 if 구문은 탐색하지만, 그에 종속된 then 구문과 else 구문을 탐색하지 않아서 발생하는 오류였다. 해당 오류는 각 구문을 그에 종속된 자료구조와 함께 탐색하게 함으로써 해결할 수 있었다. 하지만 CFG 탐색 방법을 고쳤다고 하더라도 CFG가 제대로 생성되는지에 대한 확신은 할 수 없었다.

선행 연구의 실험 결과를 보면 ERIKA IoT 프로그램을 대상으로 CFG 및 AST를 검증하였다. 하지만 해당

프로그램은 사이즈가 작을 뿐만 아니라 이에 속한 함수의 수가 10개 미만으로 매우 적다. 이는 테스트가 충분히 이루어지지 않았음을 알 수 있는 증거이다. 따라서 CFG가 제대로 생성되는지에 대한 추가적인 검증이 필요하였다. 이에 CFG 생성의 타당성 검증을 위한 테스트 자동화 도구를 개발하게 되었다.

3. C코드 재생성 및 테스트 자동화

3.1 C코드 재생성

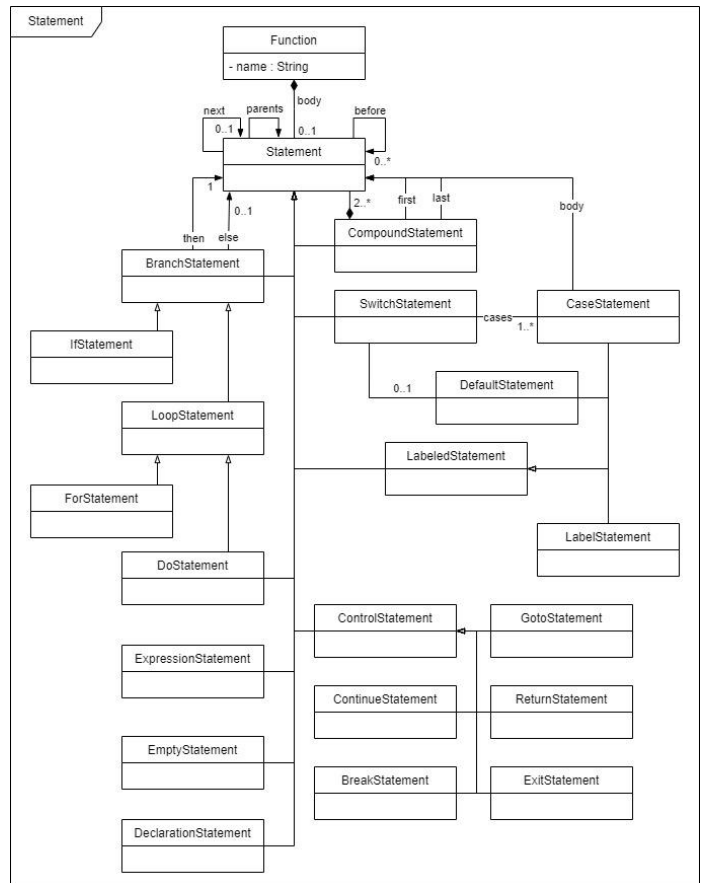


그림 3 구문들의 클래스 다이어그램

그림 3은 구문들 간의 관계를 나타낸 클래스 다이어그램이다. 해당 도구에서는 Understand로부터 받아온 CFG 정보들을 이용해 CFG 노드를 만든다. CFG 노드와 구문(Statement) 사이에는 매핑이 존재하는데 이 매핑을 기반으로 구문은 CFG 노드에 저장된 정보를 참조할 수 있다. 해당 정보들을 이용해 구문들의 관계를 정의할 수 있으며 이 관계가 CFG의 간선이 된다. 구문들 사이에는 많은 관계들이 존재하며 대표적으로 next, before 관계가 존재한다. next 관계는 해당 구문 다음에 이어질 수 있는 구문을 이야기하고, before 관계는 해당 구문의 앞에 올 수 있는 구문을 이야기한다. C코드의 재생성 과정에서는 next 관계를 이용해 다음에 이어질 수 있는 구문을 탐색함으로써 재생성을 진행하였다. C코드 재생성 부분을 테스트하기

위해서는 구문들 간의 연결 관계를 확인하는 작업이 필요하였다. 그러나 이를 자동으로 확인하기에는 어려움이 있어서 일일이 수작업으로 확인하는 과정을 거쳤다.

그림 3의 클래스 다이어그램에서 볼 수 있듯이 함수는 구문의 집합이다. 즉, 함수 몸체는 구문들로 이루어져 있다. 구문에는 다양한 객체 타입이 존재하는데, 각각의 객체 타입은 고유의 자료구조를 가지고 있다. 몇 가지를 예로 들자면 If 구문에는 then, else 속성으로 구문들이 종속되어 있고, Switch 구문에는 cases 속성으로 Case 구문이 종속되어 있다. 이러한 종속 관계들은 곧 그 구문의 자료구조가 된다.

C코드의 재생성은 하나의 함수를 기준으로 이루어진다. 함수의 첫번째 구문부터 하향식 접근으로 제어흐름을 탐색하여 구문의 객체 타입을 확인한다. 구문의 객체 타입을 확인한 후, 구문의 문자열, 식별자, 숫자, 키워드 등을 포함하는 어휘항목(lexeme) 데이터를 기반으로 코드를 제작하여 소스코드 객체에 추가한다. 이 과정은 각 객체 타입의 자료구조에 따라 진행된다. 만약 객체 타입의 자료구조에 따라 탐색이 완료되었다면 함수 몸체에 저장된 다음 구문을 탐색하게 된다. 함수의 마지막 구문까지 탐색이 완료되면 모든 구문이 포함된 최종 소스코드 객체를 얻을 수 있다. 이 소스코드 객체가 재생성된 C코드이다. 이를 시퀀스 다이어그램으로 나타내면 그림 4과 같다.

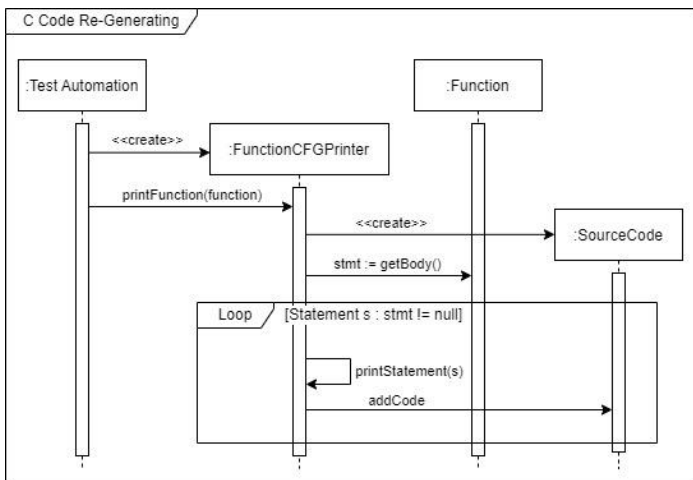


그림 4 C코드 재생성의 시퀀스 다이어그램

C코드 재생성 시퀀스 다이어그램에서 Test Automation 객체는 FunctionCFGPrinter 객체를 생성한다. FunctionCFGPrinter 객체는 CFG를 하향식으로 탐색하면서 C코드를 재생성하는 객체이다. 이 객체는 특정 함수(function)를 파라미터로 가지는 printFunction이라는 메소드가 호출되면 SourceCode 객체를 생성한다. 그 후, getBody 메소드를 호출해 printFunction 메소드의 파라미터로 넘어온 Function 객체의 함수 몸체를 이루는 구문들을 가져오게 된다.

마지막으로 함수 몸체에 저장된 구문이 없을 때까지 printStatement 메소드를 실행하면서 lexeme 데이터를 이용해 해당 구문에 맞는 코드를 제작한 후, 이를 SourceCode 객체에 저장해준다.

3.2 테스트 자동화

본 도구는 도구의 입력으로 사용되는 C 프로그램 코드와 CFG 기반으로 재생성된 C코드를 각각 리스트로 저장한 후, 리스트를 문자열로 변환한다. 그리고 두 문자열을 비교하여 문자열이 같으면 테스트 성공을 반환하고, 같지 않으면 테스트 실패를 반환하는 형식으로 테스트 자동화를 수행하였다. 그림 5는 기존 C 프로그램 코드와 CFG 기반으로 재생성된 C코드를 비교한 그림이다. 자세한 테스트 자동화 과정은 다음과 같다.

(A) Original C code	(B) Generated C code
<pre> 1. /* Calculate Odd number 2. * and Even number 3. */ 4. void odd_even(int num) 5. { 6. for (int i = 1; i <= num; i++) 7. { 8. if (i % 2 == 0) // even number 9. { 10. printf("even number : %d\n", i); 11. } 12. else // odd number 13. { 14. printf("odd number : %d\n", i); 15. } 16. } 17. }</pre>	<pre> 1. void odd_even(int num) { 2. for (int i = 1; i <= num; i++) 3. { 4. if (i % 2 == 0) { 5. printf("even number : %d\n", i); 6. } 7. else { 8. printf("odd number : %d\n", i); 9. } 10. } 11. }</pre>

그림 5 Original C code와 생성된 C코드 비교

과정 1 CFG 기반으로 재생성된 C코드에는 주석이나 불필요한 공백이 없지만, 기존 C 프로그램 코드에는 주석이나 불필요한 공백이 있을 가능성이 높다. 따라서 주석과 공백을 처리해주는 전처리 과정을 거친다. 함수명과 함수 몸체를 구분해주기 위해 중괄호가 필요하기 때문에 중괄호({})는 전처리 과정에서 처리하지 않는다.

과정 2 전처리 과정을 거친 후 C 프로그램 코드를 리스트에 저장하고, CFG가 저장되어 있는 소스코드 객체로부터 리스트의 형태로 코드를 가져온다. 리스트에 저장된 결과는 그림 6와 같다. 여기서 리스트 비교를 통해 테스트를 수행하게 되면, 줄바꿈으로 인해 테스트 결과가 달라질 수 있다. 따라서 문자열로 바꿔서 문자열 비교를 실행해야 한다.

List (A)		List (B)	
[1]	voidodd_even(intnum){	[1]	voidodd_even(intnum){
[2]	for(inti=1i<=num,i++){	[2]	for(inti=1i<=num,i++){
[3]	if(i%2==0){	[3]	if(i%2==0){
[4]	printf("oddnumber:%d",i)	[4]	printf("oddnumber:%d",i)
[5]	}else{	[5]	}
[6]	printf("evennumber:%d",i)	[6]	else{
[7]	}	[7]	printf("evennumber:%d",i)
[8]	}	[8]	}
[9]	}	[9]	}
		[10]	}

그림 6 리스트 저장 결과

과정 3 각각의 리스트를 문자열로 합쳐주게 되면 List(A)에 대한 문자열 하나와 List(B)에 대한 문자열 하나를 얻을 수 있다. 해당 문자열에서 CFG에 해당하는 부분은 함수 몸체 부분이기 때문에 함수명 부분과 중괄호를 제거한 문자열을 최종 문자열이라고 볼 수 있다. 그림 6의 예제에 적용해보면 그림 7과 같은 문자열을 얻을 수 있다.

string(A)	for(inti=1i<=num,i++){if(i%2==0)printf("oddnumber:%d",i) elseprintf("evennumber:%d",i)}
string(B)	for(inti=1i<=num,i++){if(i%2==0)printf("oddnumber:%d",i) elseprintf("evennumber:%d",i)}

그림 7 최종 문자열

최종 문자열을 비교해서 같으면 테스트 성공을 반환하고, 같지 않으면 테스트 실패를 반환한다.

4. 구현 및 실험

4.1 구현

개발된 테스트 자동화 도구는 Java 언어로 구현되어 있으며, 파일의 경로명과 파일의 이름을 가지는 엑셀 파일을 입력으로 사용한다. CFG의 생성이 제대로 이루어지는지에 대한 테스트들은 하나의 파일을 단위로 이루어지고, 해당 파일을 Understand로 분석한다. 분석 결과 파일은 udb 파일로 저장되며, 저장된 정보 중 필요한 정보를 Perl API를 통해 받아온다. 이 정보를 통해 CFG 생성 작업을 수행한다. CFG 생성 작업을 거쳐 생성된 CFG를 기반으로 C코드 재생성 과정을 거치게 되는데, 이 과정 결과로 생성되는 C코드와 기존 C 프로그램 코드를 문자열 비교하여 테스트 결과를 얻게 된다. 이 테스트 결과들은 함수명과 함께 pass 또는 fail로 엑셀 파일에 입력된다.

4.2 실험

본 도구에서는 C 프로그램 소스코드를 사용할 때

전처리된 소스 코드를 입력으로 사용한다. 전처리의 대상은 매크로이다. Understand와 Perl API를 이용한 정보 추출 단계에서 매크로와 관련된 정보를 추출하고자 하였지만, 매크로에 전달되는 파라미터 타입과 매크로에 정의되어 있는 구문에 대한 파악이 어려웠다. 따라서 전처리 작업이 추가적으로 필요하였다. 조건부 컴파일 매크로(#ifdef, #ifndef, #endif)의 경우 의미 관계를 파악하여 전처리하였고, 매크로 함수의 경우 함수로 제작하여 전처리하였다.

본 도구를 Erika IoT 임베디드 운영체제, OSEK/VDX 기반의 개방형 차량용 운영체제인 Trampoline을 포함한 총 96개의 코드에 적용시켜 실험해보았다.

표 1 실험 결과

코드 사이즈 (라인 수)	파일 개수	함수 개수	Pass	Fail
小(1~100)	40	193	192/192	0/192
中(101~500)	32	511	510/511	1/511
大(501~2000)	23	6481	6481/6481	0/6481
합계	95	7184	7183/7184	1/7184
실행 시간	4분 25초			

표 1은 테스트 자동화 도구를 적용한 실험 결과이다. CFG가 제대로 생성되는지 확인해본 결과 7184개의 함수 중 1개의 함수를 제외한 나머지 7183개의 함수에서는 제대로 CFG가 생성됨을 확인할 수 있었다. 총 테스트 수행 시간은 4분 25초이다.

테스트 실패가 반환된 함수를 분석해본 결과, 어셈블리 코드가 포함되어 있었다. 어셈블리 코드에 대한 처리는 추가적인 연구가 이루어져야 할 것으로 보인다.

5. 결론 및 향후 연구

본 논문에서는 제어흐름분석 결과 도출되는 제어흐름그래프(CFG)의 타당성 검사를 위한 테스트 자동화 도구를 개발하였다. C 프로그램의 제어흐름을 분석하고자 하는 도구의 개발은 기존에도 이루어졌다[2,5]. 하지만 제어흐름분석을 통해 도출되는 CFG를 테스트하는 과정은 대부분 수작업으로 이루어졌다. 하지만 본 논문에서 소개한 테스트 도구는 테스트 과정을 자동화함으로써 한 번의 실행으로 여러 소스 코드, 여러 함수에서 도출되는 CFG에 대한 테스트를 진행할 수 있다. 또한 해당 도구는 기존 제어흐름분석도구[5]의 테스트를 목적으로 개발되었지만 제안된 테스트 방식은 분석도구의 구현방식에 상관없이 일반적으로 적용될 수 있다.

본 연구에서는 어셈블리 코드에 대한 구문 탐색이

제대로 이루어지지 않았다. 향후 어셈블리 코드 탐색에 대한 부분을 추가로 고려하여 모든 구문 객체에 대한 CFG를 테스트할 수 있는 도구를 개발할 계획이다.

Acknowledgement

이 논문은 2021 년도 정부(교육부, 과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2016R1D1A3B01011685, NRF-2021R1A5A1021944).

참고 문헌

- [1] 김희연, 오호균, 김지훈, 유재욱, 신정훈, 김경곤 : “AST와 CFG에 기반한 Node.js 모듈 취약점 분석”, 한국정보처리학회 학술대회논문집, 26권, 2호, 4쪽, 2019
- [2] 정아랑, 김현수 : “XML을 사용한 C언어 소스 코드 분석 및 제어 흐름 분석”, 한국정보처리학회 추계학술발표대회 논문집, 15권, 2호, 4쪽, 2008
- [3] 조장우, 이정수 : “Java의 예외 제어 흐름을 포함한 제어 흐름 그래프 생성”, 한국정보과학회 학술발표논문집, 29권, 2호, 3쪽, 2002
- [4] Hoijin Yoon, “Effort Reduction of Unit Testing by Supporting CFG Generation and its Test Design”, International Journal of Smart Home, 7권, 5호, 10쪽, 2013
- [5] 정인준, 강민정, 김선희, 김동우, 최윤자 : “Understand 기반 C 프로그램 AST 분석”, 3쪽, 2016
- [6] ERIKA Enterprise, <http://erika.tuxfamily.org>
- [7] Trampoline, <http://trampoline.rts-software.org>
- [8] Understand, <https://www.scitools.com/>

Kubernetes를 활용한 멀티 클라우드 모니터링 솔루션

박신영¹⁽⁰⁾, 김병권², 배유진³, 심석인¹

동국대학교 산업시스템공학과¹, 동국대학교 경영학부², 동국대학교 의생명공학과³
 psh990331@naver.com, mountainrabbit01@gmail.com, ebeuy8636@gmail.com,
 shimdol22@naver.com

A New Solution for Multi Cloud Monitoring using Kubernetes

ShinYoung Park¹⁽⁰⁾, ByungGwon Kim², YuJin Bae³, SeokIn Shim¹

Department of Industrial and Systems Engineering, Dongguk University¹,

Department of Business administration, Dongguk University²,

Department of Biomedical Engineering, Dongguk University³

요 약

최근 디지털 전환(Digital Transformation)이 기업의 지속 가능성을 위한 필수 조건으로 떠오르면서 다양한 산업 분야에서 디지털 전환이 진행 중이다. 디지털 전환을 가속하는 기반 기술로 클라우드 컴퓨팅이 있다. 특히 한 개 이상의 클라우드를 사용하는 멀티 클라우드는 장애 발생 시 리스크를 최소화하기 때문에 유용하다. 그러나 멀티 클라우드 환경은 어플리케이션의 구조가 복잡해지기 때문에 클라우드에 대한 전문 지식이 없다면 운영될 수 없다. 본 논문에서는 쿠버네티스를 기반으로 AWS, Azure, GCP로 이루어진 멀티 클라우드 환경을 한 눈에 모니터링 할 수 있는 새로운 솔루션을 대시보드 형태로 제안한다. 또한 특정 클라우드에 장애 발생 시 연결된 다른 클라우드에 서비스를 자동으로 복구하는 Auto Healing 기능을 추가하여 개발자의 멀티클라우드 서비스 운영을 돕고, Human error를 제거한다.

1. 서 론

클라우드(Cloud, Cloud Computing)는 사용자가 직접 서버를 구성하는 온프레미스(on-premise) 방식에 비해 초기 비용이 적게 들며, 사용한 만큼만 비용을 지불하면 되기 때문에 경제적이다. 또한 사용자 요구량에 따라 유연하게 서버 크기를 증감할 수 있다는 장점 또한 가지고 있다. 그럼에도 불구하고 클라우드가 가진 치명적인 단점은 바로 사용자가 클라우드 리전(region) 단위의 장애에 대처하는 것이 거의 불가능하다는 점이다. 클라우드 시장에서 가장 높은 점유율을 차지하고 있는 AWS(Amazon Web Service)도 이러한 문제점에서 자유롭지 않다. 예를 들어 올해 2월에는 AWS 도쿄 리전 데이터센터에서 냉각 장치가 작동하지 않아 대규모 장애가 발생한 적이 있다[1].

이처럼 클라우드 상에 문제가 발생한다면, 클라우드를 이용하고 있는 기업들은 해당 CSP(Cloud Service Provider)가 자사의 기능을 복구할 때까지 서비스를 운영할 수 없다. 이러한 문제를 해결할 수 있는 방법이 바로 서비스를 멀티클라우드(Multi-cloud)로 운영하는 것이다. 멀티클라우드란 단어 그대로 하나의 서비스를 서로 다른 여러 개의 클라우드를 이용해 운영하는 것이다. 이 경우 하나의 클라우드가 장애를 겪더라도 다른 클라우드를 이용해 서비스를 계속 운영할 수 있다. 실제로

2018년 AWS 서울 리전 장애 시, 멀티클라우드를 이용하던 기업들은 장애를 겪지 않았다[2].

CSP가 제공하는 모니터링 툴은 자사 서비스에 대한 모니터링 기능만을 제공한다. 특히, 대부분의 IT기업들이 컨테이너를 이용해 배포를 진행하는 최근에는 클라우드와 더불어 컨테이너 오케스트레이션 쿠버네티스(Kubernetes, k8s)의 중요성이 커지고 있다. 하지만 AWS는 같은 회사의 쿠버네티스 엔진인 EKS(Elastic Kubernetes Service)에 대한 모니터링만 지원하고, MS Azure는 AKS(Azure Kubernetes Service)에 대한 모니터링만 지원하기 때문에 멀티 클라우드 시스템과 맞지 않는 부분이 있다. 즉, 다양한 CSP들을 멀티 클라우드로 운영해야 하는 필요성은 존재하지만, 한 번에 여러 클라우드의 관리형 쿠버네티스 기능을 동시에 모니터링 할 수 있는 도구는 존재하지 않는다.

따라서 본 프로젝트에서는 여러 클라우드의 관리형 쿠버네티스 기능을 모니터링할 수 있는 대시보드를 개발한다. 우선 앞서 언급한 문제점을 보완할 수 있도록 서로 다른 클라우드 상의 쿠버네티스 파드(Pod)의 개수, CPU사용량, API 호출량의 증감과 같은 클러스터의 변동 사항을 하나의 대시보드에서 일괄적으로 보여주어 관리자가 쉽게 상황을 판단할 수 있도록 모니터링 기능을 구성한다. 이를 통해 사용자의 편의성을 증대하고 서비스가 한 가지 클라우드에 종속되는 문제(Lock-in)를 해

결한다.

또한 한 클라우드에 문제가 발생하여 특정 CSP의 서비스 전체를 이용할 수 없는 경우에 대처할 수 있도록 한다. 하나의 클라우드에 시스템 장애가 발생하면, 정상 작동하고 있는 타사의 클라우드에 장애가 발생한 클라우드의 서비스를 복구하여 지속적으로 서비스가 정상 운영될 수 있도록 하는 기능을 추가한다. 예를 들어 AWS에 문제가 발생할 시 모니터링 툴이 이를 감지하고 GCP(Google Cloud Platform)에서 복구 기능을 수행한다.

결론적으로 해당 프로젝트에서는 CNCF(Cloud Native Computing Foundation)에서 인정한 쿠버네티스, 프로메테우스(Prometheus) 등의 어플리케이션을 활용하여 1. 클라우드 종속성에 구애받지 않는 모니터링 기능, 2. CSP문제로 클라우드 서비스 전체가 중단된 상태에서 다른 클라우드에 자동으로 복구하는 기능을 포함한 대시보드를 개발하고자 한다. II. 관련 연구에서는 본 연구의 핵심 기술인 멀티클라우드, 쿠버네티스에 대해 설명하고 기존 유사시스템에 대해 분석한다. III. 제안하는 시스템에서는 제안하는 시스템에 대한 구조를 설명하고, 프로젝트 결과물을 2개의 시나리오를 통해 실제로 테스트해보는 과정을 기술하였다. 마지막으로 결론을 통해 프로젝트 결과물의 의의에 대해 고찰하였다.

II. 관련 연구

2.1 멀티클라우드

멀티 클라우드는 가상 데이터 스토리지 또는 컴퓨팅 성능 리소스에 다른 프라이빗 클라우드 및 온프레미스 인프라 외에 2개 이상의 퍼블릭 클라우드 서비스 공급업체를 사용하는 것을 의미한다[3].

2.2 쿠버네티스

쿠버네티스는 컨테이너화된 어플리케이션의 자동 디플로이, 스케일링 등을 제공하는 관리시스템으로, 오픈 소스 기반이다. 원래 구글에 의해 설계되었고 현재 리눅스 재단에 의해 관리되고 있다[4].

2.3 유사시스템

2.3.1 오픈시프트(OpenShift)

오픈시프트는 컨테이너 기반 소프트웨어의 디플로이 및 관리를 위한 레드햇(Red Hat)의 컴퓨터 소프트웨어 제품이다. 구체적으로 말해, 가속화된 어플리케이션 개발을 위해 도커 컨테이너와 데브옵스 도구를 사용하는 것으로 쿠버네티스의 레드햇 버전이다[5].

2.3.2 넥스클리퍼(NexClipper)

넥스클리퍼는 국내 CNCF 멤버 회사 넥스클라우드에서 개발한 클라우드 네이티브 인프라스트럭처 모니터링 시스템이다. 쿠버네티스 및 하이브리드 환경에 대한 가시

성을 제공하는 관리형 프로메테우스 서비스로 대시보드, 수치 정보 추출을 위한 쿼리 서비스, 알림 기반 장애 관리, 설치 및 구성 자동화와 스케줄링 기능 등을 포함하고 있다[6].

2.4 유사 연구

2.4.1 클라우드 바리스타(Cloud-Barista)

클라우드 바리스타는 단일 클라우드 활용의 한계성을 극복하고 멀티 클라우드의 활용 확산을 위해 다양한 멀티 클라우드 인프라 서비스와 멀티 클라우드 애플리케이션의 운용 및 관리에 필수적인 공통 기반 기술과 누구나 손쉽게 사용할 수 있는 개방형 API를 제공한다. 사용자들은 이 플랫폼을 통해 멀티 클라우드 서비스 및 애플리케이션의 실행/관리에 공통적으로 필요한 기능들에 대한 중복 개발의 비효율성을 제거하여 생산성을 크게 향상할 수 있으며 다양한 산업 및 서비스 분야에서 멀티 클라우드 활용이 가능하다[7].

2.4.2 멀티 클라우드 기술

멀티클라우드 기술은 다수의 퍼블릭 클라우드 인프라를 연계하여 자원의 제약이 없는 최적의 컴퓨팅 자원을 활용할 수 있도록 하는 멀티 클라우드 인프라 기술과 구성된 멀티 클라우드 인프라에서 다양한 응용을 전 세계 원하는 곳으로 배치, 운용, 관리할 수 있도록 하는 멀티 클라우드 응용 기술로 크게 나눌 수 있다.

멀티 클라우드 인프라 사용 환경은 사용환경과 같이 사용자가 서로 다른 클라우드가 제공하는 이종의 API 및 제어 방법을 복합적으로 사용하여 필요한 클라우드 인프라 제어를 수행해야 하는 어려움이 있으며, 신규 클라우드의 추가는 더욱 복잡하고 통합 운영 관리에도 어려움이 있다[8]

III. 제안하는 시스템

3.1 제안하는 내용 및 목표

현재 시장에 나와있는 멀티 클라우드 관리 시스템은 자사가 개발한 관리 어플리케이션을 토대로 데이터를 수집하고 이를 대시보드로 보여주는 형태이다. 그러나 이러한 시스템들은 클라우드 장애 시 복구 방법을 제공하지 않으며, 유료로 서비스를 제공하는 경우도 있다. 또한 기업에서 제공하는 시스템이기 때문에, 역으로 모니터링 시스템이 벤더 종속성을 가진다는 문제가 있다.

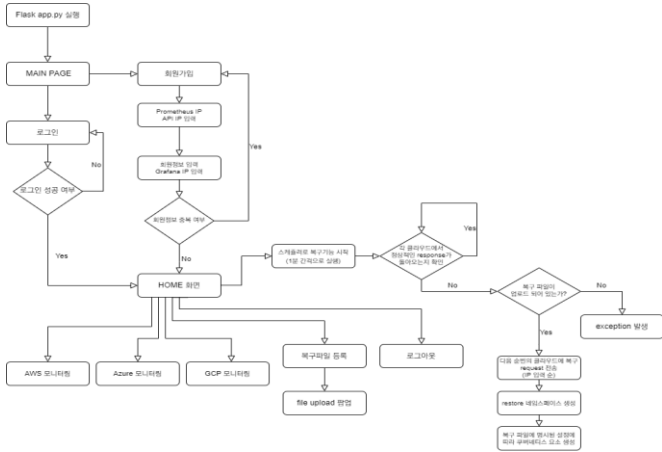


그림 1 복구 알고리즘 플로우 차트

본 프로젝트에서는 멀티 클라우드 환경을 쉽게 운영할 수 있도록 1) CSP에 구매받지 않고 여러 클라우드의 관리형 쿠버네티스를 모니터링할 수 있으며, 2) CSP 장애로 클라우드의 서비스가 운영 중단되었을 때, 정상적으로 작동하고 있는 다른 클라우드의 쿠버네티스에 중단된 서비스를 복구하여 클라우드 중단 상황에 대응할 수 있는 기능을 추가한 대시보드를 개발하고 이를 오픈소스로 배포하였다.

클라우드 사용 비용 문제로 클라우드 하나에 한 개의 k8s 클러스터가 존재하는 상황을 가정했다.

3.2 제안하는 시스템/모델 구현 방법

3.2.1 테스트 환경을 위한 AWS, AZURE, GCP 클러스터 구축 및 테스트용 웹 서비스(투표 서비스) 구축

AWS, Azure, GCP에 모니터링을 할 쿠버네티스 클러스터를 구축하였다[9][10][11]. 모니터링 기능과 시나리오를 테스트할 수 있도록 모든 클러스터에 Nginx와 Redis DB 기반의 간단한 투표 서비스, 복구 기능을 위한 API, 각 쿠버네티스 클러스터의 메트릭(metric) 데이터를 수집하는 프로메테우스를 올렸다.

1) 프로메테우스는 helm의 stable chart를 이용하여 설치했다[12]. 프로메테우스 서버를 구성하는 다양한 컴포넌트들에 대한 설정값들이 모여 있는 프로메테우스의 values.yaml을 편집하여 서비스가 LoadBalancer 타입으로 배포될 수 있도록 하였다[13].

그림2 투표 서비스

2) 그림2는 테스트용 웹 서비스로 사용한 투표 서비스로 수강신청 전 수요를 조사하는 상황을 가정했다. 투표 서비스 구축을 위해 docker-compose로 redis, azure-vote-front라는 이미지를 만들고 이를 컨테이너 레지스트리에 docker push했다. 해당 이미지의 경로를 yaml 파일에 입력 후 yaml 파일을 기반으로 클러스터에 배포했다[14].

3) API 복구 명령 전송 및 복구 수행

스케줄러가 클라우드의 작동중지를 감지하면 업로드 되어있던 yaml파일을 기반으로 쿠버네티스 자원 생성 요청을 정상 작동중인 클라우드로 보낸다. 모니터링 중인 클러스터에 위치한 플라스크 API는 이 request를 받아 복구 대상을 지정된 클러스터에 복구한다[15].

해당 과정을 수행하는 API를 docker로 패키징(packaging)하여 공개된 컨테이너 레지스트리에 업로드 하였고 이를 클러스터에 배포했다.

3.2.2 OCI 클러스터 구축

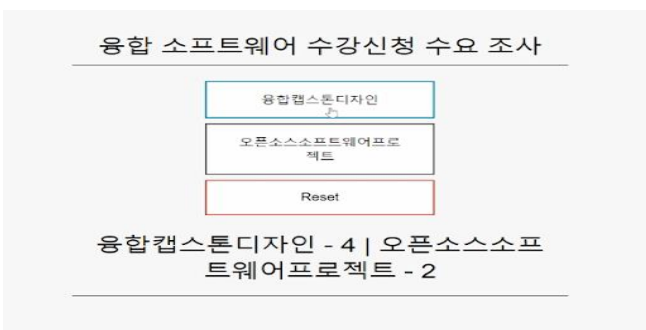
OCI에는 AWS, Azure, GCP에 대한 모니터링 기능을 수행하는 그라파나(Grafana)와 대시보드를 올렸다[16]. 그라파나는 AWS, Azure, GCP의 프로메테우스로 수집한 메트릭 데이터를 시각화한다. 그라파나 역시 프로메테우스와 같이 helm의 stable chart를 이용하여 설치했다[12]. 마찬가지로 values.yaml을 편집하여 세부 사항을 수정하여 배포될 수 있도록 하였다. 구체적으로는 프로메테우스 타입의 데이터소스(datasource)의 이름, url 주소를 지정하고, 이 데이터소스와 json 형태의 대시보드를 연결하여 배포 시 자동으로 데이터소스, 대시보드 설정이 완료되도록 구성하였다. 또한 외부에서 iframe을 통해 접속이 가능하도록 auth.anonymous 항목을 enabled: true로 변경하였다[17]. 마지막으로 복구기능을 수행할 수 있도록 AWS, AZURE, GCP의 Flask와 OCI의 Flask가 http 통신할 수 있도록 했다.

3.2.3 멀티클라우드 모니터링 시스템 설계

AWS, Azure, GCP의 프로메테우스에서 수집한 메트릭 데이터를 OCI의 Grafana로 시각화했고 시각화된 패널들을 iframe으로 가져와서 대시보드 웹 페이지로 구현했다[18].

회원가입 과정에서 모니터링할 클라우드의 프로메테우스 및 API의 외부 ip 주소를 입력 받아서 cloud_info.json과 grafana-values.yaml에 반영하여 클라우드의 사용 현황과 정상 작동 여부를 모니터링하고, 복구 명령을 내린다. 또한 그라파나 외부 ip를 입력 받아 해당 그라파나로부터 메트릭 데이터를 시현하는 패널을 받아오도록 했다.

AWS, Azure, GCP 클라우드의 클러스터 정보를 각 클



라우드 탭에서 확인할 수 있다. 파드 현황 탭에서는 쿠버네티스 내 파드 개수, 네임스페이스 개수, 전체 파드 가동률을, 노드 현황 탭에서는 노드 이름, 생성 후 경과 시간, OS를 비롯한 노드 관련 정보들을, 네트워크 탭에서는 CPU 사용률, 메모리 사용량, 네트워크 I/O 및 API 서버 응답량을 확인할 수 있다.

3.2.4 장애 복구 시스템 설계

모니터링 할 쿠버네티스 클러스터에 구성된 API로 recover 명령을 받을 수 있도록 했다. OCI의 대시보드는 스케줄러를 통해 지속적으로 모니터링하는 쿠버네티스 클러스터가 정상적으로 작동하는지 request를 보내 정상적으로 접속되는지 여부를 확인하여 health check를 한다. 이 때 클러스터의 장애가 감지되면 자동으로 정상 작동중인 클라우드 내 쿠버네티스 클러스터에 API를 이용하여 복구 request를 보내 미리 업로드한 yaml 파일을 바탕으로 서비스를 복구한다[15]. 복구가 이루어지는 클라우드는 회원가입 과정에서 IP를 입력한 순번으로 바로 다음 순서의 클라우드이다. 이 과정에서 문제가 발생한 클라우드는 모니터링 대상에서 제외된다.

3.3 실험방법 및 결과

3.3.1 대시보드의 기본 기능 테스트

1) 회원가입

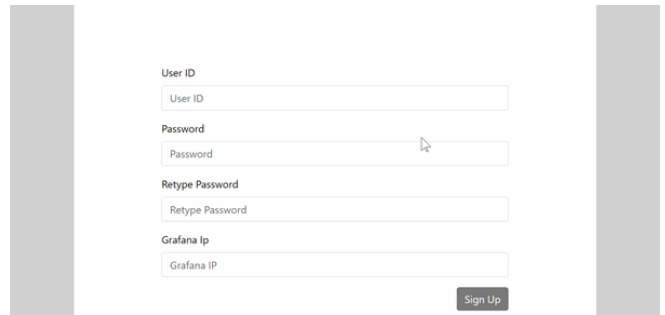


그림 2, 5 회원가입

사용자는 대시보드에 접속하여 회원가입과 로그인을 진행한 후, 대시보드 탭에서 해당 클라우드의 현황을 확인할 수 있다.

대시보드에 처음 접속한 사용자는 서비스 이용을 위해 회원가입을 진행해야 한다. 그림 3은 로그인 전 대시보드 메인 화면이다. SIGN UP 버튼을 눌러 회원가입을 한다. 회원가입은 총 두 단계로 이루어져 있다. 먼저 그림 4,5와 같은 창에서 각 클라우드에 설치된 프로메테우스 서버와 플라스크 API의 서버의 IP를 입력한다. 이 과정은 대시보드가 클라우드의 사용 현황과 정상 작동 여부를 모니터링하고, 복구 명령을 내리기 위해 필요하다. 다음으로 사용자 ID와 비밀번호, 사용자의 그라파나 IP를 입력하여 회원가입을 완료한다.

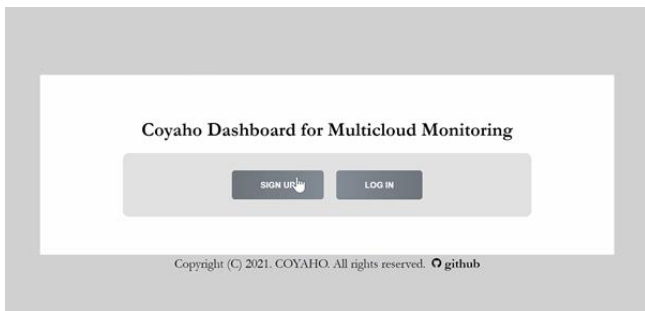


그림 1 로그인 전 메인 화면

2) 로그인 및 대시보드 열람



그림 3 로그인 후 메인 화면

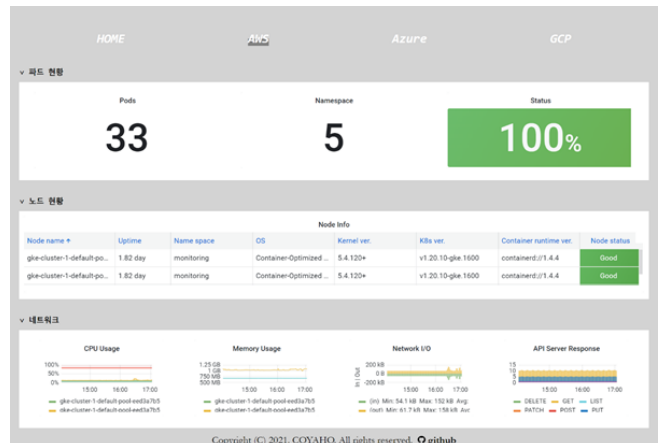
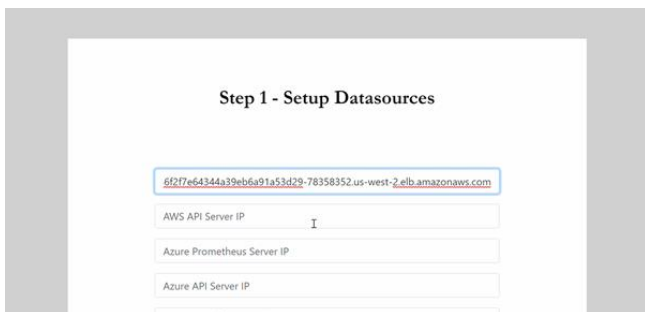


그림 4 클라우드 모니터링 탭

회원가입을 완료 후 로그인을 진행하면 그림 6와 같이 파일 등록 버튼과 4개의 탭 (HOME, AWS, Azure, GCP) 로 접속할 수 있다. 그림 7는 AWS 탭에 접속하여 해당 클라우드를 모니터링하는 모습이다. 각 탭에서는 파드 현황, 노드 현황, 네트워크 정보를 확인할 수 있다.

3.3.2 부하 모니터링

시나리오 1

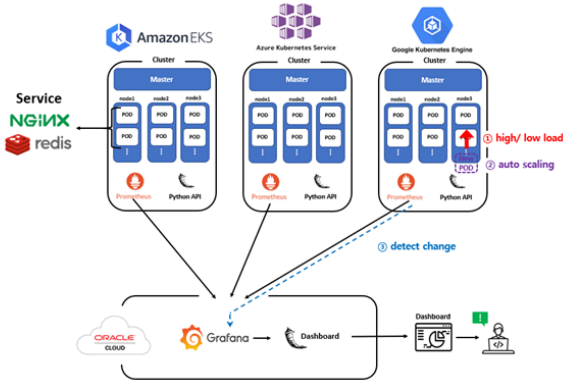


그림 5 부하 모니터링 아키텍처

그림 8과 같이 대시보드의 부하 모니터링을 테스트한다. 만일 서비스에 많은 사람이 몰린다면, 쿠버네티스의 오토 스케일링 기능이 작동하여 자동으로 파드의 개수가 증가한다. 또한 CPU 사용량, 네트워크 I/O 등 네트워크 트래픽이 증가한다. 따라서 임의로 쿠버네티스 클러스터에 부하를 주어 클러스터의 변화가 정상적으로 모니터링 되고 있는지 테스트한다.

그림 8에서 각 단계별 설명은 다음과 같다.

- ① 임의로 부하를 생성하는 로드 제너레이터(Load Generator)를 사용하여 특정 쿠버네티스 클러스터에 부하를 증가시킨다[19].
- ② 쿠버네티스는 자동으로 부하가 늘어남에 따라 파드 수를 증가한다(Auto Scaling).
- ③ 프로메테우스는 증가된 파드 수를 비롯한 여러 metric 데이터들을 감지하여 OCI의 그라파나로 전달한다.
- ④ 그라파나는 전달받은 Pod 개수, Node 개수, Network 트래픽, CPU 사용량 등의 메트릭 데이터를 그래프로 시각화하고 플라스크로 구성된 대시보드 웹 페이지에 전달한다.
- ⑤ 최종적으로 시스템 사용자는 대시보드를 통해 부하 증가로 인한 변화를 확인할 수 있다.

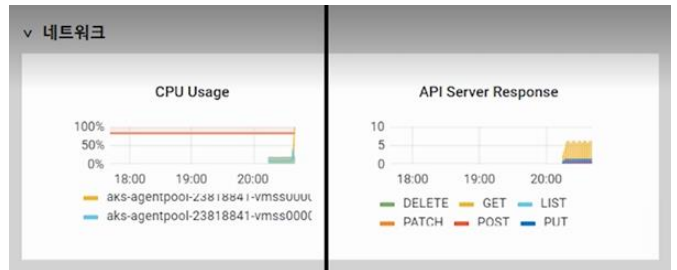
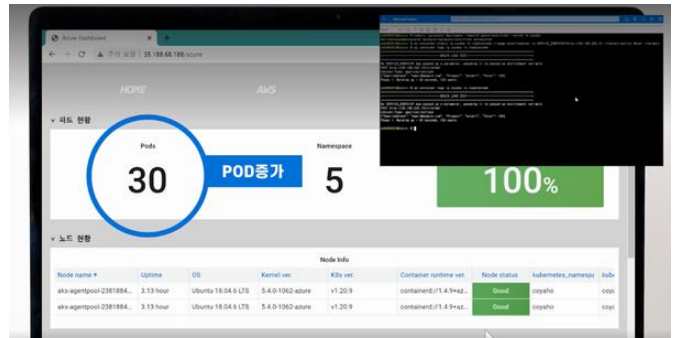
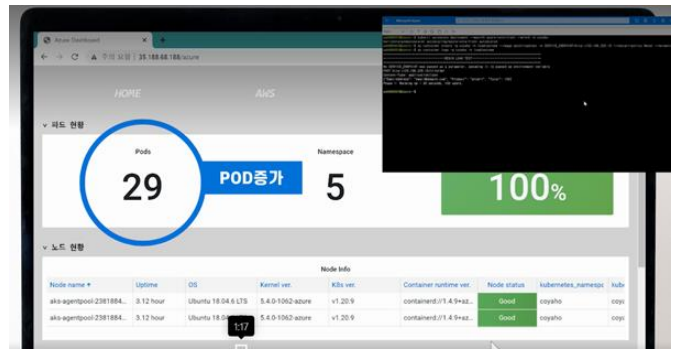
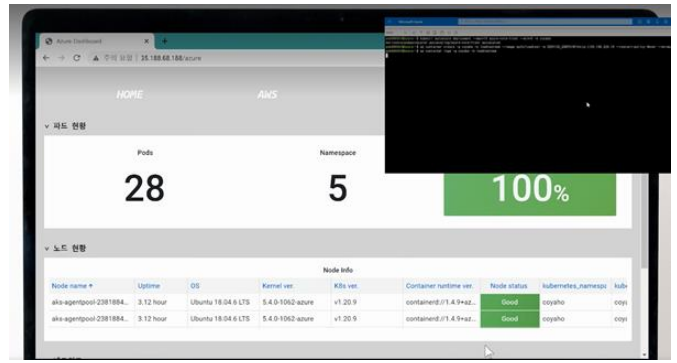


그림 6~12 부하 모니터링 테스트 화면

그림 9에서 파드가 28개인 클라우드에 로드 제너레이터를 이용하여 100명이 접속한 것과 같은 부하를 주었다. 그 결과로 그림 10처럼 파드가 자동으로 하나 증가하였고, 이를 대시보드에서 확인할 수 있다. 30초 뒤, 추가로 100명의 부하를 주자 그림 11과 같이 파드가 30개로 증가하였다. 그림 12에서의 CPU 사용량과 API 서버 응답을 확인하여 대시보드가 정상적으로 클러스터의 변화를 감지하고, 시각화하여 보여준다는 것을 알 수 있다.

3.3.3 장애 복구 및 클라우드 간 스케일링

시나리오 2

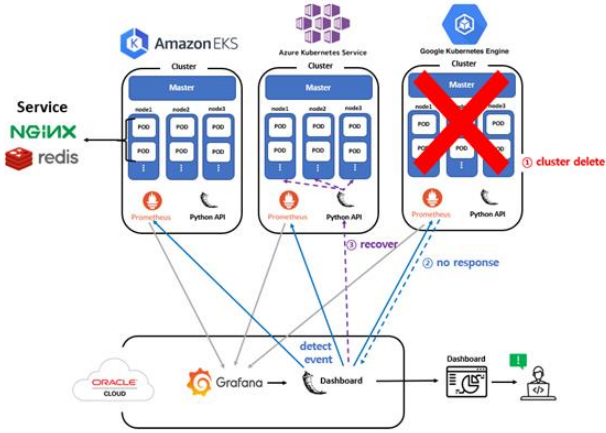


그림 7 복구 및 클라우드 간 스케일링 아키텍처

그림 13과 같이 대시보드의 장애 복구 및 클라우드 간 스케일링 기능을 테스트한다. 클라우드 리전 장애가 발생하면 해당 위치에 존재하는 클러스터는 갑작스럽게 작동하지 않게 된다. 대시보드는 API 통신을 통해 계속 클라우드의 작동 여부를 감지하고 있다가, 클라우드의 작동이 멈추면 다른 클라우드로 복구 요청을 보낸다. 결과적으로 장애가 일어난 지역에 있던 파드의 수만큼 다른 클라우드의 파드 수가 증가하기 때문에, 사용자 과부하의 문제없이 서비스를 계속해서 운영할 수 있도록 해준다. 이 기능을 테스트하기 위해 3개의 클라우드 중 하나의 클라우드에 있는 클러스터를 삭제하여 장애가 발생한 상황과 비슷한 상황을 연출하고, 그 결과를 확인한다.

그림 13에서 각 단계별 설명은 다음과 같다.

- ① 클라우드 장애로 특정 클라우드의 모든 서비스가 중단되는 상황 발생을 가정한다. 쿠버네티스 클러스터를 삭제하여 해당 상황을 시뮬레이션한다.
- ② 스케줄러는 특정 클라우드에서 정상적으로 response가 돌아오지 않거나 timeout이 발생할 경우 장애 발생을 감지한다[20].
- ③ 스케줄러[20]는 정상적으로 작동중인 다른 클라우드에 복구 request를 전송한다. 업로드 되었던 yaml 파일을 기반으로 파드를 생성하여 총 파드개수를 유지하여 손실없이 작동할 수 있도록 한다.

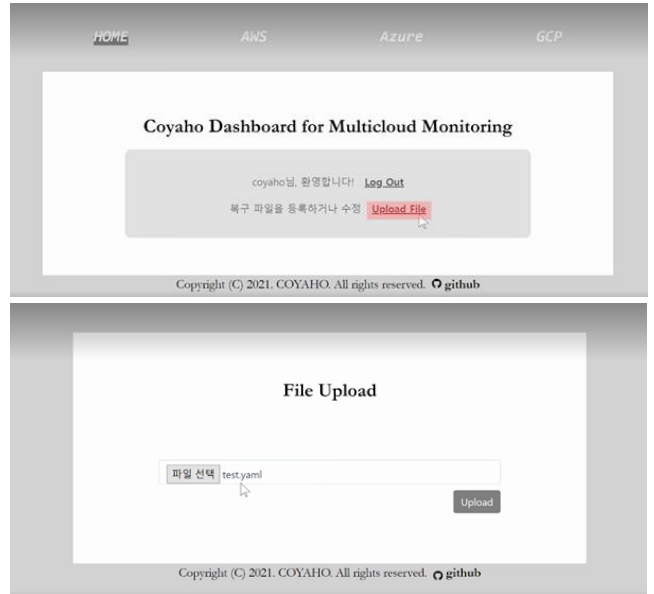


그림 14, 15 복구 파일 업로드

대시보드의 복구 기능은 yaml 확장자의 configuration 파일을 기반으로 작동하기 때문에, 사용자가 configuration 파일을 미리 대시보드에 업로드한다. 그림 14의 홈 화면에서 upload file을 누르면 그림 15의 file upload 페이지로 이동하여 해당 작업을 진행할 수 있다.

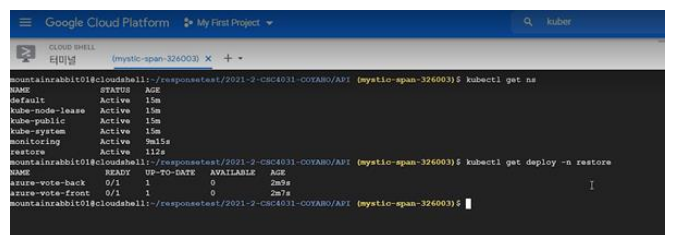
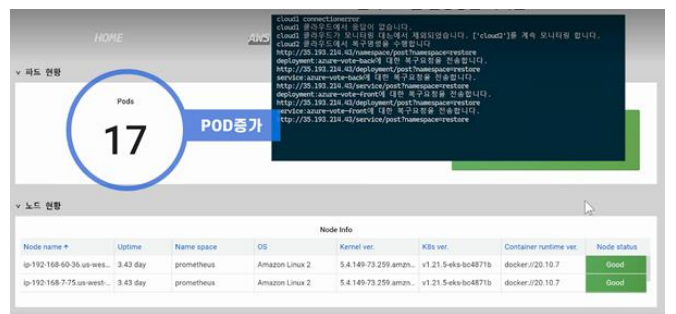
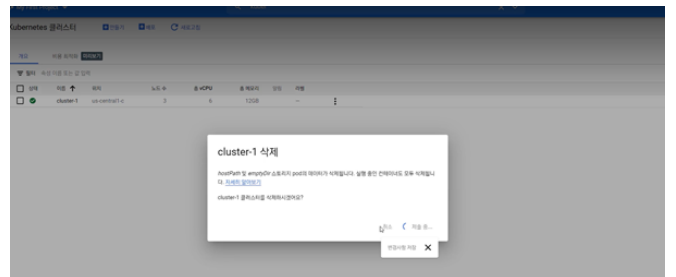


그림 16~8 클러스터 삭제 후 복구 모니터링
다음으로 GKE의 클러스터 하나를 삭제하여 실제로 복

구가 진행되는지 테스트했다. 모의 클라우드 장애 상황을 만들기위해 그림 16과 같이 클러스터를 삭제하고 터미널 창을 확인하면, 클라우드1에 응답이 없다는 메시지와 함께 클라우드2에 복구가 진행됨을 알 수 있다. 또한 그림 17과 같이 클라우드2 모니터링 페이지에서 파드 수가 증가한 것을 통해 복구가 완료되었음을 알 수 있다. 복구가 완료된 후 클라우드 터미널에 접속하여 클라우드2에 정상적으로 복구되었는지 확인해보면 그림 18과 같이 restore라는 네임스페이스가 새롭게 생성되었고, 그 안에 삭제된 클러스터의 파드와 동일한 디플로이먼트와 서비스가 생성되었음을 확인할 수 있다. 따라서 복구 기능도 정상적으로 작동함을 알 수 있다.

IV. 결 론

멀티클라우드는 클라우드 장애 발생에 유연하게 대처할 수 있다는 장점이 있다. 반면 단일 클라우드에 비해 서비스 구조의 복잡성이 증가하므로 관리가 어렵다. 본 프로젝트의 멀티클라우드 모니터링 시스템은 구현한 대시보드를 이용하여 여러 클라우드 상의 클러스터의 변동 사항을 실시간으로 확인할 수 있고, 장애로 인해 특정 클러스터가 삭제되더라도 서비스를 자동으로 복구하여 서비스의 정상 작동을 지원하여 개발자의 멀티클라우드 서비스 운영을 돕는다.

현재 클라우드 모니터링 서비스 시장은 아직 점유율이 높은 시스템이 없는 블루오션이다. 특히, 코로나 19 바이러스의 확산으로 인해 비대면 업무, 회의 등 기업의 업무 환경이 디지털로 변화되면서 클라우드 도입을 통한 디지털 트랜스포메이션을 목표로 하는 사례가 증가하고 있다. 이러한 상황 하에서 복잡한 멀티클라우드 기반 시스템을 한 눈에 모니터링 할 수 있는 모니터링 시스템을 공개함으로써 많은 기업에게 운용상 이점을 제공할 수 있다. 멀티클라우드를 기반으로 서비스를 구축하여 사용하고 있는 고객이나, 멀티클라우드 기반으로 마이그레이션을 기획 중인 고객이라면 누구나 쉽게 해당 모니터링 시스템을 사용할 수 있다. 오픈소스로 코드를 공개하고 무료로 시스템을 제공하기 때문에 유료 서비스만 존재하는 멀티클라우드 모니터링 시스템 시장에 새로운 Game Changer가 될 것으로 기대한다.

향후 본 서비스에서 대시보드의 이벤트 탭을 생성하면 사용자 편의성을 높여 더욱 편리할 것이다. 또한 사용자가 yaml 파일을 직접 올리지 않고 클러스터의 상태나 설정 정보를 저장하는 쿠버네티스의 etcd를 이용하여 클러스터를 복구할 수 있도록 개선할 수 있다.

ACKNOWLEDGMENT

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학지원사업의 연구결과로 수행되었음 (2016-0-00017)

참 고 문 헌

[1] AWS 도쿄 리전 장애 발생, 특정 AZ 일부 구역의 온도 상승이 원인
(<https://www.44bits.io/ko/post/news--2021-02-20-aws-ap-northeast-1-outage>)

[2] AWS 대규모 접속장애... '단일 클라우드' 사용한 고객사 피해 커(종합)
(http://it.chosun.com/site/data/html_dir/2018/11/22/2018112201730.html)

[3] 멀티 클라우드
<https://en.wikipedia.org/wiki/Multicloud>

[4] Kubernetes <https://kubernetes.io/>

[5] 오픈시프트 openshift.com

[6] 넥스클리퍼 <https://nexclipper.io/>

[7] 멀티 클라우드 서비스 공통 플랫폼 설계 및 구현 (김수영 외 5인) - 2021.01 한국멀티미디어학회논문지

[8] 멀티 클라우드 기술 개요 및 연구 동향(김병섭 외 8인) - 2020년 과학기술정보통신부

[9] AWS
<https://docs.aws.amazon.com/>
<https://github.com/237summit/Kubernetes-on-AWS/commit/8ea59409d6078a88ce28574cf601390a46a074e0>

[10] Azure
<https://docs.microsoft.com/azure>
<https://docs.microsoft.com/ko-kr/azure/aks/kubernetes-walkthrough-portal>

[11] GCP
<https://cloud.google.com/docs>
<https://dreamgonfly.github.io/blog/creating-gke-cluster/> <https://cloud.google.com/sdk/docs/install>

[12] Helm Stable Chart
<https://github.com/helm/charts>

[13] Prometheus <https://prometheus.io/>

[14] Azure Voting App
<https://docs.microsoft.com/ko-kr/azure/aks/tutorial-kubernetes-prepare-app>

[15] kubernetes API
<https://github.com/kubernetes-client/python/>

[16] OCI <https://docs.oracle.com/>

[17] Grafana <https://grafana.com/>

[18] Grafana Share a panel
<https://grafana.com/docs/grafana/latest/sharing/share-panel/>

[19] Load Generator
https://www.youtube.com/watch?v=-kVXhaf_as8&t=1s

[20] flask-apscheduler
<https://github.com/viniciuschiele/flask-apscheduler>

부 록
팀 깃허브

<https://github.com/CSID-DGU/2021-2-CSC4031-COYAH0>

아동·청소년을 위한 인공지능 스피커: 심리 상담 통합 시스템 개발

김수아^{*1}, 이예빈^{*}, 정수민^{*}, 최미소^{*0}, 최혜빈^{*}

이화여자대학교 휴먼기계바이오공학부

sua917@ewhain.net, jeyb1222@naver.com, sumillion@ewhain.net, miso8070@ewhain.net, hyebly9117@gmail.com

Integrated Counseling System for child

Sua Kim^{*}, Yebin Lee^{*}, Soomin Chung^{*}, Miso Choi^{*0}, Hyebin Choi^{*}

Division of Mechanical and Biomedical Engineering, Ewha Womans University

요 약

아동·청소년의 우울증 문제와 마음 챙김의 중요성은 지속적으로 강조되고 있다. 그러나 도움을 받을 수 있는 기관의 부재와, 아동·청소년의 상담에 대한 부정적 정서로 인해 해결이 어려운 실정이다. 이에 본 논문에서는 아동·청소년을 위한 심리 상담 통합 시스템을 제안한다. 본 논문이 제안하는 심리 상담 통합 시스템은 하드웨어 부분과 소프트웨어 부분으로 구성되어 있다. 스피커의 하드웨어는 라즈베리 파이 3B+ 모델을 사용하여 제작되었으며, 외관의 완성도를 높이기 위해 3D 프린터로 케이스를 제작하였다. 소프트웨어 부분은 딥러닝 기반의 자연어 처리 모델, 음성 합성 모델, 감정 분석 모델을 포함한다. 모델 학습에 필요한 문답 페어 데이터 셋과 연예인 음성 데이터 셋은 전처리 과정을 거쳐 직접 구축하였으며, 추가로 오픈 데이터 셋을 활용하였다. 최종적으로 완성된 시스템은 내담자가 인공지능 스피커에 고민을 얘기하면 스피커가 그에 합당한 위로와 조언을 건네주는 것부터, 스피커의 유명한 목소리 합성, 웹 애플리케이션을 통한 내담자의 전체 발화 내용 기반의 감정 분석 리포트 확인까지의 통합을 이루고 있다. 타겟 이용자인 아동·청소년 58 명을 대상으로 설문조사를 진행한 결과, 전체 시스템에 대한 만족도 81.7%를 보였다. 따라서 최종 시제품은 아동·청소년의 상담 진입장벽을 낮추고, 친근한 목소리를 사용하여 편하게 감정 상태를 노출할 수 있게 도우며, 입체적인 감정 분석 보고서를 제공하고자 한 본 논문의 목표를 달성하였음을 보인다.

1. 서 론

질병관리청의 통계에 따르면, 19 년 기준 아동·청소년의 평균 우울감 경험률은 28.4%로 10.3%인 성인에 비하여 약 3 배 이상 높은 수치를 보인다. 18 년도 중·고등학생 스트레스 인지율의 경우, 39.9%에 달하여 전국 10 명의 청소년 중 4 명의 청소년이 스트레스를 받고 있다고 느끼는 것을 알 수 있다[1]. 마지막으로 18 년도 기준 최근 8 년간 아동·청소년의 연도별 사망원인 1 위는 고의적 자해로, 국내 아동·청소년들이 정서적 고통과 우울감에서 자유롭지 못함을 알 수 있다[2]. 이렇게 아동·청소년의 마음 챙김과 우울증에 관한 문제는 지속해서 문제가 되고 있으나 아동·청소년들은 성인보다 자발적으로 도움을 받을 수 있는 기관을 찾기 어려우며, 아동·청소년들이 상담에

대해 부정적인 견해를 가지기 때문에 도움을 받기 더욱 힘든 상황이다. 학교급별 전체 전문상담교사 배치 학교 비율은 2005 년부터 지속적으로 증가 추세를 보이는 중이나, 20 년도 기준 평균 35.5%로, 전국의 절반 이상의 학교에서 학생들이 고민이 생겼을 때 전문적인 상담을 받을 수 있는 교사가 부재한 것을 알 수 있다. 상담교사와 동일한 비교과 교사인 보건교사, 영양교사와 비교하였을 때 17 년도 기준 보건교사 배치율은 64.5%, 영양교사는 41.3%, 상담교사는 15.6%로 상담교사의 비율이 현저히 낮은 것을 확인할 수 있다[3].

앞서 말했듯이 청소년이 상담 자체에 대해 부정적인 정서를 가진 것도 한 문제이다. 서울 경기지역 청소년 433 명을 대상으로 한 설문조사에서 ‘상담 자체에 대한 반응 유목 설문 결과’를 보면, ‘불쾌 거부감’이 15.6%로 제일 높은 비율을 보였으며, ‘상담실에 대한 이미지 반응 유목 설문’의 경우 ‘문제 있는 사람이 가는 곳’이 13.6%로 제일

¹ All authors contributed equally.

높은 비율을 보였다. 이렇게 청소년이 상담에 대해 부정적으로 생각함을 알 수 있었고, 부정적인 정서의 요인 파악을 위한 ‘상담 방해 요인 반응 유목 설문’ 결과를 보면 ‘낙인(창피)’가 22.6%로 제일 높은 비율을 보였고, ‘개방의 두려움’이 16.2%로 그다음을 차지했다. ‘상담실에 대한 편견’ 8.2%, ‘비밀 누설’ 5.1% 등이 그 뒤를 따랐다. 아동·청소년들은 또래 친구들의 시선과 평가에 민감하고, 상담 자체에 대해 막연한 편견을 갖고 있기에 상담에 대하여 진입장벽을 높게 느끼는 것을 알 수 있다[4].

이를 해결하기 위해 인공지능을 활용한 상담 매체들이 등장하고 있다. 대표적으로 모바일 심리 상담 플랫폼 ‘트로스트’에서 제공하는 마음 관리 챗봇 서비스 ‘티티’가 있다. 이는 챗봇과 심리 상담을 진행한 후 내담자의 감정 상태에 관한 결과가 모바일 리포트로 전달되어지는 형태의 서비스이다. 이러한 스마트 기기를 활용한 대화는 우울 감정 해소에 도움이 되며[5], 접근성이 용이하고 익명성이 보장된다는 장점으로 인공지능 기반 상담 서비스는 사용자들로부터 긍정적인 피드백을 얻었다[6]. 하지만 텍스트로만 진행되기 때문에 감정 분석 리포트 생성 시 내담자의 반언어적 표현 반영이 어렵고, 서비스 이용 시 1 초도 안되는 짧은 시간 내에 답변이 오는 형태 때문에 일방적인 소통처럼 느껴진다는 피드백이 존재하였다[6].

따라서 본 논문은 오프라인 상담의 한계와 더불어 기존 인공지능 상담 서비스의 한계를 개선한 ‘인공지능 스피커를 활용한 아동·청소년 대상 심리 상담 시스템’을 개발하였다. 기존에 존재하는 정신건강과 관련된 인공지능 스피커로는 국내 SK 텔레콤의 ‘누구’와 해외 Amazon 사의 ‘Alexa’가 존재한다. ‘누구’가 제공하는 마음 챙김 서비스인 ‘누구 마음보기’의 경우, 상담 형식이 아닌 단순 명상 서비스에 그친다. ‘Alexa’가 제공하는 mindscape의 경우, 사용자의 감정을 파악하고 이에 대한 해결 방안을 제시하지만, 서비스가 지원되는 언어가 영어뿐이기에 국내에서의 사용에는 한계가 있다.

이렇게 유사 서비스의 한계점까지 보완하고자 한 본 논문의 핵심적인 목표는 다음과 같다. 1. 상담실에 가지 않아도 적절한 답변을 제공하는 스피커를 통해 상담을 진행할 수 있기에 주변 시선을 의식하여 상담실 방문을 꺼리는 청소년들에게 상담에 대한 진입장벽을 낮춘다. 2. 상담 시 아동·청소년이 좋아하는 유명인, 캐릭터의 목소리를 사용하여 친근감을 느끼고 개인의 감정 상태를 편하게 노출할 수 있게 돕는다. 3. 발화로 진행되는 구조로 내담자가 쌍방향적 소통이 진행되고 있음을 느낄 수 있을 것이며, 감정 분석 리포트 생성 시 보다 입체적인 감정 분석을 한다.

2. 본문

2.1. 설계 목표

본 논문에서는 아동·청소년의 정신 건강 문제를 해결하기 위해 인공지능 심리 상담 스피커를 선택했다. 아동·청소년의 정신 건강을 증진시키기 위해서는 상담이 필요한데, 아동·청소년의 상담 자체에 대한 기피와 상담 내용 누설에 대한 두려움을 해결하기 위해서는 접근성이 좋아야 하며 상담 내용 비밀 보장이 중요하기 때문에 사람이 아닌 인공지능이 적합할 수 있다. 실제로 상담자가 컴퓨터라는 것을 인지할 때 두려움과 압박감이 낮기 때문에 객관적인 감정 표현을 수월하게 할 수 있다는 사실은 증명되어 있다[5]. 이런 문제를 해결하기 위해 인공지능 상담 챗봇이 상용화되어 있지만, 텍스트에만 의존하기 때문에 상담자의 심리 상태를 분석하기 어려우며 즉각적으로 답변이 나오는 일방적인 소통으로 사용자는 대화를 나눈다는 느낌을 받기 힘들다. 또한, 자기 이해는 정서 안정 및 정신 건강에 있어 매우 중요한 역할을 한다. 본인이 어떤 상황으로부터 어떤 감정을 가지고 있는지 이해하는 능력은 내적 동기와 자기존중감을 향상시켜 긍정적인 정서로 이어진다. 이에 청소년의 자기 이해를 돕고자 객관적으로 내담자 본인의 상황을 파악할 수 있도록 감정 분석 기능 추가를 고려하였다.

아동·청소년 심리 상담 스피커를 제작하기 위해서는 상담 데이터 셋으로 학습되어 심리 상담사의 답변을 내출 수 있는 자연어 처리 모델, 사용자가 거부감을 느끼지 않게 친근한 목소리를 낼 수 있도록 음성을 합성해 출력하는 음성 합성 모델이 필요하며, 감정 분석 기능을 구현하기 위해서는 감정 분석 모델 및 결과 출력용 웹 애플리케이션 제작이 필요하다.

본 논문에서 제안하는 스피커는 상담 고민을 입력으로 받았을 때 공감과 격려를 답변으로 출력하는 자연어 처리 모델을 필요로 한다. 학습 시 활용할 상담 데이터 셋의 고민 글과 답변 글의 길이가 길기 때문에, 자연어 처리 모델은 문장 끝과 끝의 단어들 간의 관계까지 잘 파악할 수 있는 모델이어야 한다. 또한 빠른 속도로 모델을 학습시켜 여러 번 결과를 비교한 후 데이터의 전처리와 모델의 하이퍼파라미터를 조절을 통해 최적화할 수 있어야 한다.

기존의 인공지능 스피커는 단순히 나긋나긋한 여성, 혹은 친절한 남성의 목소리를 낼 뿐이어서 아동·청소년이 친근감을 느끼기 어렵다. 본 논문에서는 아동·청소년이 상담에 친근감을 느끼게 하기 위해서 유명인의 음성을 합성하는 방법을 선택했다. 음성 합성 모델은 실시간으로 음성 합성이 가능하면서 성능이 불안정하지 않아야 한다.

감정 분석 모델의 경우, 멀티 모달(multimodal) 데이터 셋을 바탕으로 한 모델을 필요로 한다. 감정 표현에는 발화 내용뿐만 아니라 억양, 높낮이 등 반언어적인 표현도 상당한 비중을 차지한다. 단순 텍스트만을 분석하기보다 학습 데이터에 음성 정보도 추가한다면

모델이 학습하는 정보량이 증가하기 때문에 더욱 정밀한 감정 분석이 가능하다. 따라서 내담자의 감정 분석에 적용할 모델은 멀티 모달 데이터 셋을 활용하되, 오디오와 텍스트를 유기적으로 연결하여 최종 감정을 예측해 줄 수 있어야 한다.

2.2. 설계 방법

본 논문은 하드웨어 부분과 소프트웨어 부분으로 구성되어 있다. 소프트웨어 부분에서는 자연어 처리 모델, 음성 합성 모델, 감정 분석 모델 총 세 가지 모델로 구성되어 있다.

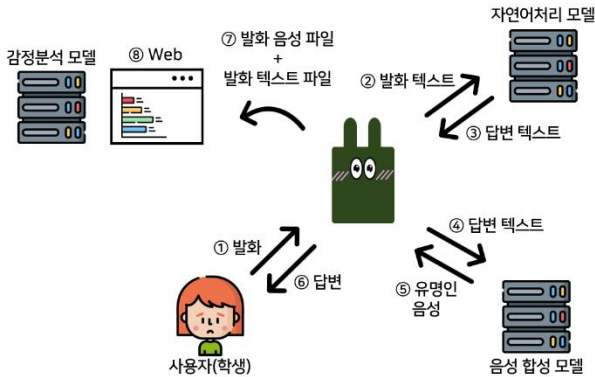


그림 1. 전체 프로세스

2.2.1. 하드웨어

스피커의 하드웨어는 라즈베리 파이 3B+ 모델을 사용하여 제작하였다(그림 2). 본 논문에서는 총 3 가지의 인공지능 모델을 사용하므로 와이파이를 통해 각 로컬에서 삼바 서버를 공유해야 한다. 또한, 음성 인식 및 음성 합성 결과 출력 등 운영체제 내에서 직접 프로그래밍하는 과정이 필요하다. STT 를 위한 음성인식 모듈을 사용하기 위해 마이크 보드 및 스피커를 보이스 실드에 연결하였으며 이를 통해 내담자의 발화 음성은 아날로그 신호에서 디지털 신호로 변환된다. 변환된 데이터는 GPIO 로 연결된 라즈베리 파이로 전송되어 자연어 처리, 음성합성 등의 소프트웨어 내부 작업을 거치게 된다.

본 논문의 이용 대상이 아동·청소년인 만큼 아기자기한 외형을 통해 사용을 독려하고자 토끼 모양을 본뜬 스피커 외관을 3D 프린터로 제작하였다(그림 2). 스피커 본체, 스피커 덮개, 한 쌍의 토끼 귀 모양을 포함해 총 4 개의 파트로 설계하였으며 내담자의 목소리가 마이크 보드에 원활하게 전달될 수 있도록 덮개 상단에 작은 구멍을 여러 개 추가하였다. 스피커는 잦은 이동을 필요로 하지 않고 변형이 일어날 만큼의 높은 열이 발생할 우려가 없기 때문에 가격이 저렴하고 편리한 PLA 필라멘트를 사용하여 출력하였다.

2.2.2. 소프트웨어

1) 자연어 처리 모델

자연어 처리 모델 학습을 위한 데이터 셋은 직접 구축한 지식 iN 자연어 처리 모델 학습을 위한 데이터 셋은 직접 구축한 지식 iN 데이터 셋과 오픈 데이터로 구성하였다. 지식 iN 서비스를 통해 이루어지고 있는 질문과 답변은 아동·청소년의 다양한 고민과 전문가의 답변을 동시에 수집할 수 있다는 점에서 본 논문의 상담 서비스에 적합한 데이터 셋이다. 심리 상담 질문과 정신건강의학과 권순모 전문의 외 5 명의 전문가 답변, 고민 Q&A 파트 질문과 여성가족부의 답변을 문답 페어로 구축하였고, selenium 라이브러리를 사용하여 크롤링 하였다. 인사말, 끝인사와 같은 불필요한 문장과 '~님'과 같은 특정 글씨를 지칭하는 말, 이모티콘과 같은 특수문자는 모두 제거하고, 맞춤법, 띄어쓰기 등의 오류를 수정하였다. 지나치게 답변이 긴 경우, 하나의 답변을 문단 단위로 나누어 하나의 고민 글에서 질문과 답변 페어가 총 N 개가 생성되도록 하였다. 추가적으로, 대화체의 오픈 데이터인 AI hub 가 제공하는 신촌 세브란스 병원 정신 건강 상담 기록으로 구성된 웰니스 대화 스크립트 데이터 셋² 문답 페어 1023 개와, 일상대화를 담고 있는 챗봇 데이터셋³ 문답 페어 11876 개를 사용하였다. 결과적으로 86503 개의 문답 페어로 구성된 학습 데이터 셋을 구축하였다.

자연어 처리 모델은 2.1 에서 기술한 요구 조건을 충족시키기 위해 Transformer 로 선정하였다. 자연어 처리 모델의 기본인 RNN 은 직전의 출력 결과를 입력으로 받아들여 가까운 단어끼리 연관성이 높게 나타나는 장기 의존성 문제가 있고 LSTM 은 멀리 있는 단어의 영향을 전달할 수 있지만 연산량이 많아 학습 속도가 느리다. 하지만, Transformer 는 RNN 을 통해 계산한 값을 어텐션(attention)을 이용해 동적으로 인코더(encoder)의 모든 상태 값을 반영하여 더 빠르고, 좋은 성능을 보인다는 장점이 있다.

2) 음성 합성 모델

음성 합성 모델의 데이터 셋으로는 유튜브 영상 32 개에서 추출한 유명한 음성을 사용하였다. 음성을 3 초 단위로 자르고, 숨소리, 긴 공백을 모두 제거하였으며, 그에 상응하는 텍스트를 함께 추출하였다. 전처리 후 2.85 시간의 데이터 셋을 구축했으나 음성을 합성하기에는 부족했다. 이를 해결하기 위해 12 시간 분량의 여성 성우 음성 데이터인 KSS 데이터 셋[8]을 함께 사용해 데이터를 보충하였다.

본 논문의 음성 합성 모델로는 Multi-speaker Tacotron2 를 선택하였다[9]. 현재 한국어 음성 합성 모델은 Tacotron2 와

² <https://aihub.or.kr/opendata/keti-data/recognition-laguage/KETI-02-006#>

³ https://github.com/songys/Chatbot_data

FastSpeech2 [10]가 주로 사용된다. FastSpeech2 는 LJSpeech 데이터 셋에서 MOS 3.83 이라는 높은 성능을 가지고 있으며 다른 Transformer 기반의 TTS 모델에 비해 학습에 필요한 시간이 절반이라는 장점이 있다. 하지만 음성과 그에 따른 스크립트 텍스트 외에 phoneme-utterance 얼라인먼트(alignment) 데이터가 추가적으로 필요하기 때문에 데이터를 직접 구축해야 하는 본 논문에 적용하기에는 어려움이 있다. 이에 비해 Tacotron2 는 추가적인 데이터가 필요하지 않으며, MOS 또한 3.70 으로 우수한 성능을 보여 본 논문의 음성합성 모델로 선택했다.

Tacotron2 에서 생성된 멜-스펙트로그램(mel-spectrogram)은 보코더(Vocoder)를 통해 wav 형식의 음성 파일로 생성된다. 여러 보코더 중 규칙 기반 알고리즘인 Griffin-Lim 과 뉴럴 보코더(Neural Vocoder)인 MelGAN, WaveNet 을 이론, 실험적으로 비교 분석하여 선정하였다. 각 모델은 속도와 정확도 간 trade-off 가 존재하였으며, 속도는 Griffin-Lim, MelGAN, WaveNet 순으로 우수하였고, 정확도는 WaveNet, MelGAN, Griffin-Lim 순으로 우수하였다[11]. 다만 MelGAN 은 GAN 모델 특성상 성능이 일정하지 않다는 단점이 존재했으며, 본 논문의 적은 데이터 셋으로 학습을 진행한 후 실험적으로 비교했을 때 Griffin-Lim 의 성과와 MelGAN 의 성능이 유사하였다. 빠른 속도와 다양한 문장에 일관된 성능을 요하는 본 논문의 성격을 고려하여 Griffin-Lim 을 보코더로 선정하였다.

3) 감정 분석 모델

감정 분석 데이터 셋으로는 AI Hub 로부터 10,351 개의 연기 영상 데이터 및 스크립트⁴와 상황에 따른 24,627 건의 발화 음성 및 감정 라벨링 데이터⁵를 활용하였다. 본 모델은 멀티 모달로써 오디오와 텍스트만을 필요로 하기 때문에 제공받은 m2ts 파일 형식을 wav 파일로 변환하여 오디오를 추출하였고 이를 1 차원 벡터로 변환 후 라벨링 결과를 대응한 pickle 파일 형식으로 입력 데이터를 생성하였다.

본 스피커의 감정 분석 모델로는 멀티모달 Transformer 를 채택하였다[12]. 기존의 멀티 모달 모델에서는 텍스트와 오디오의 서로 다른 시퀀스(sequence) 길이를 동일하게 맞춰주기 위해 얼라인먼트 방식을 적용해왔다. 그러나 장기 의존성 문제를 가지는 LSTM 에 적용했을 때 국소 의존성에는 유리하나 광역 의존성 반영에는 한계점이 존재했다. 따라서 얼라인먼트 대신 닷-프로덕트(dot-product) 어텐션을 적용함으로써 서로 다른 타임스탬프(timestamp)에 존재하는 정보들을 직접적으로 연결하는 방식이 감정 분석에 더

효과적인 것으로 판단하여 본 모델을 채택하게 되었다.

오디오와 텍스트는 각각 MFCC 와 KoBERT 를 거치며 특징 추출 과정을 거치고 닷-프로덕트 어텐션을 사용하기 위해 1 차원 컨볼루션(convolution)을 거치며 차원을 동일하게 맞춰준다. 이는 crossmodal Transformer 의 입력값으로 활용되며 텍스트와 오디오는 각각 소스와 타겟 역할을 하며 두 개의 페어가 모델에 각각 적용되고 각 출력값은 셀프 어텐션을 거쳐 최종적으로 감정을 분류하게 된다.

2.3 프로토타입

2.3.1. 인공지능 스피커 시제품

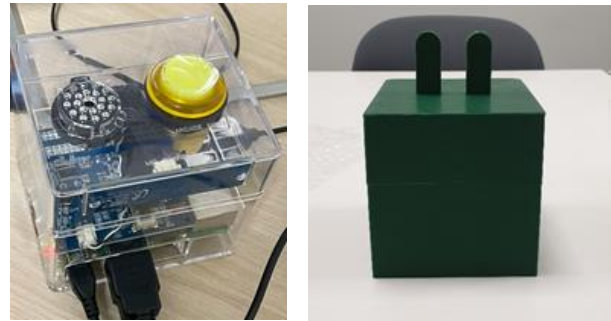


그림 2. (좌) 하드웨어 본체, (우) 3D 프린터로 출력한 스피커

2.3.2. 시제품의 프로세스

라즈베리 파이의 한정적인 메모리와 각 딥러닝 모델이 필요로 하는 패키지의 버전 충돌 때문에 각 모델을 다른 PC 에서 실행하면서도 개발자가 다른 조치를 취하지 않아도 라즈베리 파이와 각 PC 들이 파일을 공유하며 모든 프로세스가 연속적으로 이뤄져야 한다. 이를 해결하기 위해 삼바 서버를 사용해 라즈베리 파이의 공유 폴더에 승인된 PC 만 접근할 수 있도록 만들었고 Watchdog 패키지를 이용해 입력 파일이 새로 생성되는 이벤트가 발생하면 자동으로 모델의 입력값으로 들어가도록 개발했다. 사용자가 스피커를 호출어로 부르면 스피커는 사용자의 모든 발화를 일시적으로 저장하면서 상담을 시작한다(그림 1). 사용자의 음성을 라즈베리 파이에서 KT Voice2Text API ⁶ 를 이용해 텍스트로 변환한다. STT 결과(stt_result.txt)는 공유 폴더 내에 저장해 자연어 처리 모델의 Watchdog observer 가 이벤트를 인식하게 한다. 자연어 처리 모델은 STT 결과를 입력값으로 사용하여 상담 답변 텍스트를 출력값(answer.txt)으로 공유 폴더에 저장한다. 이를 음성 합성 모델의

⁴ <https://aihub.or.kr/opensdata/kefi-data/recognition-visual/KETI-01-001>

⁵ <https://aihub.or.kr/opensdata/kefi-data/recognition-language/KETI-02-002>

⁶ <https://apilink.kt.co.kr/api/menu/apiSpDetail.do?apiSpId=57>

Watchdog observer 가 인식하여 입력값으로 사용하고 텍스트에 맞는 유명한 음성 파일(answer.wav)을 출력값으로 공유 폴더에 저장한다. 라즈베리 파이의 Watchdog observer 가 음성 파일을 인식하면 자동으로 스피커로 출력한다. 사용자의 입장에서는 스피커가 유명인의 목소리로 답변을 한 것처럼 보인다.

2.3.3. 웹 프로토타입



그림 3. (좌) 웹 초기 화면, (우) 로그인 후 메인 화면

감정 분석 리포트

신유면 친구의 심리 상담 감정 분석 결과는 아래와 같습니다.

오늘의 기분은 ['슬픔']이네요!



고민되고 걱정스러운 일이 신유면님을 많이 슬프게 만들었군요. 실컷 울어버리고 싶은 마음이 들 때는 한번 울고 털어버리는 것도 하나의 방법이 될 수 있어요. 자신의 감정을 솔직하게 받아들이고 자신을 위로하는 시간을 가져 보세요.

분석 그래프

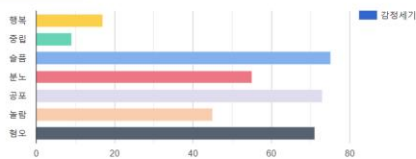


그림 4. 감정분석 결과 출력 화면

본 논문에서는 내담자의 발화 내용을 바탕으로 감정 분석 결과 리포트를 제공하기 위해 웹 애플리케이션을 추가적으로 제작하였다. 감정 분석에 사용되는 인공지능 모델이 파이썬 기반임을 감안하여 원활한 적용을 위해 파이썬 웹 프레임 워크인 플라스크(Flask)를 사용하였다.

상담을 수행한 본인만 결과지에 접근 가능하도록 로그인 기능을 구현하였으며 로그인을 해야만 감정 분석 리포트를 열람할 수 있도록 설정하였다. 사용자가 로그인 후 메인화면의 '상담 분석하러 가기' 버튼을 누름으로써 데이터 전처리 및 감정 분석이 시작된다(그림 3). 라즈베리 파이는 상담 중 이루어진 내담자의 모든 발화 음성을 문장 단위 bin 파일로 저장하며 각 문장별 STT 결과 또한 하나의 텍스트 파일로 지정된 경로에 저장한다. 이렇게 수집한 데이터는 삼바 서버를 통해 플라스크에서 접근 가능하게 되고 플라스크에서는 bin 파일 내

존재하는 미가공 데이터를 오디오로 변환 후 1 차원 벡터로 만든다. 각각의 음성 벡터들은 STT 결과와 병합되어 멀티모달 Transformer 에 전달된다. 모델은 7가지 감정에 대한 예측 결과를 행벡터 형태로 반환해주며 이를 결과 리포트 페이지에서 그래프로 출력하기 위해 0 과 1 사이의 값으로 정규화시킨 형태로 값을 넘겨주고 그중 최댓값을 사용자의 대표적인 감정으로 설정해 주었다. 이를 통해 사용자는 대표적인 감정에 따라 7 가지의 다른 일러스트 및 멘트를 제공받게 되고 하단의 그래프를 통해 감정별 분포도를 확인할 수 있다(그림 4).

2.4 모델 결과 및 평가

본 논문의 달성도를 각 모델의 정확도와 설문조사로 평가했다. 자연어 처리 모델, 음성 합성 모델의 경우 모델의 성능을 수치적으로 표현하기가 모호하기 때문에 설문조사를 통해 평가를 추가적으로 진행하였다.

2.4.1. 모델 결과

1) 자연어 처리

```

output = predict("반백정 나왔는데 친한 친구가 하나도 없어")
Input: 반백정 나왔는데 친한 친구가 하나도 없어
Output: 반백정이 잘 되지 않아서 새학기가 걱정이 되겠어요 . 새학기가 되면 새로운 반에 새로운 친구를 사귀어 나가면 된답니다 .

output = predict("친구들이랑 잘 지내고 싶은데 다가가는 방법을 모르겠어")
Input: 친구들이랑 잘 지내고 싶은데 다가가는 방법을 모르겠어
Output: 친구들과 잘 지내고 싶은 마음이 느껴지네요 . 먼저 친구들에게 다가가서 말을 걸어보면 어떨까요 .

output = predict("내가 말을 걸면 친구들이 불편해 할까봐 걱정이야")
Input: 내가 말을 걸면 친구들이 불편해 할까봐 걱정이야
Output: 말을 잘 하지 않으면 상대방도 눈치가 보일수 있어요 . 그러나 너무 긴장하지 말고 편하게 말을 건네보세요
    
```

그림 5. 자연어 처리 답변 결과

Transformer을 200번의 에폭(epoch)만큼 학습시킨 결과 35%의 정확도에 도달하였다. 정확도가 일반적인 학습 결과에 비해서 좋지 않게 나온 것을 볼 수 있는데, 자연어 처리 모델의 평가는 정확도로 하지 않고 퍼플렉서티(perplexity, PPL)로 하기 때문이다. 퍼플렉서티는 모델이 실제로 관측되는 값을 얼마나 잘 예측하는지를 뜻하며, 이 값이 작을수록 모델이 문서를 잘 반영한다. 하지만, 언어 모델은 일반적으로 학습 데이터에 전적으로 의존하기 때문에 만약 평가 데이터가 다른 도메인이거나 문장의 스타일이 다르다면 기존 모델의 퍼플렉서티 값이 낮더라도 결과가 좋지 않을 수 있다. 반대의 경우도 마찬가지로, 모델의 문서 반영도는 낮더라도 모델의 예측 값이 실제 대화와 같이 자연스러울 수 있다. 따라서 자연어 처리 모델의 성능 평가는 여러 문장을 테스트한 결과를 직접 보고 평가하였다. 그림 5는 세 문장을 모델에 각각 입력으로 넣었을 때 예측한 문장을 반환한 것이다. 아동·청소년들이 공통적으로 가질 만한 고민으로 새 학기 친구관계의 주제로

질문	답변					
	외개인아가 (카카오톡)	심심이	바비 (헬로우봇)	대경대학교 (헬로우봇)	가짜톡	하이 휴둥(Ours)
“반배정 나왔는데 친한 친구가 하나도 없어”	“친구야 많지. 그래도 너와 제일 친해”	“응”	“언젠가 마음이 꼭 맞는 진정한 친구를 만날 거예요”	“못 알아 듣는 말”	“ㅋㅋ”	“반배정이 잘 되지 않아서 새학기가 걱정이 되는 것 같아요. 새학기가 되면 새로운 반에 새로운 친구를 사귀어 나가면 된답니다.”
“친구들이랑 잘 지내고 싶은데 다가가는 방법을 모르겠어”	“다가가려고 노력은 해봤어?”	“네 사이 좋게 잘 지낼게요”	“**님은 할 수 있어요 제가 응원하고 있을게요”	“못 알아 듣는 말”	“아 그렇구나”	“친구들과 잘 지내고 싶은 마음이 느껴지네요. 먼저 친구들에게 다가가서 말을 걸어보면 어떨까요”
“내가 말을 걸면 친구들이 불편해 할까봐 걱정이야”	“”굳이 말을 안 걸면 잘 모르는 사람들이 덜 불편하게 느껴질 것 같아”	“알겠어요..”	“그런 일이 있었군요”	“못 알아 듣는 말”	“걱정하지마 다 잘될꺼야”	“말을 잘 하지 않으면 상대방도 눈치가 보일 수 있어요. 그러니 너무 긴장하지 말고 편하게 말을 건네보세요”

표 1. 타 챗봇과의 답변 비교 결과

세 문장을 받았을 때 위로와 격려를 담은 답변을 자연스럽게 주는 것을 확인할 수 있다.

적다는 것을 고려하면 뛰어난 성능이며, 데이터가 추가된다면 더 높은 성능을 보일 것으로 기대된다.

2) 음성합성

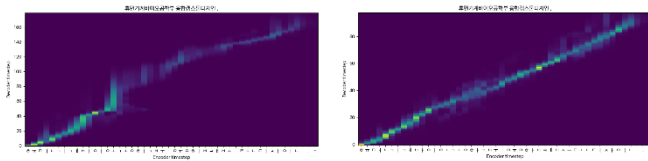


그림 6. Tacotron2의 어텐션 얼라인먼트 시각화 결과

Tacotron2는 멜-스펙트로그램 예측값과 실제 멜-스펙트로그램 사이의 MSE loss를 사용하여 학습했다. 그림 6에서 왼쪽은 1만 스텝(step), 오른쪽은 10만 스텝을 학습시킨 후 “휴먼기계바이오공학부 융합 캡스톤디자인”이라는 학습 데이터에 없는 단어로 테스트했을 때 어텐션 얼라인먼트를 시각화한 모습이다. 완전히 학습된 후에는 어텐션이 매 시점 디코더(decoder)에 정확하게 정보를 전달하는 것을 확인할 수 있다. 실제 유명한 음성과 굉장히 유사한 음성을 생성하지만, 합성된 음성엔 기계음 같은 노이즈가 섞여 있다는 단점이 있다. 하지만 본 논문에서 사용한 데이터 수가 아주

3) 감정분석

감정 분석의 경우, 멀티모달 Transformer의 학습 데이터에 대한 F1 점수는 7가지 감정 모두 1이라는 우수한 결과를 보여주었다. 실제로 심리 상담 결과에서 개별적인 발화에 대해 분류가 잘 이루어지는 것을 확인할 수 있었다. 다만 특정 상황에서 다양한 감정이 존재할 수 있고 이는 주관적인 판단으로 이루어지기 때문에 한 발화에 대해 7가지 감정 중 하나의 감정으로 완벽하게 분류하는 것에는 한계가 있어 보였다. 7개 이상의 세부적인 감정 라벨링이 가능하다면 더욱 더 섬세한 분석 결과와 함께 사용자의 구체적인 자기 이해가 가능할 것이다.

2.4.2. 관련 서비스와의 비교

멘탈케어 어플 '트루스트'에서 제공하는 AI 챗봇의 주된 동작 방식은 버튼식 대화흐름으로, 사용자가 자신의 감정을 털어놓고 위로를 얻는 깊이있는 대화를 하기 어렵다. 또한 전문가와의 상담을 유도하는 역할을 하기 때문에 대화의 끝은 종종 전문 상담사 추천으로

마무리된다. AI 채팅 어플 '심심이'의 경우, 간단한 일상 대화를 하는데에는 높은 성능을 보이지만, 아동 청소년이 가지고 있을 만한 고민을 입력 시 적절한 답변을 주지 못하였다. 가상친구 채팅 어플 '가짜톡'의 경우, AI 챗봇의 성격 및 사용자와의 관계를 설정하는 기능이 있어, 사용자가 원하는 방식의 대화상대를 만들 수 있다는 차별점이 있다. '듣고 싶은 말' 기능을 사용하면 대화 예시들을 입력하여 자신이 원하는 대화 스타일을 학습시킬 수 있다. 하지만, 마음이 힘들고 도움이 급박한 사용자의 경우 고민 상담을 위한 대화 방식을 직접 입력하는 과정을 감수해야 한다는 단점이 있다.

시중에 존재하는 가상 챗봇, 심리상담 챗봇과 본 논문에서 제안하는 스피커를 비교하고자 총 5개의 챗봇을 선정하여, 동일한 질문에 대한 답변을 확인하였다(표 1). 실험 결과, 시중에 서비스되고 있는 다른 챗봇의 답변은 적절하지 못하거나 단순한 위로인 반면, 본 논문에서 제안한 스피커는 적당한 위로와 구체적인 해결방안을 제시하였다. 문맥에 맞는 답변을 준 외개인이자 챗봇과 헬로우봇의 바비 챗봇의 경우에도 모든 질문에 적용될 수 있는 간단한 위로만을 하였고 구체적인 해결방안은 제시하지 못하였다는 점에서 심리상담으로 활용되기에는 역부족이다. 그에 비해 본 논문에서 제안한 스피커의 답변은 적절한 위로와 해결방안을 각각 한 문장씩 도출함으로써 아동 청소년을 위한 심리상담에 특화된 우수성을 입증하였다.

2.4.3. 설문조사

인공지능 스피커의 성능 측정을 위해 타겟 이용자인 아동·청소년 58명을 대상으로 설문 조사를 진행하였다(표 2). 설문 조사는 자연어 처리, 음성합성, 감정 분석, 전체 만족도 4 가지를 평가하기 위한 문항으로 구성하였고, 답변은 1점부터 5점 중에 선택하도록 하였다. 2-2 문항만 예외로 단답형식으로 답변을 받았다. 표 2에 기입된 점수는 백분율로 환산한 값이다.

	설문조사 문항	점수
1번	(자연어 처리 관련) 스피커의 대답이 내담자의 발화에 적합하게 어울리는가?	81.38%
2-1번	(음성 합성 관련) 시연 영상에서 어떤 유명인의 음성을 합성한 것일지?	37명/58명 (63.79%)
2-2번	(음성 합성 관련) 스피커의 목소리가 특정 유명인의 실제 목소리와 비슷한가?	75.52%
3번	(감정 분석 관련) 상담을 한 내담자의 감정이 다음 7가지 감정 중에 무엇에 가장 가까운 것 같은가?	70.69%

4번	(전체 만족도) 해당 인공지능 스피커가 아동, 청소년의 마음챙김 및 우울증 개선에 도움이 될 것 같은가?	81.72%
----	--	--------

표 2. 설문조사 문항 및 각 문항별 평균 점수

스피커의 주요성능인 4 가지 카테고리에 대해 모두 70 점 이상의 점수를 받았다. 스피커의 답변이 마음 챙김과 위로의 기능을 할 수 있는지 자연어 처리 모델의 적절성 판단 척도로 1번 문항을 포함했고, 81.38%의 응답자가 스피커의 답변이 아동·청소년의 마음챙김에 적절하다고 판단하였다. 음성합성 모델의 적절성 판단 척도로 2-1,2 문항을 포함했다. 2-1 문항은 음성 합성에 사용한 특정 유명인 이름을 언급하기 이전에 해당 유명인이 누구일지 단답형으로 답변을 받았고, 58명 중 63.79%에 해당하는 37명이 정답을 맞추었다. 이름을 언급한 후 진행한 2-2 문항은 75.52%의 응답자가 합성된 목소리가 유사하다고 판단하였다. 감정분석 모델의 적절성 판단 척도로 3번 문항에서는 스피커가 제시한 내담자 감정을 질문했고, 70.69%의 응답자가 감정분석 모델과 동일한 감정을 선택했다. 마지막으로 전체적인 서비스 만족도를 파악하기 위해 4번 질문을 포함했고, 81.72%의 응답자가 만족한다는 결과를 얻었다.

3. 결론 및 향후 연구

본 논문에서는 아동·청소년을 대상으로 상담의 심리적 진입장벽을 낮추고 사용자의 정신 건강을 증진시키고자 인공지능 스피커를 이용한 아동·청소년 심리 상담 통합 시스템을 제안하였다. 내담자의 음성을 입력하면 딥러닝 기반의 자연어 처리 모델이 답변을 생성하고, 음성 합성 모델을 거쳐 유명인의 목소리로 음성을 출력한다. 이후 상담 내용을 바탕으로 내담자의 감정 상태를 분석하여 웹을 통해 확인할 수 있다. 일련의 시스템은 하드웨어와 소프트웨어 부문의 여러 과제에 대해 통합적인 서비스를 제공한다는 면에서 장점을 가진다.

실제 구현된 아동·청소년 심리 상담 통합 시스템과 현재 서비스되고 있는 타 상담 챗봇에 대해 동일한 값을 입력하고 생성된 답변 비교를 통해 우수성을 검증하였다. 아동·청소년을 대상으로 진행한 평가에서도 전반적으로 높은 점수를 받으며 이용자의 공감을 이끌어내는 것을 확인하였다.

향후 연구에서는 시스템 개선 및 확장을 위해 추가적인 데이터 구축 및 모델 개발을 진행할 계획이며, 하나의 라즈베리파이 내에 모든 인공지능 모델을 서빙할 수 있도록 모델 압축 방안을 모색할 예정이다. 또한, 본 논문에서는 이용자인 아동만을 대상으로 성능 평가를 진행하였다. 차후 연구에서는 상담 센터와 협력하여 복합적인 감정에 대해서도 자세한 심리분석이 가능하도록 고도화하고자 한다.

[참고 문헌]

- [1] 질병관리청,교육부. 2020. 『청소년건강행태조사 통계: 2005-2456』
- [2] 통계청. 2019. 「인구동향조사」및「사망원인통계」
- [3] 교육부, 한국교육개발원. 2020. 「교육통계분석자료집-유초중등교육통계편」
- [4] 홍혜영, “청소년의 상담에 대한 인식 및 태도에 관한 탐색적 연구”, The Korea Journal of Counseling, 2006
- [5] Yunhui Chey et al, “ The Mediating Effects of Self-Efficacy and Social Support on the Relationship between Interactive SNS Usage of Smart Devices and Depression in the Elderly”, Korean Journal of Clinical Psychology, 2018
- [6] Ara Lee et al. “A Study on the Client Experience using Chatbot based on Counseling Theory”,JESK, 2019
- [7] Ashish Vaswani, “Attention is All You Need”,Computer Science Computation and Language, 2017
- [8] K. Park, “KSS Dataset: Korean Single speaker Speech Dataset,” <https://kaggle.com/bryanpark/korean-single-speaker-speech-dataset>, 2018.
- [9] J. Shen et al., "Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions," 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 4779-4783, doi: 10.1109/ICASSP.2018.8461368.
- [10] Yi Ren et al., “FastSpeech2: Fast and High-Quality End-to-End Text to Speech,” 2021 International Conference on Learning Representations (ICLR), 2021
- [11] Kundan Kumar et al, “MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis”, NeurIPS, 2019
- [12] Tsai et al.,“Multimodal Transformer for Unaligned Multimodal Language Sequences”, ACL 2019.

생체 정보의 비식별화를 위한 지문 탐지 모델

배은기[○] 김한동 권동균 한종대

상명대학교

withjust7@naver.com, aggsae@gmail.com, kdg5436@gmail.com, elvenwhite@smu.ac.kr

Fingerprint detection model for de-identification of biometric information

Eun-Gi Bae[○] Han-Dong Kim Dong-Kyun Kwon Jong-Dae Han
Sangmyung University

요 약

본 논문에서는 사진에 대하여 개인식별이 가능한 지문을 비식별화 하기 위한 첫 단계로, 다양한 조건에서 촬영된 사진에 대해 지문의 존재를 인식하는 모델의 구현을 다루었다. 이를 위해 본 연구에서는 객체 탐지 모델인 YOLOv5 과 Faster R-CNN 모델을 사용하여 탐지된 결과 이미지를 통해 정성적인 평가를 진행하고, mAP(Mean Average Precision)와 mIoU(Mean Intersection Over Union)를 측정하여 정량적인 평가를 진행하였다. 평가 결과는 다음과 같다. YOLOv5는 Pascal VOC의 mAP 평가 방식 기준으로 검증용 이미지에 대해 0.887과 평가용 이미지에 대해선 0.716을 달성하였고, Faster R-CNN은 52.6의 mIoU 값을 달성하였다. 본 연구의 성과를 활용하여, 사진 앱 및 SNS 등에서 도용 가능성이 있는 지문을 식별하고 사용자에게 경고할 수 있으며, 향후 추가 연구를 통하여 지문을 변조 혹은 비가역적 변환하여 개인정보의 도용을 막을 수 있을 것으로 기대된다.

1. 서 론

최근 화상기기의 발전에 따라, 디지털 사진의 해상도가 점차 증대되는 추세이다. 대부분의 최신형 핸드폰은 4K 해상도의 카메라를 탑재하고 있으며, 이에 따라 인물 사진에 포함된 개인식별 정보(지문, 홍채 등)의 정보값이 사이버 보안 위협을 일으키기에 충분할 정도로 커지고 있다. 언론 보도[6][7]에 따르면 이미 현재의 기술로도 어렵지 않게 고해상도 사진으로부터 지문을 복제하여 악용할 수 있음이 알려져 있고, 이에 따라 SNS등에 사진을 올릴 때 개인식별 가능한 생체 정보를 비식별화(De-Identification)할 필요가 커지고 있다.

본 연구는 사진에 포함된 개인식별 가능한 생체 정보 중 지문에 대한 비식별화를 위한 첫 단계로, 여러 사진에 대해 지문의 존재를 인식하는 모델의 구현을 목표로 한다.

2장에서는 어떤 방식으로 수행했으며 어떤 데이터를 사용하여 진행했는지에 대해 설명한다. 3장에서는 YOLOv5를 사용하여 학습한 것에 대해 설명한다. 4장에서는 Faster R-CNN을 사용하여 학습한 것 대해 설명한다. 5장에서는 결론 및 향후 진행할 연구에 대하여 논하였다. 6장에서는 본 연구를 진행하면서 참고한 문헌들에 대해 서술하였다.

2. 연구의 수행 방법 및 데이터 구성

본 연구의 수행에 있어 핵심적인 요소는 딥러닝 기술을 사용하여 지문에 해당하는 객체를 식별하는 문제이다. 객체 식별(Object Detection)은 딥러닝의 대두와 함께 많은 기술적 향상이 있어온 분야이며, 다양한 모델이 연구되어 성과를 거두어 왔다. 여러 모델 중 YOLOv5와 Faster R-CNN을 모델로 사용하여 구현하였다.

딥러닝 기반 객체 식별의 경우 지도 학습(Supervised Learning)을 통해 수행되어야 하며, 이를 위하여 잘 정제되고 적절하게 레이블링 된 데이터셋이 필요하다. 그러나 공개된 데이터셋 가운데 손동작 인식을 위하여 손끝(Fingertip)이 레이블링 되어 있는 데이터셋은 손가락 끝의 앞면과 뒷면을 구분하지 않아 지문 검출에 부적합하고, 지문(Fingerprint) 데이터셋의 경우 오직 지문만이 화상 내에 포함되어 있어 일반적인 사진에서 지문을 검출하는 데 부적합하다. 이에 따라 자체적으로 데이터셋을 준비했으며, 각 모델에 대한 데이터셋 형식을 맞춰주어 구축하였다.

자체적으로 구축한 데이터셋을 통해 모델을 훈련된 모델을 사용하여 탐지된 결과 이미지를 통해 정성적인 평가를 진행하고, Mean Average Precision(mAP), mean Intersection Over Union(mIoU)를 측정하여 정량적인 평가를 진행하였다.

3. YOLOv5를 이용한 모델 구현

3.1 YOLOv5의 개념 및 구현

Yolo-v5 모델은 1-step 방식의 객체 탐지 모델로 이미지를 작은 그리드로 나누어 객체가 있을 영역을 학습하고 해당 영역에 존재하는 객체가 어떤 객체인지에 대한 정보를 학습한다. 객체의 정확한 영역에 대한 학습을 진행하지 않기 때문에 객체의 영역에 대한 정보를 추가로 학습하는 2-Step 방식의 객체 탐지 모델에 비해 빠르지만 일반적으로 성능은 다소 떨어진다.

하지만 YOLOv5는 탐지 속도는 더욱 빨라졌으며 2-Step 방식에 비해 낮았던 성능을 향상시켰다. YOLOv5는 크게 Backbone과 Head 부분으로 구성된다. Backbone은 이미지로부터 Feature map을 추출하는 부분으로 'CSP-Darknet'을 사용한다. YOLOv5는 Backbone의 구조에 따라 세부 모델이 나뉘는데 가장 가벼운 n부터 가장 무거운 x까지 'n,s,m,l,x'가 있으며 총 5개로 이루어져 있다. Head는 추출된 Feature Map을 바탕으로 물체의 위치는 찾는 부분이다. Anchor Box를 처음에 설정하고 이를 이용하여 최종적인 Bounding Box를 생성한다. YOLOv3와 동일하게 3가지의 Scale에서 바운딩 박스를 생성하고 각 Scale에서 3개의 Anchor Box를 사용한다. 총 9개의 Anchor Box를 사용하는 것이다.

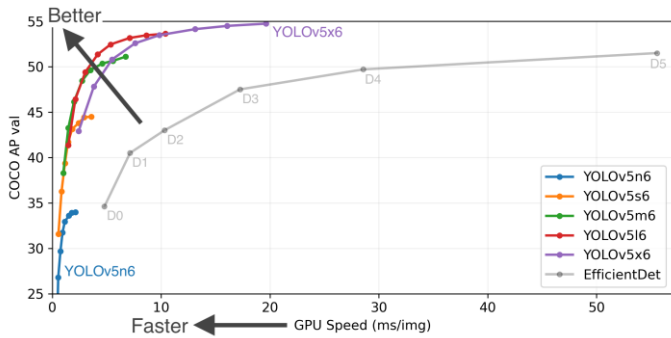


그림 1. YOLOv5의 세부 모델 별 성능 지표

YOLOv5의 최신 버전은 v6.0이며 (그림1)과 같이 5가지의 세부 모델로 나뉘어 있다. YOLOv5x6 모델은 성능이 비교적 뛰어나지만 처리속도가 느리고, YOLOv5n6 모델은 처리속도는 비교적 빠르지만 성능이 뒤떨어진다는 것을 확인할 수 있다. 본 연구에서는 지문의 영역을 탐지하는 성능이 중요하므로 성능이 뒤떨어지는 YOLOv5n6 모델을 제외한 4개의 모델에 대해 학습을 진행 후 평가하였다.

정성적, 정량적 평가를 위한 모델 선정은 다음과 같다. 먼저 YOLOv5의 세부 모델은 x6, l6, m6, s6가 있으며,

각 세부모델마다 Batch-Size와 전이학습에 사용시에 사용할 Weight 총 2가지를 변경하며 훈련하였다. (그림2)는 설정 별 세부 모델의 훈련 결과를 나타내는 그래프이다. 각 그래프의 명칭은 훈련 시 설정한 값에 따라 작성하였으며 '세부모델 - BatchSize - Weight'와 같이 기록하였다. (그림2)에서 'l6-32-l6' 모델이 가장 좋은 성능을 보여주는 것을 확인할 수 있으므로, 'l6-32-l6' 모델로 정성적, 정량적 평가를 진행하였다.

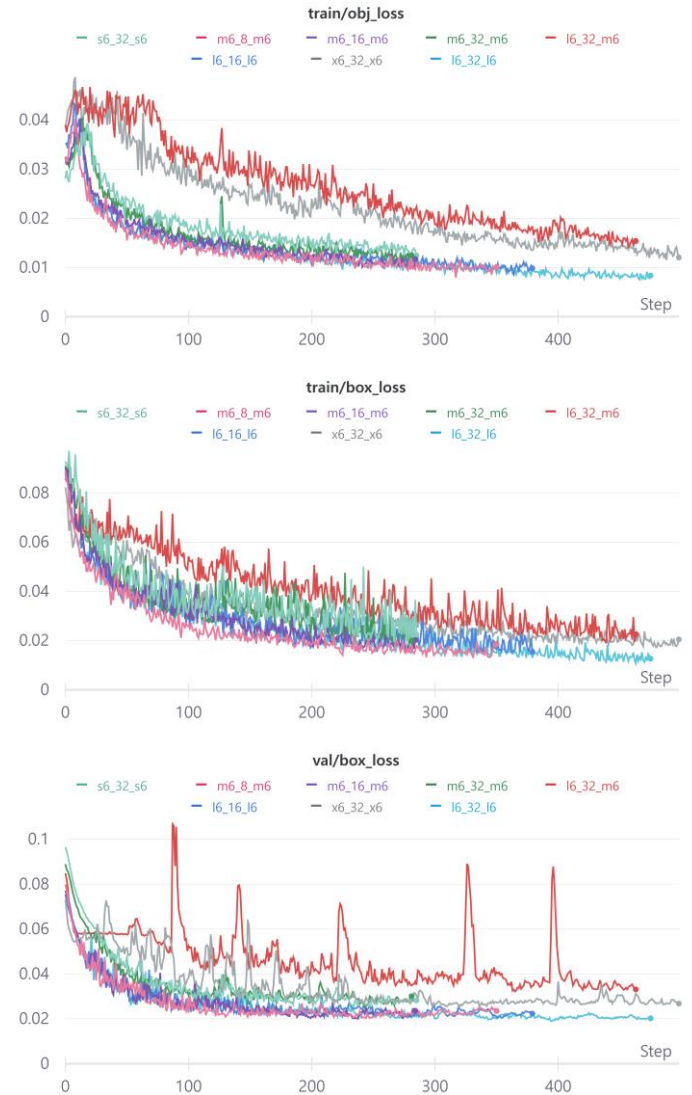


그림 2. YOLOv5의 세부 모델 별 훈련 결과

연구에서는 128개의 학습 데이터, 37개의 검증 데이터를 사용하여 YOLOv5 모델에 대한 학습을 진행하였다. 그리고 모델이 잘 학습되었는지에 대한 정성적, 정량적 평가를 진행하기 위해 22개의 이미지를 활용하였다. (그림 3, 4, 5, 6)는 정성적 평가 결과이다. 지문 식별을 실패한 케이스는 1개(그림 5)이다. 식별에 실패한 이미지를 제외한 21개의 이미지 중 지문이 존재하지 않는 이미지가 아니라면, 최소 1개 이상의 지문을 식별한 것을 확인할 수 있다. 단, (그림 5)와 같이 몇몇 이미지에서는 지문을 포함해 손톱까지

탐지하거나, 지문이 위치한 배경과 지문의 색이 비슷한 경우에는 탐지하지 못한 결과를 확인할 수 있었다.



그림 3. 탐지 성공 이미지 (지문 有)

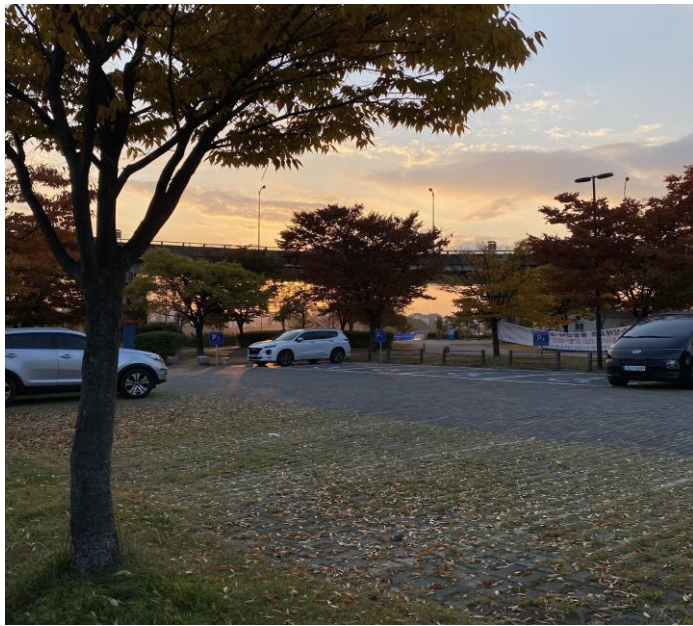


그림 4. 탐지 성공 이미지 (지문 無)

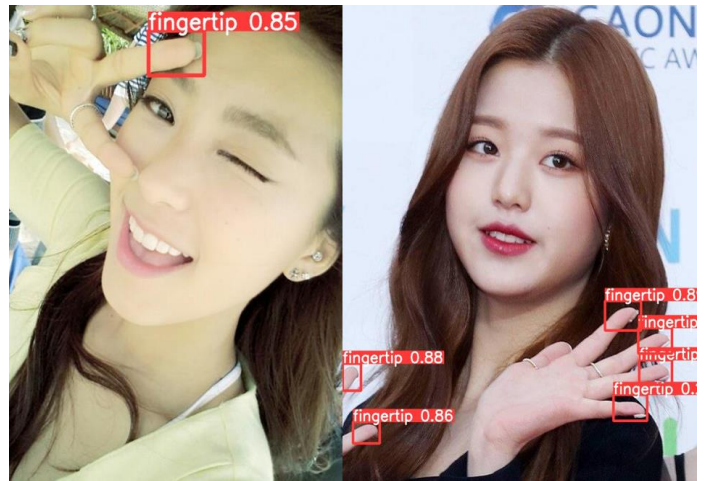
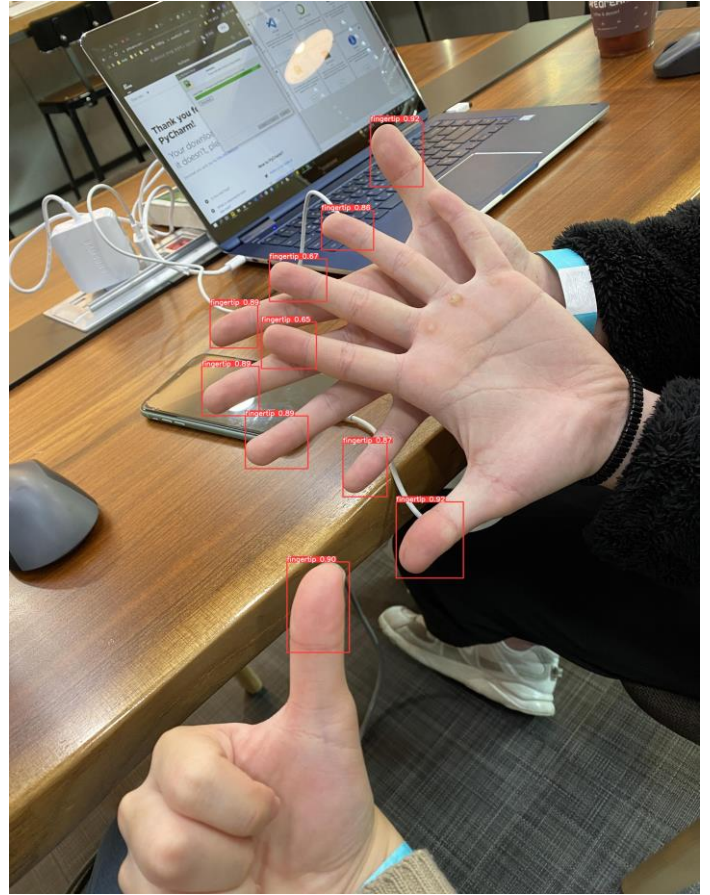


그림 6. 예외 이미지

3.2 훈련된 모델의 성능 평가

성능 평가에 대해 설명하기 앞서 본 연구에서 정량적 평가를 위해 사용된 평가지표인 mAP에 대해 간략히 설명한다. AP(Average Precision)는 컴퓨터 비전 분야에서 평가 지표로서 많이 사용된다. AP는 PR(Precision-Recall) 곡선의 그래프 선 아래쪽의 면적으로 계산된다. 검출하고자 하는 물체의 Class가 여러 개인 경우, 모든 Class의 AP를 구한 뒤 평균을 계산해서 mAP를 구하여 평가한다.



그림 5. 탐지 실패 이미지

YOLOv5에서는 정량적 평가를 위해 모델의 성능(정확도)을 mAP를 통해 확인하므로, 훈련이 가장 잘 된 Yolo-v5-l6-32-l6 모델에 대해 mAP를 측정하여 평가하였다.

표 1. mAP 측정 결과

	검증용 이미지	평가용 이미지
이미지 개수	37	22
Precision	0.919	0.728
Recal	0.831	0.843
mAP@.5	0.887	0.716
mAP@.5:.95	0.507	0.382

(표 1)에서 mAP@.5는 Pascal VOC의 mAP 평가 방식이고 mAP@.5:.95는 COCO의 mAP 평가 방식이다. Pascal VOC 기준으로 검증용 이미지에 대한 mAP@.5는 0.887, 평가용 이미지에 대한 mAP@.5는 0.716이 나온 것을 확인할 수 있다. 추가적으로 Yolo-v5에서 제공하는 최신 버전인 v6.0의 YOLOv5l6 (pretrained) 모델은 검증용 이미지에 대한 mAP@.5는 0.716이다. 또한 일반적으로 검증용 이미지에 대한 mAP 값보다 평가용 이미지에 대한 mAP 값이 낮은 경향을 보인다. 본 연구에서 진행한 평가용 이미지에 대한 mAP@.5와 Yolo에서 제공하는 Pretrained 모델의 검증용 이미지에 대한 mAP@.5 값이 동일하다는 것은, 지문 탐지에 있어서 YOLOv5는 좋은 성능을 보이고 있다는 점을 증명한다.

4. Faster R-CNN을 이용한 모델 구현

4.1 Faster R-CNN의 개념 및 구현

Faster R-CNN은 빠르고 좋은 성능으로 널리 알려져 있는 실시간 객체 탐지(Real Time Object Detection) 모델 중 하나이다. 앞서 사용한 Yolo-v5 모델과는 다른 2 Step 방식을 사용하고 있다. 1 Step 방식과 2 Step 방식의 가장 큰 차이점은 객체가 존재하는 영역을 탐지하는 ‘Region Proposal’의 수행 여부이다. 1 step 방식에서 객체가 존재하는 영역은 미리 객체를 작은 그리드로 나눈 후 그리드 내에 객체가 존재하는지 여부를 통해서 이미지 내에 존재하는 객체를 탐지한다. 반면, 2 Step 방식에서는 실제 객체가 존재하는 영역에 대한 정보를 기반으로 학습이 이루어진다. 학습 과정에서 이미지 내에 존재하는 객체에 대한 영역 정보와 해당 객체가 어떤 것인지 동시 학습한다. 그렇기 때문에 1 Step 방식에 비해 상대적으로 느리지만 더 좋은 성능을 얻을 수 있다.

Faster R-CNN의 핵심 아이디어는 실제 객체가 존재하는 영역인 ‘Region of Interest’(RoI) 탐지에 ‘Region Proposal Network’(RPN)를 사용했다는 것이다. 이전 모델인 Fast R-CNN에서는 RoI를 얻기 위해서 ‘Selective Search’(SS)를 사용하였다. Fast R-CNN에서는 SS를 통해 Region Proposal을 추출한 후, 전체 이미지에 대해 CNN 연산을 진행하고 결과를 통해 RoI 값 추론과 객체의 분류 작업으로 이어지는 방식으로 진행되었다. 이 방식의 문제는 SS가 CNN 외부에서 진행되기 때문에 속도의 한계가 있다는 문제가 있었다. Faster R-CNN에서는 이를 RPN으로 대체하여 RoI를 계산한다. 이를 통해 CNN 외부에서 진행되는 Region Proposal 작업을 CNN 내부에서 수행할 수 있게 함으로써 수행 속도를 크게 향상시킬 수 있었다. (그림 7)은 Faster R-CNN의 동작을 그림으로 표현한 것이다. 입력 이미지에 대해서 CNN을 통해서 이미지 내에 존재하는 특징(Feature)을 추출해낸다. 추출해낸 특징은 RPN에 입력으로 주어진다. RPN은 CNN에서 추출해낸 특징을 통해서 RoI를 얻어낸다.

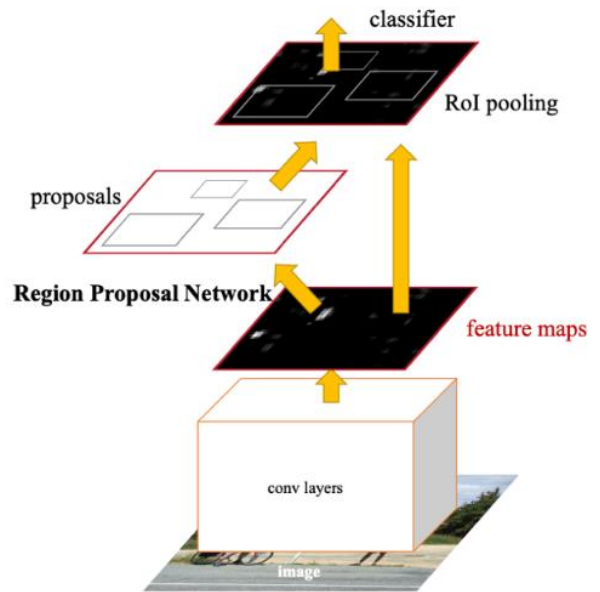


그림 7. Faster R-CNN의 동작 방법

이후 얻어낸 RoI와 해당 객체에 대한 정보는 합쳐서 학습에 사용되게 된다. 이전 RoI의 계산이 학습 과정과 별개로 이루어졌던 것과 달리 학습 과정에서 객체에 대한 정보와 RoI를 한꺼번에 처리하기 때문에 더 빠르고 정확하게 객체 탐지를 수행할 수 있게 된다. (그림 8)은 Faster-RCNN에서 기존 모델과의 성능 비교를 위해 논문에서 진행한 실험 결과이다.

Table 2: Detection results on PASCAL VOC 2007 test set. The detector is Fast R-CNN and VGG-16. Training data: “07”: VOC 2007 trainval, “07+12”: union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2k. †: this was reported in [5]; using the repository provided by this paper, this number is higher (68.0±0.3 in six runs).

method	# proposals	data	mAP (%)	time (ms)
SS	2k	07	66.9 [†]	1830
SS	2k	07+12	70.0	1830
RPN+VGG, unshared	300	07	68.5	342
RPN+VGG, shared	300	07	69.9	198
RPN+VGG, shared	300	07+12	73.2	198

그림 8. Faster R-CNN의 실험 결과

기존 사용되던 SS는 1,830ms 동안 70.0%의 mAP를 나타낸 것에 비해 RPN을 Region Proposal로 사용했을 때는 훨씬 더 좋은 73.2%의 mAP를 약 10배 빠른 198ms 동안 수행하는 것을 확인할 수 있다.

Faster R-CNN은 기존 학습 과정과 별도의 과정을 통해 RoI를 계산하던 모델들과 달리 RPN을 적용하여 학습 과정 내에 이미지에 대한 특징을 기반으로 RoI를 계산하여 더 빠르고 정확하게 객체의 정보와 RoI에 대한 정보를 얻어낼 수 있음을 실험을 통해 확인할 수 있었다. 연구에서는 이미지 내에 존재하는 지문을 탐지하기 위해 1 Step 방식 외의 2 Step 방식인 Faster R-CNN을 활용하여 주어진 이미지 내에 존재하는 지문을 탐지할 수 있는지 확인하였다.

연구에서는 총 106개의 Pascal VOC 방식으로 구성된 학습 데이터를 사용하여 Faster R-CNN 모델에 대한 학습을 진행하고 27개의 평가 데이터를 사용하여 모델에 대한 평가를 진행하였다. 모델이 잘 학습되었는지 확인하기 위해 정성적, 정량적 평가를 진행하였다. 이미지가 탐지해낸 객체의 정보를 기반으로 bounding box를 그려 실제 지문이 존재하는 영역을 잘 탐지하였는지 정성적으로 평가를 진행하였다. 또한, 실험 결과에 대한 mean Intersection Over Union(mIoU)를 계산하여 정량적인 평가를 진행하였다. IoU는 객체가 탐지한 영역과 실제 지문 진실(ground truth)의 영역이 겹치는 정도를 수치적으로 표현한 값으로 주로 객체 탐지 또는 Semantic Segmentation의 성능 평가를 위한 척도로 주로 사용된다. IoU는 다음과 같이 탐지한 영역과 지문 진실의 영역을 합한 영역 중 겹치는 영역이 차지하고 있는 비율로 표현된다.

$$IoU = \frac{\text{Overlapping area}}{\text{Union area}}$$

그림 9. IoU 계산 방법

모델을 통해서 식별한 이미지 내 존재하는 객체의 박스와 지문 진실의 박스에서 겹치는 영역을 계산한다. 이렇게 하나의 이미지에 대한 모델의 성능을 평가할 수 있다. 주로 전체 모델의 성능을 파악하기 위해 평가에

사용된 모든 이미지에 대한 IoU를 계산한 후 이에 대한 평균을 내는 mIoU를 평가 척도로써 사용한다.

(그림 10, 11, 12)는 정성적 평가 결과이다. 이미지 내에 존재하는 지문 식별을 실패한 2개(그림 12)의 이미지를 제외한 25개의 이미지에서 최소 1개 이상의 지문을 식별한 것을 확인할 수 있다. 단, (그림 11)과 같이 몇몇 이미지에서는 지문이 아닌 손끝을 탐지하거나 지문과 전혀 관계없는 객체를 탐지하거나 같은 지문을 여러 번 식별하는 결과를 확인할 수 있었다.



그림 10. 지문 탐지 성공 이미지

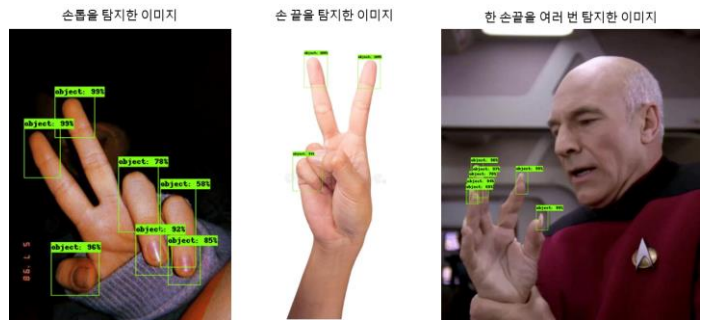


그림 11. 예외 이미지

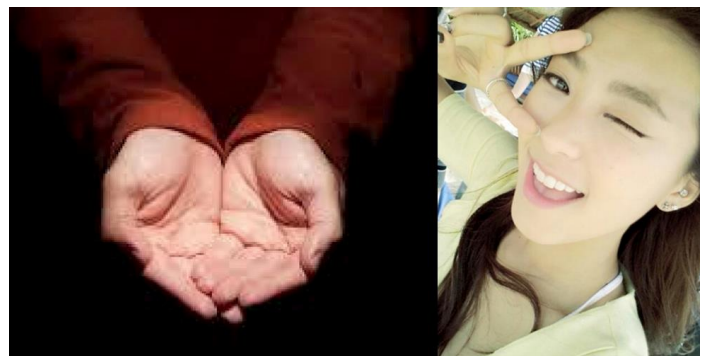


그림 12. 탐지 실패 이미지

4.2 훈련된 모델의 성능 평가

(표 2)는 평가 이미지에 대해 수행한 정량적 평가 결과이다. 정량적 평가를 위해 각 이미지에 존재하는 지문에 대한 IoU를 계산하여 이미지 별 mIoU를 계산하였고 이를 종합하여 전체 모델에 대한 mIoU를 평가하였다.

표 2. mIoU 측정 결과

이미지 id	지문 수	모델이 탐지한 지문 수	mIoU 값
1	3	4	39.9
2	5	6	45.9
3	10	7	70.2
4	3	2	60
5	2	2	90.4
6	2	3	85.2
7	2	2	49.1
8	5	6	37.3
9	19	11	28.9
10	7	0	-
11	5	5	79.8
12	5	5	61
13	2	2	78.3
14	2	2	80.6
15	5	5	34.4
16	2	3	72.7
17	5	5	42.4
18	5	3	41
19	2	0	-
20	2	3	60
21	2	7	35.4
22	5	3	40.3
23	3	4	77.6
24	5	5	42.2
25	4	4	71.8
26	2	2	71.6
27	5	7	24.2
전체 mIoU			52.6

계산된 mIoU 값은 소수 둘째 자리에서 반올림을 진행하였다. 탐지를 실패한 2개의 이미지를 제외한 25개의 이미지에서 가장 낮은 mIoU는 24.2이고 가장

큰 mIoU는 90.4이다. 각 이미지의 mIoU를 종합하여 모든 이미지에 대한 mIoU를 계산하면 52.6임을 얻을 수 있다.

5. 결론 및 향후 계획

자체적으로 구축한 이미지에서 YOLOv5와 Faster R-CNN 두 가지 모델을 학습시켰으며, 지문을 탐지할 수 있도록 구현하였다. 두 모델 모두 지문 탐지에 충분히 사용할 수 있다는 것을 보여주는 결과가 도출되었다.

앞으로 더 많은 데이터에 대해 학습을 진행하고, 수집된 지문 데이터에 대해 융선(fr), 색상(fc), 지문 유형(fp)를 인지하여 보다 정교한 비식별화를 수행할 계획이다. 이에는 GAN을 이용한 가상의 지문 생성, 비식별화 평가 모델 학습 등의 추가적인 연구가 필요할 것으로 예상된다.

본 연구의 성과를 활용하여, 사진 앱 및 SNS 등에서 도용 가능성이 있는 지문을 식별하고 사용자에게 경고할 수 있으며, 향후 추가 연구를 통하여 지문을 변조 혹은 비가역적 변환하여 개인정보의 도용을 막을 수 있을 것으로 기대된다.

6. 참고 문헌

[1] REN, Shaoqing, et al. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 2015, 28: 91-99.

[2] KUZNETSOVA, Anna; MALEVA, Tatiana; SOLOVIEV, Vladimir. Detecting apples in orchards using YOLOv3 and YOLOv5 in general and close-up images. In: International Symposium on Neural Networks. Springer, Cham, 2020. p. 233-243.

[3] PATTERSON, Genevieve; HAYS, James. Coco attributes: Attributes for people, animals, and objects. In: European Conference on Computer Vision. Springer, Cham, 2016. p. 85-100.

[4] GIRSHICK, Ross. Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. 2015. p. 1440-1448.

[5] RAHEJA, Jagdish Lal; DAS, Karen; CHAUDHARY, Ankit. An efficient real time method of fingertip detection. arXiv preprint arXiv:1108.0502, 2011.

[6] Alex Hern, Hacker fakes German minister's fingerprints using photos of her hands, The Guardian, 2014

[7] Alice Yan, China's scissor-hand selfie-takers warned of cybersecurity threat, South China Morning Post, 2019

스마트 팩토리 - 가스라이터 제조 공정에서의 스티커 부착과 관련된 불량 검출 프로그램 개선

신관수, 송호준, 조영진⁰, 신연순
동국대학교 컴퓨터공학전공

happysun7080@dgu.ac.kr, sson514@naver.com, guswind8@gmail.com, ysshin@dongguk.edu

Smart factory – Improvement of defect detection program related to sticker attachment in gas lighter

Gwan-Su Shin, Ho-Jun Song, Young-Jin Cho⁰, Youn-Soon Shin
Department of Computer Science and Engineering, Dongguk University

요 약

최근 외국산 라이터가 가격 경쟁력을 바탕으로 국내 시장을 공략하고 있다. 국내 제조사인 A사는 국내 라이터 시장에서 가장 큰 점유율을 유지하고 있으나, 외국산 라이터 점유율도 계속 오르고 있다. 이러한 상황에서 국산 라이터 제품은 외국산 제품과 가격 폭은 좁히고, 불량률은 줄여서 품질 격차를 벌려가며 시장 수성에 나서고 있다. 불량률을 줄이기 위해서는 불량품을 적절하게 검출해야 한다. 이에 따라, 본 논문에서는 OpenCV를 활용하여 라이터의 불량 사례 중 하나인 스티커의 부착 상태 불량과 훼손 상태에 대한 불량 검출 자동화 시스템을 제안한다. 해당 시스템을 개발하고, 실험을 통해 정확성을 검증하였으며 결과적으로 실험 대상인 세 종류의 불량 사례에 대해 매우 높은 확률로 불량을 판별할 수 있었다.

1. 서론

[1]에서는 2018년 12월 정부는 중소 제조업체 절반을 스마트 공장으로 만들어 '제조 강국'을 실현하겠다는 비전을 발표했고, 구체적으로는 2022년까지 스마트 공장 3만 개 보급, 그리고 스마트 산업단지 10곳을 만들겠다는 목표를 제시했다. 이에 따라 2019년 중소벤처기업부의 산하기관인 '스마트 제조혁신 추진단'[2]이 출범하여 스마트 공장 구축에 필요한 다양한 지원을 해왔고, 2020년에는 중소벤처기업부 내에 '중소기업 스마트 제조혁신 기획단'이 출범하여 각종 정책을 추진하여 스마트 공장 보급에 힘쓰고 있다. 스마트 공장 구축에 힘쓰는 이유는 제조 공정의 스마트화는 제품 생산성과 품질을 높이고 불량률은 줄여서 제조업체에 큰 경쟁력을 제공할 수 있기 때문이다. 본 연구진은 공정의 스마트화를 추진하는 국내 라이터 제조업체인 A사와 협력하여 일회용 가스 라이터에 부착되는 스티커의 부착 상태와 훼손 상태에 대한 불량을 검출하는 시스템을 개발했다.

A사에서 생산하는 라이터는 국내 라이터 시장 점유율의 약 50%를 차지하고 있으며, 해당 공장은 국내 유일의 라이터 생산 공장이다.[3] 그러나 최근 외국산 라이터가 저렴한 가격 경쟁력을 바탕으로 국내 시장을 공략하고 있다. A사는 제조 공정의 원천기술을 통해 외국산 제품과 가격 폭은 좁히고, 품질 격차를 벌려 시장 점유율을 유지하기 위해 노력하고 있으나 외국산

라이터의 점유율이 조금씩 올라가고 있는 어려움을 겪고 있다. 이에 따라 공장의 스마트화를 추진하여 불량률은 줄이고 생산성과 품질을 높이고자 한다. 종래의 공정에서 스티커 불량 검출은 전부 육안으로 이루어졌다. 스티커 불량은 상황 특성상 육안으로 판별하는 것이 정확도는 높을 수 있으나, 하루 생산량이 약 20만 개에 달하는 라이터를 현장 근무자가 일일이 검출하는 데는 분명한 한계가 존재한다. 불량 검출 정확도는 근무자 피로도의 영향을 받으며, 이러한 불량 검출 방식은 전체 공정의 속도를 낮춰 생산성을 저하시킨다. 따라서 본 연구에서는 OpenCV 실시간 이미지 프로세싱을 기반으로 한 라이터 제조 공정에서의 스티커 불량 자동 검출 시스템을 제안하였다. 스티커 불량 상황은 크게 두 가지로 구분하였다. 스티커 부착 상태에 대한 불량, 그리고 스티커 훼손 상태에 대한 불량이다. 실험 결과 제안 시스템은 테스트 세트에 대해 100%의 정확도로 모든 불량 상황을 불량으로 검출해 냈고, 정상 상황은 99.7%의 정확도로 정상으로 판별할 수 있었다.

2장에서는 본 연구에 참고한 선행 기술 및 연구를 소개하고, 3장에서는 스티커 불량 검출 알고리즘을 설명하고, 4장에서 실험 과정과 결과를 보여주고, 5장에서 결론을 맺는다.

2. 관련 연구

2.1 스마트 팩토리 적용 시스템

본 연구를 진행하기에 앞서, 종래에 연구되었던 스마트 팩토리에 적용하기 위한 시스템을 참고했다.

[4]에서는 중소기업 생산 공장에서 사용할 수 있는 OpenCV 기반 재고관리 시스템을 제안하였다. 우선 라즈베리파이에 연결된 카메라를 통해서 재고 이미지를 실시간으로 전송받았다. 받은 이미지에서 재고의 라벨 및 바코드 영역을 인식하기 위해 OpenCV를 사용한 이미지 처리로 원하는 영역을 추출하였다. 라벨 영역 인식 과정은 다음과 같았다.

먼저 가우시안(gaussian) 필터를 활용하여 이미지의 노이즈를 제거하여 잘못된 엣지 검출 결과를 사전에 제거하였다. 다음으로 가우시안 필터가 적용되어 블러(blur) 처리된 이미지의 각도를 찾았다. 이어서 얻어낸 각도의 크기와 방향을 활용하여, 원치 않는 픽셀을 제거하기 위해 이미지의 전체 스캔을 수행하였다. 마지막으로 이진화(thresholding)를 적용하여 모든 엣지가 실제 엣지인지 아닌지를 결정하였다. 이렇게 하여 검출된 영역은 다양한 각도로 검출될 수 있으므로 이미지의 4개의 엣지 좌표를 추출하고 함수를 적용함으로써 반듯한 사각형으로 변환된 이미지를 얻었다. 그 이후 반듯한 이미지의 조명 명암을 제거해주면 전처리는 완료되고 최종적으로 라벨 외곽을 추출할 수 있었다. 본 연구에서는 실시간 불량 검출을 위해 [4]를 기초로 OpenCV를 활용하여 스티커 영역과 바코드 영역을 추출하였다.

2.2 OpenCV

본 연구에서는 OpenCV를 사용하여 라이터를 촬영하는 영상에서 원하는 영역만 추출하였다.

[5]에서는 의료 영상(X-ray 이미지)을 전처리, 가공하여 특징점을 검출해 내는 방법을 제안하였다. 의료 영상을 전처리하는 과정은 이미지를 일반화(normalization)하는 과정과 외곽선(contour)을 이용해 배경을 제거하는 과정으로 이루어졌다. 일반화 과정은 금속정과 뼈의 명암 대비를 높여 금속정 경계의 외곽선 찾기 위함이다. 일반화 이후, findContour() 함수를 이용해 금속정의 외곽선을 검출하고, 원본 이미지에서 금속정을 제외한 배경부를 마스킹하여 제거하였다.

특징점 검출 과정은 선 탐지와 각도 탐지를 수행하였다. 선 탐지는 이미지를 흑백(gray-scale) 이미지로 변환한 뒤, 비슷한 밝기 값을 가지는 부분들이 같은 값을 갖도록 이진화 해주었다. 처리된 이미지에 findContour() 함수를 이용하여 외곽선을 추출하고, drawContour() 함수를 이용해 추출한 외곽선을 그려주었다. 각도 탐지는 HoughLinesP() 함수를 이용해 허프 변환(hough transformation)을 수행하여 찾은 직선의 각도를 계산하였다.

본 연구에서 스티커 영역과 바코드 영역의 외곽선을

탐지하고 그리는 데 [5]를 참고하였다.

또한 외곽선을 연결한 직선의 각도를 계산하는 부분에서 착안하여 스티커의 각도를 사용하여 불량을 판별하는 알고리즘을 제시하였다.

3. 제안하는 스티커 불량 검출 알고리즘

3.1 불량 라이터 세트 판단

제안하는 불량품 검출 시스템은 촬영한 라이터 세트의 영상을 바탕으로 ROI(Region of Interest)를 설정한다[6]. ROI내 라이터에 부착되어 있는 스티커를 검출하여 해당 스티커의 각도와 부착 여부, 바코드 훼손 여부를 통해 불량 여부를 판단한다.

스티커 부착 상태는 다음과 같은 4가지 경우로 정의할 수 있다. 각 상태는 모든 라이터에 스티커가 정상적으로 부착된 경우, 라이터에 스티커가 부착되지 않은 경우, 스티커가 일정 각도를 벗어나 기울어져 다른 라이터와 겹쳐 부착된 경우, 스티커의 바코드 영역이 인식 불가능할 정도로 훼손된 경우를 말한다. 이러한 네 가지 상태를 바탕으로 라이터 세트의 불량품 여부를 판별하기 위해서는 판단의 기준이 명확해야 하고 판단의 정확성을 검증할 수 있어야 한다.



그림 1-(a) ㉠형 스티커



그림 1-(b) ㉡형 스티커

현재 A사에서 출하중인 일회용 가스 라이터는 그림 1-(a) ‘㉠형 스티커’와 그림 1-(b) ‘㉡형 스티커’로 구분할 수 있다. 3.3절에서 언급될 두 바코드는 EAN13(European Article Number) 타입의 1차원 선형 바코드이며, ㉠형 스티커의 바코드는 바코드 시작과 끝 부위에 약간의 백색 여백이 존재하기에 최하단에 텍스트가 적힌 ㉡형 스티커에 비해 높은 인식률을 확보할 수 있다.

라이터는 노랑, 빨강, 초록, 보라, 파랑으로 총 5가지 색상으로 구별된다. 하지만 4절에서 언급할 환경 제약 요인에 따라 라이터의 색상은 스티커 및 바코드 검출에 영향을 주지 않기에 구분하지 않고 개발을 진행하였다.

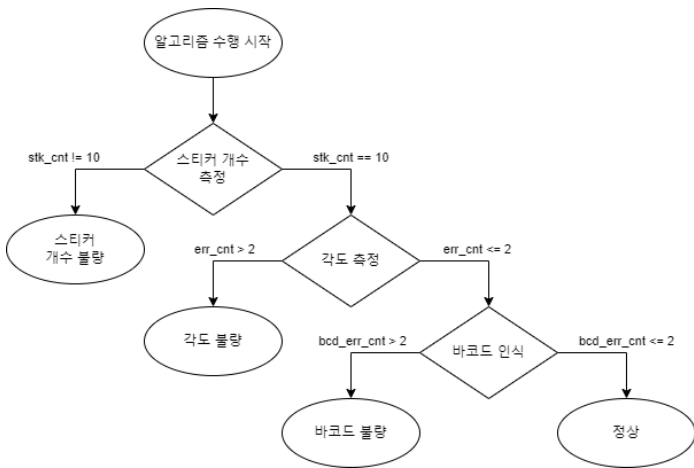


그림 2. 알고리즘 흐름도

라이터 세트는 총 3단계에 거쳐 불량 여부가 판단된다. 첫 번째 단계에서 10개의 라이터의 각 스티커를 검출하여 스티커 개수를 파악한다. 두 번째 단계에서는 각 스티커의 각도를 측정한다. 마지막 단계에서는 바코드를 탐색하여 최종적으로 정상 여부를 판별한다.

3.1 스티커 검출 후 개수 측정

우선 사용자는 촬영된 영상에서 라이터의 스티커가 정상적으로 붙어 있을 수 있는 몸통부분을 ROI로 설정하고 초점과 밝기, 대조 등의 환경적 잡음 없이 스티커가 가장 선명한 상태가 되도록 설정한다. ROI를 통해 해당 영역 내의 영상 이미지를 추출하고, 이 때 해당 이미지를 각도 계산 과정에도 사용하기 때문에 라이터의 헤드 부분을 제외시킨 하단 70%의 이미지를 추출해낸다.

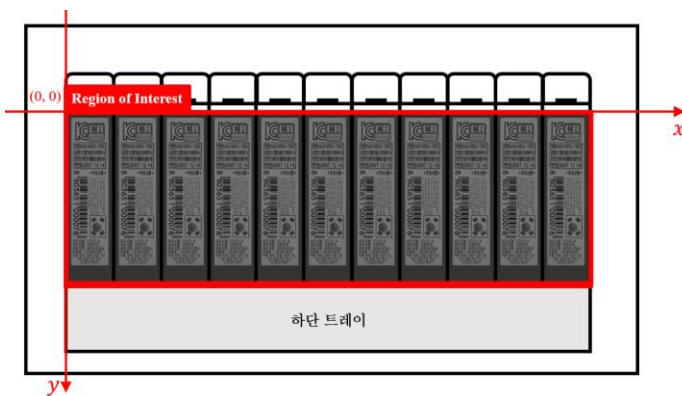


그림 3. ROI 및 픽셀 좌표계

본 알고리즘에서 사용될 픽셀 좌표계는 ROI의 좌측 상단 꼭짓점을 원점으로 갖는다. 또한 ROI내에서 하나의 스티커가 가질 수 있는 최소 영역 값을 roi_min_area , 최대 영역 값을 roi_max_area 로 설정한다. 추출한 이미지는 $cvtColor()$ 를 사용하여 BGR 이미지를 흑백 이미지로 변환하고 변환한 흑백 이미지는

$threshold()$ 를 사용하여 이진화한다. 이진화된 이미지에서는 해당 이미지에 있는 외곽선들을 $findContours()$ 를 사용하여 모두 추출해낸다.



그림 4. ROI 하단 70% 영역에서 추출된 외곽선

추출한 외곽선들은 반복문을 통해 하나씩 해당 영역이 스티커인지를 판별한다. 외곽선의 영역 값이 앞서 설정한 roi_min_area 와 roi_max_area 내에 포함되었을 경우에만 해당 외곽선을 스티커의 영역으로 판별하고 스티커의 개수를 카운트하는 stk_cnt 를 증가시킨다. 해당 이미지의 모든 외곽선을 검사한 후 스티커의 개수가 한 세트의 라이터 개수인 10개와 동일하다면 스티커는 정상적으로 부착되어 있다고 판별하고, 스티커의 개수가 적거나 많다면, 해당 라이터 세트는 ‘스티커 불량품 세트’로 판별한다.

3.2 검출된 스티커 각도 측정

3.1의 과정을 통해서 스티커로 검출된 외곽선들을 $minAreaRect()$ 을 사용하여 해당 직사각형의 좌상단의 꼭짓점 좌표, 너비와 높이, 각도를 저장한다. 이 때 각도는 픽셀 좌표계로 표현된 직사각형의 기울기로 정의한다.

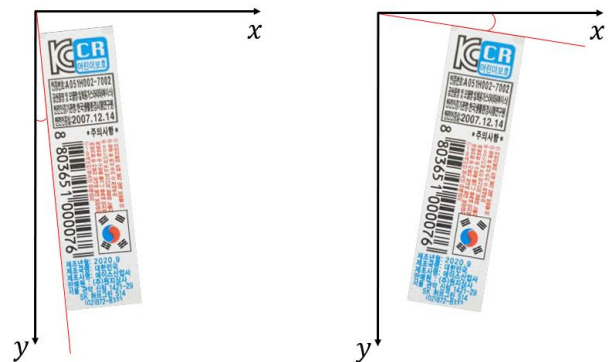


그림 5. 픽셀 좌표계로 정의한 스티커 각도

그림 5와 같이 스티커의 각 변이 구성하는 벡터와 픽셀 좌표계의 x축 또는 y축의 사이각을 계산한다. 이 때 스티커의 위상단 꼭짓점이 좌상단 꼭짓점보다 위쪽에 위치할 경우 세로변을, 그렇지 않을 경우 가로변을 벡터로 설정한다. 계산된 각도가 6도보다 클 경우에는 인접한 라이터까지 스티커가 중복되어 부착된 상태로 판별하여 각도 불량으로 판단될 수 있다. 이 때 제품 공정 상 각도가 6도 이상인 스티커가 라이터 세트에서 2개 이하로 발생하는 경우는 없다고 가정하기 때문에 6도가 넘어가는 스티커가 2개를 초과하는 경우에 해당 라이터 세트를 ‘각도 불량품 세트’로 판별한다.

3.3 바코드 인식

3.2에서 정상 스티커로 인식된 라이터들은 바코드 인식 알고리즘을 거쳐 최종 불량 여부가 판단된다. 우선 바코드 인식 정확도 향상을 위해 각도 계산에 사용했던 라이터의 헤드 부분이 제외된 ROI의 하단 70% 이미지를 기존 ROI 이미지로 복원하는 작업이 진행된다.

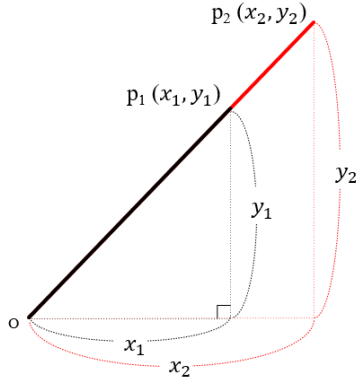


그림 6. ROI 복원

3.1에서 제외시킨 ROI의 상단 30% 영역을 복원하는 과정은 다음과 같다. $O = (0, 0)$, 선분 OP_1 의 길이가 ROI 높이의 70%, 선분 OP_2 의 길이가 ROI 높이의 100%라고 할 때, p_2 는 다음과 같다.

$$p_2 = \left(\frac{10}{7} x_1, \frac{10}{7} y_1 \right)$$

따라서 $O = (x_0, y_0)$ 이라면 p_2 는 다음과 같다.

$$p_2 = \left(x_0 + \frac{10}{7} x_1, y_0 + \frac{10}{7} y_1 \right)$$

위 과정으로 도출된 p_2 를 그림 3의 원점에 대입하여 바코드 인식 알고리즘을 수행한다.

본 절의 바코드 인식 알고리즘 수행 전 바코드의 크기와 부착 위치가 상이한 ㉠형과 ㉡형을 구분한다. 제조사에 따라 바코드만 존재하는 영역을 추출하고, 이를 'bcd_img'로 설정한다.



그림 7-(a) ㉠형

그림 7-(b) ㉡형

바코드 인식을 위해 bcd_img에 다음과 같은 영상 처리과정을 수행한다.

- (1) 영상을 시계 방향으로 90도 회전

- (2) 흑백 영상으로 변환
- (3) 세로(horizontal) 요소들 제외
- (4) 영상 블러 처리 및 이진화
- (5) 모폴로지(morphology) 연산 (closing)
- (6) 외곽선 추출 후 외곽선 사각형의 가로(vertical) 길이를 'bcd_width'로 설정

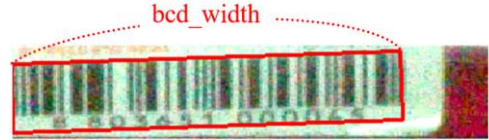


그림 8. 바코드의 외곽선과 bcd_width

'bcd_width'가 정상 바코드가 가질 수 있는 범위에 속하는 경우 해당 스티커를 정상적으로 부착된 스티커로, 그렇지 않은 경우 바코드 불량품 스티커로 판별한다. 또한 불량품 스티커의 개수가 2개 초과일 경우 '바코드 불량품 세트'로 판별한다.

4. 실험

4.1.1 실험 환경



그림 9. 전체 실험 환경



그림 10. 라이터-카메라 실험 환경

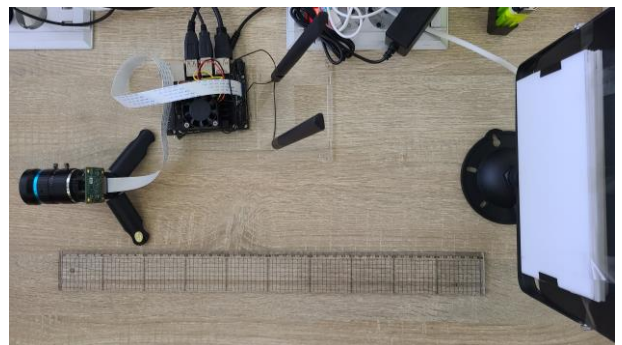


그림 11. 카메라-편광판 실험 환경

그림 9는 전체 실험 환경을 보여주는 사진이다. 그림 10은 실제 공정에 적용 가능한 거리에 라이터와 검은색의 배경, 카메라를 설치한 실험 환경을 보여주는 사진이다. 이 때, 지면으로부터 50mm 떨어진 위치에서 카메라가 라이터 세트를 촬영한다. 카메라 렌즈부터 라이터 세트 사이의 거리는 실제 공정에 적용 가능한 300mm이며, 라이터 세트 20mm 뒤에 검은색 배경을 설치한다. 그림 11은 실제 공정과 유사한 환경을 조성하기 위하여 카메라 렌즈부터 500mm 떨어진 위치에 편광판을 설치한 환경을 보여준다. 실제 공정에서는 라이터 세트가 컨베이어 벨트에 놓여져 있기 때문에 카메라는 지면과 수직으로 라이터 세트 위에 설치되어 있다고 가정한다. 천장에 있는 빛이 라이터 세트, 카메라와 일직선 상으로 위치해야 라이터 세트가 받는 빛의 양과 반사가 가장 일정하다. 따라서 실험 환경에서는 위와 같이 일정한 빛의 양을 라이터 세트와 카메라에 일직선 상으로 제공하기 위해 주변 빛을 모두 제거하고 카메라와 멀리 떨어진 후방에서 편광판의 빛을 약하게 비추어 진행한다. 아래의 표 1은 본 연구에서 실험에 사용한 하드웨어와 소프트웨어 정보를 보여준다.

표 1. 하드웨어 종류 및 소프트웨어 버전

하드웨어
Nvidia Jetson Nano B01
Raspberry Pi HQ Camera 12.3MP
16mm Telephoto Lens for Raspberry Pi HQ Camera
소프트웨어
Nvidia JetPack 4.4.1
OpenCV 4.5.0
Python 3.8.9

4.1.2 프로그램 설정

솔루션 작동을 위해 프로그램을 실행할 때 먼저 ROI를 설정해줘야 한다. ROI는 라이터 세트의 연료탱크 부분만큼만 직사각형으로 설정한다. 다만 라이터 세트 하단의 트레이는 제외하여 트레이 윗부분 경계까지 설정한다. 또한, 실험 환경의 조명 및 밝기에 따라 솔루션 작동 전 이미지 밝기와 대조를 설정해야 한다. 실험이 어디서 진행되는지에 따라 다르겠지만 대체로 밝기와 대조가 낮을 때 정상적인 좋은 결과가 나타난다. 추가적으로 3.1에서 언급한 바와 같이 ㉠형과 ㉡형 구별을 위해 수동으로 스티커 종류를 설정한다.

4.2 스티커 불량 검출 실험

4.2.1 실험 목적

실제 공정에서 발생 가능한 연료탱크 부분에 부착되는 스티커의 부착 상태와 훼손 상태에 대한 불량을 검출하는 정확도를 측정한다. 측정한 결과로 본

논문에서 제안하는 시스템의 실용성을 확인한다.

4.2.2 실험 방법

실험 환경 설정과 프로그램 설정을 마친 뒤 준비된 테스트 세트들에 대해 실험을 진행한다. 총 일곱 가지의 테스트 세트 중 두가지는 정상 세트이고, 다섯 가지는 불량 세트이다. 각 불량 세트를 스티커, 각도, 바코드 3개의 불량 케이스로 분류하여 총 네 개의 케이스에 대해 각각 1,000회씩, 총 4,000회 실험을 진행한다. 실험 결과 출력은 ‘정상 세트’, ‘Sticker 불량품 세트’, ‘Angle 불량품 세트’, ‘Barcode 불량품 세트’ 총 네 가지로 출력되며 ‘정상 세트’로 출력된 경우만 정상 라이터 세트로 판단한다.

4.2.3 실험 결과



그림 12-(a). 그림 12-(b). 정상 라이터 세트



그림 13. 스티커 불량 라이터 세트



그림 14. 각도 불량 라이터 세트



그림 15-(a). 그림 15-(b).



그림 15-(c).
바코드 불량 라이터 세트



그림 16. 스티커 및 각도 불량 라이터 세트



그림 17. 각도 및 바코드 불량 라이터 세트



그림 18-(a).

그림 18-(b).

그림 18. 스티커 및 바코드 불량 라이터 세트



그림 19. 스티커, 각도 및 바코드 불량 라이터 세트

표 2. 실험 결과

라이터 세트	실제 케이스	검출 결과	
		정상	불량
그림 12	정상	1994	6
그림 13	스티커 불량	0	2000
그림 14	각도 불량	0	2000
그림 15	바코드 불량	0	2000
그림 16	스티커 및 각도 불량	0	2000
그림 17	각도 및 바코드 불량	0	2000
그림 18	스티커 및 바코드 불량	0	2000
그림 19	스티커, 각도 및 바코드 불량	0	2000

표 2는 앞서 보여준 라이터 세트에 대하여 이를 정상, 스티커 불량, 각도 불량, 바코드 불량의 케이스들을 조합하여 각 케이스 별로 2,000회씩 실험을 한 결과이다. 이 때 각 케이스마다 불량 라이터의 개수와 위치를 변경해가며 실험을 진행하였다. 정상 라이터 세트의 경우 영상 속 노이즈에 의해 바코드 인식이 잘못되어 불량품으로 판별되는 경우가 발생한 것을 확인한다. 전체적으로 정상 케이스의 경우 99% 이상의 정확도를 보인다. 모든 불량 케이스의 경우 모든 실험 결과가 불량으로 검출되어 100%의 정확도를 보였고, 각 케이스 별 상이한 조건의 표본들 역시 동일한 결과를 보였다. 또한 16,000회의 실험 결과 평균 소요 시간은 0.178초로, 1초 이내에 판별을 완료해야 하는 실제 공정에도 도입하더라도 시간적인 문제는 발생하지 않는다.

이 때, 각도 불량 스티커가 포함된 케이스의 경우에는 이를 각도 불량 혹은 스티커 불량 두 가지의 경우로 검출하는데, 이는 두 개 이상의 스티커가 붙어있어 해당 스티커를 하나의 스티커로 판별하여 스티커의 개수가 10개보다 적어 스티커 불량으로 검출되기 때문이다. 하지만, 실제 공정에서 라이터 세트의 불량 판별은 케이스와 무관하게 해당 라이터 세트의 정상 또는 불량 여부를 정확하게 판단하는 것이 중요하다. 따라서 불량으로 판단된 세트는 케이스와 관계없이 실제 불량 라이터 세트라면 정확하게 검출한 것으로 결정한다.

5. 결론

본 논문에서는 OpenCV 실시간 영상 처리 오픈소스를 이용하여 가스 라이터의 스티커 부착 공정 시 발생할 수 있는 오류 상황 제시 및 불량품 세트를 판별하는 방법론을 제안한다. 특히 정상 세트와 불량품 세트를 높은 정확도로 분류한다는 점에 의미를 둔다. 향후 연구에서는 이동하는 컨베이어 벨트를 사용하는 실제 공정 환경을 고려한 실증 테스트를 거쳐 최적화 작업을 수행할 예정이다.

Acknowledgement

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학지원사업의 연구결과로 수행되었음 (2016-0-00017)

참고문헌

- [1] 최중윤, 중소기업스마트제조혁신기획단, “스마트제조 2.0으로 제조업 르네상스 시대 연다”, 인더스트리 뉴스, 2021,
(<http://www.industrynews.co.kr/news/articleView.html?idxno=43200>)
- [2] 중소기업기술정보진흥원 부설 스마트제조혁신추진단, <https://www.smart-factory.kr/>
- [3] 김영환, 라이터 시장, '독주' 불티나 두고 BIC 맹추격, 이데일리, 2013,
(<https://www.edaily.co.kr/news/read?newsId=01735126602877144>)
- [4] Jang, Su-Hwan, Jeong, Jopil, “Design and Implementation of OpenCV-based Inventory Management System to build Small and Medium Enterprise Smart Factory”, The Journal of the Institute of Internet, Broadcasting and Communication, pp.161-170, Feb, 2019
- [5] Kim, Hyeon-jin, Lee, Dan-Bi, Jung, So-Hee, Jo Hui-Won, Choe Su-Ah, Koh, Seok-Ju, Jo, Hyo-Jun, “Pre-processing, processing and keypoints detection of medical images (X-ray images)”, The Institute of Electronics and Information Engineers, pp.2195-2197, June, 2021
- [6] 주가기, “OpenCV를 이용한 골프공 인식 및 거리 예측 시스템”, 전남대학교 대학원, pp.4-24, 2014

수업 질문 자동응답을 위해 KoBert를 이용한 챗봇과 Doc2Vec을 이용한 챗봇과의 성능비교

허주은^{1,0}, 이선아^{1,2}

경상국립대학교 항공우주 및 소프트웨어 공학부¹, AI융합공학과²
uandeun@naver.com, saleese@gnu.ac.kr

A Chatbot Using a KoBert Model for Automatically Answering of Class Questions

Jueun Heo^{1,0}, Seonah Lee^{1,2}

Department of Aerospace and Software Engineering¹, Department of AI Convergence Engineering²
Gyeongsang National University

요 약

2019년부터 지속되고 있는 COVID-19의 영향으로 언택트 교육의 새로운 장이 열렸다. 언택트 교육의 질과 소통을 위한 플랫폼들이 개발되고 있으며 그중 하나의 도구로 챗봇이 손꼽히고 있다. 이전 연구에서 강의자가 수업 중 받은 질문과 답변 데이터셋을 구축해 Doc2Vec모형을 이용한 Q&A챗봇을 구축하였는데, 질문을 예측하고 답변을 제공하는 정확도가 낮은 아쉬움이 있었다. 본 연구에서는 학생에게 좀더 정확한 답변을 제공하는 성능이 높은 Q&A챗봇을 제공하기 위해 KoBert모형을 사용해 Q&A챗봇을 학습 및 평가하여 두 모델의 성능을 비교한다. 이를 통해 강사와 학생에게 더 좋은 답변 성능을 가진 Q&A챗봇 제공을 기대한다.

1. 서 론

2019년부터 시작된 COVID-19의 확산세는 집단면역을 위한 백신 접종에도 불구하고 멈추지 않고 있다. 최근 우리나라에서는 가장 많은 확진자들이 발생했고 이로 인해 사회적 거리두기가 다시 시행되고 있는 실정이다. COVID-19 확산 초창기에 교육계는 대면 강의가 온라인 플랫폼을 사용하는 비대면 방식으로 급격하게 전환되었고, 이에 따라 초기 온라인 교육에는 많은 보완점들이 존재했다[1]. 하지만 최근 2년간 교육계에서는 비대면 교육을 위한 여러 온라인 강의 플랫폼과 수업 방식 등을 정비해왔으며, 언택트 교육의 시대가 도래하였다[2]. 언택트 교육의 가장 큰 쟁점은 강사와 학생간의 질의 소통에 있다[3]. 현재 강사와 학생간의 질의는 일대일로 카카오톡, 문자, 메일등을 이용해 많이 이루어지고 있고 다른 학생들과 공유할 수 있는 게시판의 사용률이 현저히 낮다. 이에 따라 강사가 받는 질문이 중복되는 것이 많고, 똑같은 질문 일일이 답변해야 하는 번거로움이 다수 발생한다. 또한 게시판이나 메일등을 이용한 질의는 학생이 즉각적인 답변을 받기 힘들다는 단점이 있었다. 따라서 이전연구에서 수업을 위한 Q&A방법을 고민하여 강사와 학생 간의

반복적인 수고를 덜고 소통을 원활 하게 할 도구로 Q&A챗봇을 제안 및 구현하였다[4]. 하지만 Q&A챗봇의 학습에 사용된 Doc2vec모형이 비지도 학습 모델로 Q&A챗봇이 제공하는 답변에 대해 정확도를 측정하는데 어려움이 있었고 하이퍼파라미터 튜닝을 통해 가장 최적의 파라미터값을 찾아야 한다는 번거로움이 존재했다.

기존연구의 문제점을 보완하기위해 본 논문에서는 Q&A챗봇의 Doc2vec[5]모형을 KoBert[6]모델로 재구성 및 학습하고 Doc2vec모형과 KoBert모델의 질문에 대한 답변 정확도를 비교한다. 이를 통해 더 성능이 좋고 정확한 답변을 제공하는 Q&A챗봇을 학생들에게 제공할 수 있으며 수업의 질과 만족도도 또한 향상될 것으로 기대한다. 이를 위하여 강사가 객체지향프로그래밍 수업에서 이제까지 받은 질의를 정리하여 질문과 답변 형태의 데이터 셋으로 만들고, 이를 기반으로 KoBert 모델을 학습한다. 또한 이전 연구와 동일하게 Q&A챗봇은 학생들의 활용도 및 사용성이 가장 큰 카카오톡 메신저에 연동한다. Q&A챗봇 개발환경은 Python, Django, Nginx를 사용하였고, 카카오톡과 Django 연동은 카카오톡 오픈빌더의 외부 API를 연결하여 동작 파라미터에 반응하는 응답형식을 설계할 수

있게 제공하는 스킬 기능을 활용하였다.

본 연구의 구성은 다음과 같다. 2절에서는 배경정보에 관해서 설명한다. 3절에서는 본 연구에서 제안하는 Q&A 챗봇의 기법에 관해서 설명한다. 4절에서는 챗봇 데모에 관해서 설명한다. 5절에서는 이전연구인 Doc2Vec모델과 성능비교에 관해서 설명한다. 6절에서는 제한점에 관하여 설명한다. 7절에서는 결론 및 향후연구에 관하여 설명한다.

2. 배경 정보

2.1 챗봇

챗봇이란 챗팅과 로봇의 합성어로 시기술로 사람과 기계가 대화를 하는 인공지능 프로그램이다. 현재 많은 기업과 정부에서 기업 내부 직원 또는 고객들의 질문에 빠르게 답변하는 고객대응 및 상담 업무를 하고 있으며 기업의 업무 생산성과 효율성을 높이고 있다[7]. 특히 교육분야에서는 수강생들의 질문 및 피드백을 인공지능 기술이 대신함으로써 강사의 중복적인 업무를 줄여주는데 도움을 준다[8]. 현재 챗봇 서비스가 일부 기업사례에서 비즈니스 가치를 창출하고 있으나 서비스의 범위가 여전히 특정 기업의 프로세스에 국한되고 있으며, 서비스 범위와 편리성 및 유용성이 크지 않아 확산속도가 늦어지고 있어, 일부 챗봇 개발 서비스 업체는 사용성을 향상시키기 위한 통합 개발 플랫폼을 통해 비전문가가 챗봇을 구축할 수 있는 챗봇 서비스도 출시되고 있다[3]. 또한 교육용 챗봇은 교육적 활용에서 학습성취도, 학습 동기, 학습 참여율 향상에 긍정적인 영향을 주고있다[8].

2.2 Doc2Vec

Doc2Vec[5]은 2014년 구글 연구팀이 발표한 문서 임베딩 모델이고 Word2Vec[9]이 확장된 임베딩 방법이다. Word2Vec[9]은 문장단위에서 단어들의 유사도를 이용한다면, Doc2Vec[5]은 문장, 문단, 문서 등 더 큰 단위에서 단어 유사도를 구한다. 이전 연구[4]의 Doc2Vec[5]을 사용한 챗봇은 학습방법으로 PV-DBOW를 선택해 구현했다. PV-DBOW는 word vector를 제외하고 paragraph vector만을 이용해 학습을 하는 모델이고 예측 또한 word vector를 고려하지 않고 paragraph vector만을 사용해 단어를 예측을 한다.

2.3 KoBert

BERT[10]는 Bidirectional Encoder Representations from Transformers의 약자로 Transformers 라는 기계번역 모델의 Encoder만 사용하는 기계번역 모델이다. 2018년 10월 구글에서 발표한 BERT는 NLP 분야에서 매우 훌륭한 성능을 보여주었고, 굉장히 촉망받는 기계번역 모델이다. BERT의 장점은 사전 훈련된 pre-trained 모델이고, Fine-tuning 을 통해 레이블이 있거

나 없는 데이터들로 양방향학습을 통해 가중치를 업데이트하여 pre-trained 된 모델에 마지막 Classification layer만 수정함으로 사용자가 원하는 다양한 Task에 맞게 적용 및 구현이 가능하다. BERT[10]모델의 장점은 pre-trained되어 있기에 훈련시간이 짧아져 빠른 개발이 가능하고, Task의 데이터가 적어도 좋은 성능을 낼 수 있다.

KoBert란 Korean BERT의 약자이다[6]. KoBert는 SKT의 SKT-Brain에서 구글 BERT base multilingual cased의 한국어 성능 한계를 극복하기 위해 개발된 모델이다. 한국어 위키피디아에서 문장 5백만개와 단어 54백만개를 학습시켰고, 사전의 크기는 8,002인 한글 위키피디아를 기반으로 학습한 토큰나이저이다. 한국어의 불규칙한 언어 변화의 특성을 반영하기 위해 데이터 기반 토큰화 기법을 적용한다.

Fine-tuning 과정은 pre-trained된 모델에 원하는 Task에 대해 Classification layer만 추가하고, Task에 맞는 데이터로 학습 및 파라미터를 조절하는 작업이다. 본 연구에서는 단일 문장 분류 Task로 구현 했다. Fine-tuning은 하이퍼파라미터 변경을 통해 모델의 전체 파라미터가 변한다. 여러 번 하이퍼파라미터의 값을 수정하면서 최적의 성능을 찾아내는 과정이 정밀하고 세심한 작업이고 많은 테스트가 필요하다.

3. 제안기법

본 연구에서 제안하는 바는 기존의 Q&A 챗봇의 성능을 높이기 위해 Doc2Vec과 KoBert 모델에 기반한 두 Q&A챗봇을 구현한 후 성능비교를 통해 수업질의를 위한 교육 챗봇의 적절한 모델을 실험 및 판단하고자 한다.

3.1 전체적인 구조



그림 1. 서버구조

그림 2은 본 논문에서 제안하는 챗봇 서버구조이다. 웹 프레임워크는 Django를 웹서버는 Nginx와 uWSGI를 사용해 파이썬 애플리케이션을 배포했다. 챗봇 메커니즘을 구현하면 다음과 같다. 학생이 카카오톡에서 질문을 보내면 카카오 i 오픈빌더 스킴에 등록된 웹서버 URL로 질문이 전달되고 웹서버인 Nginx에서 Django 어플리케이션으로 질문이 전달된다. Django

어플리케이션은 받은 질문을 pre-trained된 KoBert 모델의 단일 문장에 대해 분류Task를 진행하도록 구현된 Classification layer에서 예측한 분류 값에 해당하는 답변을 웹서버 Nginx로 전달하고, Nginx는 카카오톡으로 전달한다. 본 연구에서는 Django를 이용해 카카오톡과 연동하여 Q&A챗봇을 구축한다. 이후 Doc2Vec과 KoBert 로 구현된 Q&A챗봇의 답변 정확도를 테스트한다.

3.2 KoBert Fine-tuning

본 연구에서는 BERT의 여러 Task중 단일 문장 분류 Task로 구현하였다. 단일 문장 분류는 다양한 Task에 사용되는데 감정분석, 주제 레이블링, 언어감지, 의도 분류에 사용된다. 본 연구에서는 학생의 질문의 의도를 판단해서 가장 비슷한 질문에 대한 답변을 제공하기 위해 단일 문장 분류 Task를 사용하였고, 구현 순서는 아래와 같다.

3.2.1 Pre-trained KoBert 모델 다운

SKTBrain 깃 허브에서 KoBert모델을 다운받는다.

3.2.2. 데이터셋 전처리

본 연구에 사용된 chatbot_qna_data.csv 파일은 번호, 질문, 답변 열로 구성되어 있어 질문, 답변 열만 사용한다. 이후 BERT의 단일 문장 분류 Task로 구현 하기 때문에 답변 질문과 답변을 1:1로 숫자 레이블 값을 붙인다. 질문과 답변이 1:1로 레이블링 되어 90개의 클래스 값이 생성된다.

3.2.3. KoBert입력 데이터로 변환

질문데이터를 KoBert모델의 입력 형태로 만든다. 질문의 각 토큰의 어휘 인덱스를 추출해 이를 정해진 길이의 벡터로 생성한다. 이후 문장의 개수를 구분하기 위한 토큰 타입 벡터 생성하고, 유효 길이 벡터를 생성한다.

KoBert는 입력된 문장을 3번의 임베딩 과정을 거치는데 첫번째는 Token Embedding과정을 통해 token을 임베딩 한다. 두번째는 Segment Embedding을 통해 문장의 개수를 나눈다. 세번째는 Position Embedding을 통해 sentence내의 순서인 position 정보를 알아낸다. 이후 이 세 임베딩 합을 사용한다.

KoBert입력 데이터로 변환한 후 각 문장은 3개의 값으로 출력되는데 첫번째는 문장을 토큰화와 패딩 한 시퀀스가 출력된다. 패딩은 문장의 길이가 다른 것을 같은 길이로 맞추므로 병렬연산이 가능하다. 본 연구에서는 max_len을 64로 지정했기 때문에 64 사이즈로 패딩이 일어나고 문장이 64 사이즈보다 짧을 시에는 자리를 채우기 위해 1을 채운다. 두 번째는 길이와 타입, 세 번째는 어텐션 마스크 시퀀스가 출력된다. 어텐션 마스크는 토큰 및 패딩 된 시퀀스의 1로 채워진

부분이 연산이 되지 않도록 알려주는 의미가 있다. 이렇게 KoBert모델의 입력데이터가 생성되고 이 데이터를 Pytorch에서 사용가능한 Dataset으로 변경하여 사용한다.

3.2.4. 하이퍼파라미터 값 세팅

본 연구에서는 pre-trained된 KoBert모델을 사용하였다. pre-trained된 모델에 하이퍼파라미터를 수정하여 Fine-tuning을 진행하였고, 하이퍼파라미터 값은 표 1과 같으며, 최적화 알고리즘으로는 Adam 최적화 알고리즘을 사용했다.

3.2.5. KoBert 학습모델 생성 후 학습

KoBert 학습 모델을 생성한다. 이때 설정 값은 model = bert, hidden_size=768, num_classes=91이고, 이후 KoBert 모델 학습시킨다.

3.2.6 질문에 대한 답변 예측

새로운 질문에 대한 답변 예측은 학생이 입력한 질문을 KoBert 데이터셋 형태로 만들고 Pytorch가 이해하는 형태로 변경해, Fine-tuning 된 KoBert모델에 넣으면 90개의 값으로 이루어진 텐서가 출력된다. 이때 argmax함수를 사용해서 텐서에서 가장 큰 값을 가지는 값의 위치 값과 같은 레이블에 해당하는 답변을 제공한다.

max_len	64	max_grad_norm	1
batch_size	1	log_interval	10
warmup_ratio	0.1	learning_rate	5e-5
num_epochs	20		

표 1. KoBert Fine-tuning 하이퍼파라미터 값

3.3 KoBert챗봇의 질의 처리

카카오톡을 통해 학생이 질문을 입력하면 질문이 웹서버를 통해 KoBert Q&A챗봇으로 전달된다. 이후 입력된 질문을 KoBert 데이터셋으로 변형하고 Classification layer를 통해 가장 비슷한 레이블 값을 가지는 질문의 답변을 사용자 대화창으로 전송한다.

4. 데모

무작위로 3개의 질문을 선정해서 두 챗봇의 답변을 비교해보았다. 그림 3는 Doc2Vec으로 학습한 Q&A챗봇을 테스트한 화면이다. 총 3개의 질문 중 1개만 정확한 답변을 제공했고 나머지 2개의 답변은 질문의 의도에 맞지 않는 엉뚱한 답변을 제공했다. 그림 4는 KoBert모델을 Fine-tuning한 Q&A챗봇을 테스트한 화면이다. 총 3개의 질문 중 3개의 질문 전부 정확한 답변을 제공했다.

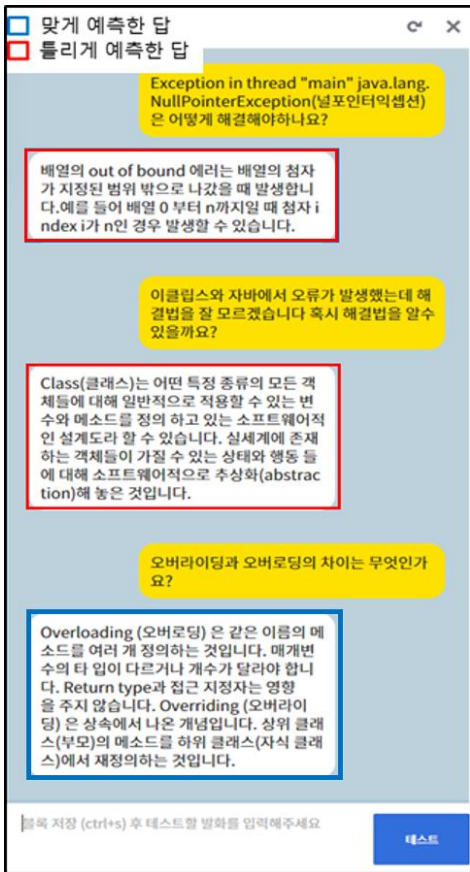


그림 2. Doc2Vec 챗봇

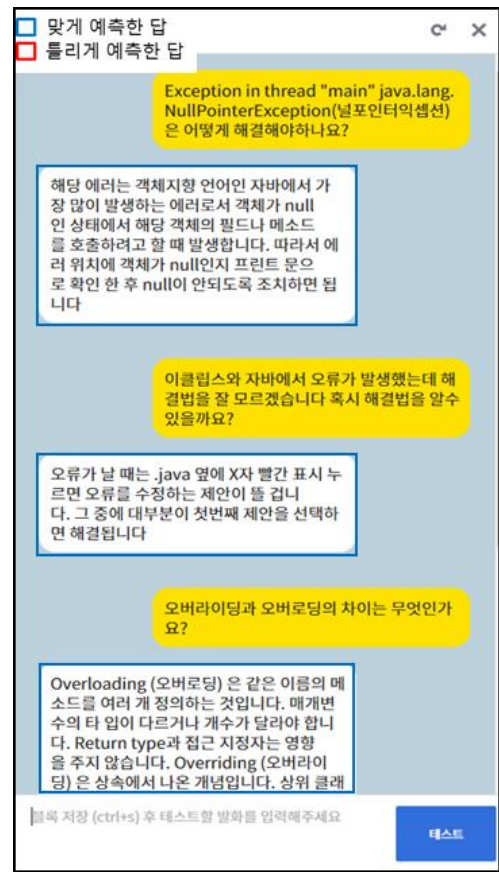


그림 3. KoBert 챗봇

5. Dco2Vec모델과 성능비교

Doc2Vec 두 Q&A챗봇의 성능 비교를 수행하여 교육 챗봇의 적절한 모델을 실험 및 판단하고자 한다.

5.1 실험방법

Doc2Vec과 KoBert 로 구현된 두 챗봇 모두 같은 데이터 셋을 가지고 테스트한다. 전체 데이터로 학습을 하고, 이후 동일한 전체 데이터로 테스트를 한다. 학습과 테스트를 각각 5번씩 진행하여 정확도를 측정한다.

5.2 실험데이터

실험데이터는 객체지향 프로그래밍 수업에서 학생들과 강사간의 오픈채팅방, 개인 문자, 게시판을 이용해 질문 및 답변한 Q&A데이터를 모아 번호, 질문, 답변 순으로 정리했고, 또한 객체지향 프로그래밍 언어인 java와 관련된 Q&A를 수집했다. 데이터 개수는 90개로 이루어져 있고 실험에 사용한 데이터는 아래 깃허브 링크에서 확인 가능하다.¹

$$\text{정확도} = \frac{(1\sim 5\text{회 동안 옳은 답변수의 합}/5)}{\text{전체 데이터 수}}$$

수식 1. 정확도 공식

¹ https://github.com/jueun4136/chatbot_qna_data.git 참조

5.3 실험결과

두 챗봇의 정확도를 측정하기 위해 총 5번의 학습과 테스트를 통해 얻은 값을 사용했다. 정확도 공식은 수식1 과 같다. 5회의 학습과 테스트를 진행해 질문의 답변과 같은 예측 값을 제공한 개수를 다 더해 나누기 5를 하여, 답변을 옳게 예측한 5회 평균을 구했고, 이후 총 데이터수로 나눠 정확도를 측정했다. 원래 학습 데이터와 동일한 데이터로 테스트를 하면 안되지만 데이터 셋의 개수가 작다는 점과, 테스트 데이터를 구축하기 어려운 점을 고려해 학습데이터로 테스트를 진행하였다. 두 모델간 성능 비교 결과 KoBert를 사용한 Q&A챗봇이 Doc2Vec을 이용한 Q&A챗봇이 더 질문에 대한 답변 에 높은 정확도를 가지는 것을 확인했다.

모델	총데이터개수	옳게 답변한 개수	정확도
Doc2Vec	90	44	0.48
KoBert	90	66	0.73

표 2. 두 모델간 정확도 비교

6. 제한점

본 연구에서는 테스트 데이터를 구성하기 어려운 점과, 비지도학습인 Doc2Vec의 정확도를 평가하기 위해

학습데이터로 테스트를 진행했다는 아쉬움이 있다. 그리고 Doc2Vec 챗봇의 정확도가 KoBert 챗봇에 비해 낮은 정확도가 나오는데, 두 모델의 하이퍼파라미터는 실행하면서 최대한 가장 좋은 성능이 나오는 것을 선택하여 비교하였다. 그러나, 좀 더 체계적인 하이퍼파라미터 튜닝을 추가적으로 수행하는 것도 고민해 볼 수 있다,

7. 결론 및 향후 연구

본 연구의 실험 결과는 KoBert 를 이용한 Q&A 챗봇이 Doc2Vec 을 이용한 Q&A 챗봇보다 입력된 질문에 더 답변 제공 정확도가 73%대 48%로, 25% 높게 나타났다. KoBert 가 다량의 데이터로 pre-trained 된 모델이므로 같은 데이터에 대한 답변 예측에 성능차이가 난 것으로 생각된다.

향후연구로는 데이터를 추가로 수집한 후, 테스트 데이터를 구성하여 테스트데이터의 성능을 측정할 것이다. 또한 Doc2Vec 과 KoBert 두 모델의 교차검증과 많은 테스트를 통한 하이퍼파라미터 튜닝을 통해 성능 개선 연구를 진행할 예정이다. 최종적으로는 수업에서 실제 활용하는 수준으로 개발해 나가는 것이 목표인데, 이 챗봇을 실제 수업의 도구로 사용되려면 수업에 관한 많은 양의 Q&A 데이터가 구축되어야 한다. 하지만 일일이 질문과 답변의 데이터쌍을 구축하는 데는 많은 인력과 시간이 든다. 이를 개선하기 위한 한가지 방법으로 Bert 로 구현가능한 Task 중 Question 과 Answering Task 로 단락에 대한 질문에 대한 답변을 기능을 구현하는 것이다. Question 과 Answering Task 는 어떤 전체 문장을 학습시켜 문장에 나타난 질문을 하면 학습된 전체 문장을 통해 답변을 예측한다. 따라서 객체지향 프로그래밍 수업의 교재나, 객체지향 프로그래밍언어 커뮤니티 등을 크롤링 하여 여러 문장으로 이루어진 단락에 대한 Question 과 Answering 기능을 추가적으로 구축해보고자 한다.

레퍼런스

[1] 이용상, and 신동광. "코로나 19 로 인한 언택트 시대의 온라인 교육 실태 연구." 교육과정평가연구
 [2] 배영임, and 신혜리. "코로나 19, 언택트 사회를 가속화하다." 이슈 & 진단 (2020): 1-26.
 [3] 정천수, and 정지환. "포스트 코로나 19 언택트 시대 대응을 위한 AI 챗봇 구축방법에 관한 연구." 한국 IT 서비스학회지 19 (2020): 31-47.
 [4] 허주은, & 이선아. 수업 질문 자동응답을 위해 Doc2Vec 모델을 이용한 챗봇
 [5] Le, Quoc, and Tomas Mikolov. "Distributed representations of sentences and documents." International conference on machine learning. PMLR, 2014.

[6] KT Brain/KoBert
<https://github.com/SKTBrian/KoBert,2019>
 [7] 박아름, 이새봄, and 송재민. "인공지능 기반 챗봇 기술의 산업 적용 연구." 한국컴퓨터정보학회논문지 25.7 (2020): 17-25.
 [8] 김민지, 염지윤, 정혜원 and 임철일. (2021). 인공지능 챗봇의 교육적 활용 연구 동향 분석: 활동이론을 중심으로. 교육정보미디어연구, 27(2), 699-721.
 [9] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).
 [10] Devlin, Jacob, et al. "Bert: Pre-training odeep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

양방향 메트릭 선택 기반 비지도 학습 결함 예측

백인선^o, 권수진, 박성수, 박수은, 남재창

한동대학교 전산전자공학부

[\[21900349, 21900044, 21700264, 21900280, jcnam\]@handong.edu](mailto:[21900349, 21900044, 21700264, 21900280, jcnam]@handong.edu)

Bi-metric Selection Based Unsupervised Defect Prediction

Insun Baek^o, Sujin Kwon, Sungsu Park, Sueun Park, Jaechang Nam

School of Science and Electronic Engineering, Handong Global University

요 약

소프트웨어 결함 예측은 개발자들이 결함을 미리 파악하도록 도움으로써 개발의 효율성을 높일 수 있다. 지도 학습 결함 예측의 한계인 데이터 수집 문제를 극복하기 위해 비지도 학습 결함 예측 방법이 활발하게 연구되고 있다. 본 논문에서는 비지도 학습 결함 예측 방법 중 하나인 CLA/CLAMI를 확장한 접근법인 CLABI를 제안한다. CLAMI의 Metric selection 과정에서 CLA의 예측 값과 메트릭 값의 불일치 정도를 나타내는 지표인 MVS (Metric Violation Score) 를 사용하는데, 이 값이 낮으면 CLA의 예측 값과 메트릭이 유사한 성향을 보이고 높으면 반대의 성향을 보인다. 기존의 CLAMI는 MVS가 가장 작은 메트릭을 선택하는 방법을 사용하였지만, CLABI는 반대로 MVS 값이 가장 높은 메트릭을 선택하는 방법을 사용하여 CLAMI가 고려하지 못한 부분을 보완하고자 한다. 본 논문에서 제안한 CLABI의 성능을 평가하기 위해 27개의 데이터를 사용해서 CLA/CLAMI와 CLABI를 포함한 7개의 비지도 학습 모델을 비교 분석했다. CLABI는 CLAMI와 반대의 성향을 보이는 메트릭을 포함시킴으로써 더 정밀한 예측을 할 수 있을 것으로 예상했지만, 결과적으로 CLABI는 CLAMI보다 낮은 성능을 보였다. 기존의 CLA/CLAMI의 성능을 높이기 위해서는 데이터에 맞는 모델을 생성하기 위한 연구가 필요하다.

1. 서 론

결함 예측은 효율적으로 소프트웨어의 품질을 보장하기 위해서 소프트웨어의 결함을 미리 예측하는 것이다[5]. 결함 예측을 통해 개발자들에게 결함의 위치에 대한 정보를 제공함으로써 개발의 효율성을 높여줄 수 있다. 결함 예측에는 크게 지도학습 방법과 비지도 학습 방법이 있으며 일반적으로 지도 학습을 기반으로 하는 결함 예측 기법들이 주로 사용되어 왔다. 하지만 지도 학습을 기반으로 한 결함예측은 데이터의 라벨 정보 수집의 한계와 새로운 프로젝트에 대한 데이터 수집의 한계를 해결하기 위해 비지도학습 기반 결함예측이 활발히 연구되고 있다.

CLA/CLAMI는 비지도 학습을 기반으로 하는 결함예측 모델이다[1]. CLA/CLAMI는 데이터의 메트릭 값을 기반으로 결함정보가 없는 데이터의 결함을 예측하는 기법이다. 이것은 복잡할수록 결함이 있을 가능성이 높다는 가정으로부터 만들어졌다. 데이터의 메트릭 값이 클수록 결함이 있을 가능성이 높다고 볼 수 있다.

CLABI는 기존의 CLA/CLAMI의 예측을 더 정확하게 하기 위해 본 연구에서 제안하는 새로운 기법이다. 이 기법에서는 CLAMI의 가정과 반대되는 값을 갖는 메트릭 또한 고려하여 예측 모델 성능을 향상시킬 수 있도록 설계하였다.

따라서 본 연구는 CLA/CLAMI를 확장한 접근법인 CLABI를 제안하고 CLABI가 얼마나 향상된 결함 예측을 수행하는지 분석해 보고자 한다. 본 논문의 구성은 다음과 같다. 2장에서는 결함 예측 관련 배경지식과 비지도 학습 결함 예측 연구들에 대해 설명한다. 3장 연구 및 방법에서는 새로 제안하는 CLABI의 접근법과 실험에 사용된 요소들을 설명한다. 4장에서는 연구 결과를 보여주며 5장에서는 본 연구의 의의를 논의하고 6장에서는 논문에 대한 결론을 정리한다.

2. 배경지식 및 관련 연구

2.1. 지도학습 기반 결함예측

일반적으로 결함 예측 과정은 지도학습을 기반으로 진행된다. 즉 결함 여부를 알고 있는 상태에서 모델을 생성하고, 새로운 인스턴스에 대해 결함을 예측한다. 지도학습 기반의 결함 예측은 수집된 정보로부터 훈련 데이터를 생성하고 새로운 데이터의 결함 여부를 예측하는 것이다. 지도학습 결함 예측은 훈련 데이터와 예측하려는 데이터가 같은 프로젝트의 데이터인지, 다른 프로젝트의 데이터인지에 따라 WPDP (Within Project Defect Prediction), CPDP (Cross Project Defect Prediction)으로 구분할 수 있다. WPDP 는 같은 프로젝트 내의 데이터에 대해

예측을 하는 방법으로 새로운 프로젝트나 역사적 데이터가 적은 프로젝트에 대해서 예측 모델을 만들기 어렵다는 점에서 한계가 있다. 이를 해결하기 위해 CPDP 기법이 제안되었다. CPDP 는 다른 프로젝트로부터 훈련 데이터를 생성하여 예측 데이터에 적용하는 방법이다. 그러나 CPDP 는 훈련 데이터와 예측 데이터가 서로 다른 프로젝트로부터 생성되었기 때문에 데이터 이질성 문제가 존재하고 성능 또한 비교적 낮다[8].

2.2. 비지도학습 기반 결함예측

비지도 학습 결함 예측 기법은 결함 여부를 모르는 상황에서 데이터의 결함을 예측하기 위한 방법으로, WPDP와 CPDP의 한계들을 해결하기 위해 다양한 기법들이 제안됐다. CPDP의 문제를 해결하기 위한 시도로, 결함 예측에 기계 학습 방법을 이용하여 TNB (Transfer Naive Bayes)를 적용하는 연구와 TCA (Transfer Component Analysis)를 적용하는 연구 등이 있었다[6], [7]. 비지도 학습 결함 예측 방법 중 하나인 CLA/CLAMI는 복잡할수록 결함이 있을 가능성이 있다는 가정을 전제로 만들어진 접근법이다. CLA/CLAMI의 자세한 과정은 2.2.1. 절에서 설명한다. 이 접근법은 결함 정보가 없는 데이터로부터 예측 모델을 자동으로 만들 수 있다.

2.2.1. CLA/CLAMI 접근법

CLA/CLAMI는 메트릭 값을 사용하여 결함 정보를 모르는 데이터의 결함 여부를 예측하는 접근법이다[1]. CLA는 (1) Clustering, (2) LAbeling 과정을 통해, CLAMI는 CLA 과정 이후에 (3) Metric selection, (4) Instance selection 과정을 추가적으로 수행하여 결함을 예측한다. Clustering 과정에서는 각 메트릭들마다 특정 임계값보다 높은 메트릭 값을 찾는다. 이후 각 인스턴스에서 임계값보다 높다고 표시된 메트릭 값의 개수를 세고 해당 개수가 같은 인스턴스끼리 그룹을 만든다. 만들어진 그룹들을 크기에 따라 두 그룹으로 나누고 값이 높은 그룹을 결함, 작은 그룹을 비결함으로 label을 붙인다. CLA는 이렇게 붙은 label을 최종 결함 정보로 여긴다. CLAMI는 CLA 이후 두 단계를 더 수행하는 데, 먼저 Metric selection이다. Metric selection 단계에서는 Metric Violation Score (MVS) 가 가장 작은 메트릭만 선택하고 나머지는 결함 경향성을 따르지 않는 메트릭이므로 제거한다. MVS란 CLA에서 예측된 값과 메트릭의 값을 비교해서 일치하지 않는 정도를 수치로 나타낸 것이다. Instance selection 단계에서는 남은 인스턴스 중 CLA로부터 얻은 label의 성향을 따르지 않는 메트릭을 가진 인스턴스를 제거한다. 이 과정을 다 거치고 최종적으로 남은 인스턴스를 예측 모델로 사용하여 원래 데이터를 예측을 하는 것이 CLAMI이다.

2.2.2. CLAMI+ 접근법

CLAMI+는 CLAMI와 비교했을 때 Clustering과 Labeling 과정에서 차이점이 있다[3]. CLAMI는 각 메트릭 값에 대해 특정 임계값보다 값이 높은 메트릭 값의 개수를 세는 방식이었지만, Yan et al. 은 CLAMI에서 인스턴스가 얼마나 violate 되었는지 고려되지 않았다고 문제를 제기했다. 따라서 Clustering 을 할 때 특정 임계값보다 큰 메트릭 값의 개수를

세는 것이 아니라, 모든 메트릭에 대해 0과 1사이의 연속 값으로 표현했다. 즉, 각 메트릭 값이 특정 임계값과 얼마나 떨어져 있는지를 violation degree로 계산했다. CLAMI+는 violation degree를 계산하기 위해 시그모이드 함수 공식 (1) 을 사용했으며 모든 메트릭 값에 대해 해당 공식을 사용해서 violation degree를 구했다. 이후 각 인스턴스 별로 violation degree의 평균값을 계산하고 인스턴스들에 대해 0.5를 기준으로 그룹을 나눠서 Labeling을 진행했다. CLAMI+는 이와 같은 방법을 적용함으로써 기존의 CLAMI와 비교할 만한 결과를 얻었다[3].

$$P(V_{ij}) = \frac{1}{1 + e^{-(x_{ij}-N_j)}} \quad (1)$$

CLAMI+를 제안한 논문에서는 CLAMI 에 강화된 Clustering 방법을 적용한 결과를 보여주었다. 본 논문에서는 해당 Clustering 방법을 CLAMI 뿐만 아니라 CLA 와 3.1 절에서 설명하는 CLABI 라는 새로운 접근법에도 적용해서 CLA+ 와 CLABI+를 만들고 해당 결과를 정리했다.

2.2.3. CUC 접근법

Connectivity-based Unsupervised Classifier (CUC)는 spectral clustering을 기반으로 하는 비지도 학습 결함 예측 방법 중 하나로, CPDP가 가진 데이터 이질성 문제를 해결하기 위한 연구이다. CUC는 데이터를 결함 entity와 비결함 entity로 나누고 두 entity 사이의 연결성을 확인했다. 그 결과 결함 entity와 비결함 entity 사이의 연결성이 결함 entity 내에서 또는 비결함 entity 내에서의 연결성보다 약하다는 것을 발견했다[4].

3. 연구 및 방법

3.1. CLABI/CLABI+ 접근법

CLABI 는 본 연구에서 제안하는 CLAMI 를 변형한 접근법으로 (1) Clustering, (2) LAbeling, (3) Bi-metric selection, (4) Instance selection 과정을 거친다. Clustering, LAbeling 은 CLA/CLAMI 와 동일한 방법으로 진행되며, 각각 다른 메트릭으로 Metric selection 과 Instance selection 을 두 번 한다는 의미에서 Bi-metric selection 이라고 이름을 붙였다. CLABI 는 Metric selection 과정에서 MVS 가 가장 큰 메트릭을 선택해서 결함을 예측 하고 CLAMI 의 예측값과 비교해서 최종적인 예측값을 낸다.

두 접근법의 다른 점은 Metric selection 과정에서 CLAMI 는 MVS 가 가장 작은 메트릭을 선택했다면, CLABI 는 MVS 가 가장 큰 메트릭을 선택한다는 것이다. MVS 값이 낮으면 CLA 에서 예측한 값과 메트릭이 유사한 성향을 띤다는 것을 알 수 있고, 반대로 MVS 값이 높으면 반대의 성향을 띤다고 할 수 있다. CLABI 에서는 MVS 값이 높은 메트릭을 선택한 예측 모델을 추가로 만들어서 CLAMI 가 고려하지 못한 부분을 보완하려고 했다.

CLABI 에서는 CLA 과정 이후 Metric selection 에서 MVS 가 가장 높은 메트릭을 선택하고 Instance selection 과정을 거쳐 결함을 예측한다. CLAMI 에서 Instance selection 과정은

CLA 에서 예측된 label 과 실제 메트릭 값이 일치하지 않으면 해당 인스턴스를 제거한다. 즉, CLA 에서 예측된 label 이 결함인데 메트릭 값은 임계값보다 낮은 상황 또는 그 반대의 상황에서 인스턴스가 제거된다. 하지만 CLABI 는 반대의 경향을 보이는 메트릭을 선택했기 때문에 메트릭 값이 임계값보다 낮으면 결함이 있을 것이라는 기준으로 Instance selection 을 한다. 모든 과정을 거치고 최종적으로 남은 인스턴스를 사용하여 모든 인스턴스의 결함을 예측한다.

CLABI 의 예측값과 CLAMI 의 예측값을 비교해서 최종적인 CLABI 의 예측값을 생성한다. 어떤 인스턴스에 대해 CLABI 와 CLAMI 가 예측 결과가 다르다면 각 모델에서 예측값이 정해진 확률을 비교해서 더 높은 확률을 가진 예측값을 따른다.

CLABI+는 2.2.2. 절에서 언급된 CLAMI+ 방법이 CLABI 와 접목된 것이다. 즉 CLABI 의 Clustering 과 Labeling 과정에서 임계값보다 높은 메트릭 값의 개수를 세는 CLAMI 의 방식을 사용하지 않고 각 메트릭 값이 임계값에서 얼마나 떨어져 있는지를 구하는 방식이다.

3.2. CLABI 접근법 알고리즘

CLABI 알고리즘은 알고리즘 1 을 통해 자세히 살펴볼 수 있다. 알고리즘 1 은 MVS 가 높은 값을 이용해서 데이터 결함 예측에 사용될 Training set 과 Test set 을 생성하는 과정이며, 이 과정은 CLABI 의 Metric selection 과 Instances selection 에 해당 된다. Training set 과 Test set 은 하나의 데이터 내에서 생성되며 Test set 은 Metric selection 까지 진행한 인스턴스이고, Training set 은 Test set 에 추가적으로 Instance selection 까지 거친 인스턴스이다.

표 1. CLAMI 와 CLABI 의 가정 비교

<p>CLAMI 모델</p>	<p>가정: 메트릭 값이 높을수록 결함이 있을 가능성이 높다. 모델 생성 방법: Training set 과 Test set 을 생성할때 가장 낮은 MVS 를 가진 메트릭만 사용한다.</p>
<p>CLABI 모델</p>	<p>가정: 메트릭 값이 높을수록 결함이 있을 가능성이 높지만 그와 반대되는 성향을 가진 메트릭도 있을 수 있다. 모델 생성 방법: Training set 과 Test set 을 두개씩 생성한다. → 가장 낮은 MVS 값을 가진 메트릭을 사용한 set / 가장 높은 MVS 를 가진 메트릭을 사용한 set</p>

CLABI 는 각각 두개의 Test set 과 Training set 을 가지고 데이터를 예측한다. 첫번째 Test set 과 Training set 은 표 1. 에 나온 CLAMI 모델의 가정으로 만들어진 것이고 두번째 Test set 과 Training set 은 표 1. 에 나온 CLABI 모델의 가정을 기반으로 만들어진 것이다. CLAMI 의 가정으로 만들어진 두개의 set 은 값이 낮은 MVS 를 가진 메트릭을 사용했으며 CLABI

모델의 가정으로 만들어진 것은 MVS 값이 높은 메트릭을 사용했다.

알고리즘 1 에서 사용되는 변수는 9 개이다. 데이터의 전체 인스턴스를 저장한 *instances*, 각 metric 마다 높은 값과 낮은 값을 나누는 *cutoffs*, 결함을 표시하는 라벨인 *positiveLabel*, 같은 MVS 값을 가진 메트릭끼리 묶어서 정리한 hashmap 인 *MVS*, *MVShashmap* 의 key 값인 MVS 값만 저장한 *keys*, training set 을 저장하는 *trainingInstances*, Test set 을 저장하는 *testInstances*, 제거해야 하는 metric 을 나열한 *selectedMetricIndices*, 제거해야 하는 instances 를 나열한 *instIndicesNeedToRemove* 가 있다.

알고리즘 1 의 1 번 줄에서 *getMVS()* 함수를 사용해서 MVS 를 구한다. *getMVS()* 함수는 각 메트릭 값을 하나씩 접근해서 violation 개수를 센다. violation 이란 메트릭 값이 cutoff 값보다 높으나 CLA 를 통해 생성한 라벨이 비결함인 경우 또는 메트릭 값이 cutoff 값보다 낮으나 라벨이 결함인 경우를 의미한다. 메트릭 내에서 violation 이 있으면 메트릭의 violation 값이 증가된다. 이 방식을 모든 메트릭에 대해 적용해서 violation 을 저장한 배열을 만든다. 메트릭 별로 key 값을 violation 값을 저장하고 value 값을 배열 내에서 같은 violation 값을 가진 메트릭의 번호로 설정해서 *MVS* 라는 hash map 을 구성한다.

3 번 줄에서 *MVS* 값이 높은 메트릭으로 Training set 과 Test set 을 생성하기 위해 *MVS* hashmap 의 key 값만 *keys* 에 저장해서 내림차순으로 나열한다. 다음으로, for 문에서는 가장 큰 *keys* 값을 시작으로 Training set 과 Test set 을 생성한다. 만들어진 Training set 과 Test set 의 남아있는 인스턴스 개수가 0 이 아니면 해당 *trainingInstances* 과 *testInstances* 을 사용하지만 그렇지 않으면 다음으로 큰 key 값을 사용해서 각각의 set 를 새롭게 생성한다. *keys* 값에 해당하는 *MVS* hashmap 의 value 를 *selectedMetricIndices* 에 가져온다. *trainingInstances* 과 *testInstances* 에 전체 메트릭에서 *selectedMetricIndices* 를 제외한 메트릭을 제거한 메트릭만 저장한다.

8, 9 번 줄에서 Instance selection 과정을 진행한다. 일부 메트릭이 제거된 *trainingInstances* 의 메트릭에 대해서 *getHigherCutoffs()* 함수를 통해 각 메트릭의 cutoff 값을 구한다. 각 메트릭의 cutoff 값을 *getSelectedInstances()* 함수에 적용해서 불필요한 인스턴스를 정리한다. 그 방법은 *trainingInstances* 의 메트릭 값과 CLA 를 통해 생성된 라벨을 비교해서 violation 이 있는지 확인한 후 violation 있는 인스턴스가 *instIndicesNeedToRemove* 에 저장한다. *getInstanceByRemovingMetrics()* 함수로 *instIndicesNeedToRemove* 에 저장된 인스턴스를 제거하면 최종적인 Training set 이 생성된다.

CLAMI Training set 과 CLABI Training set 을 사용해서 각각의 Test set 을 예측하면 전체 인스턴스에 대해서 두 개의 예측 값을 가진다. 예측에 사용한 Training set 이 다르기 때문에 두 개의 예측값이 다를 수 있다. 두 개의 예측값이 서로 다를 경우에는 확률값을 사용해서 해결한다. 해당 방법은 알고리즘 2 에 나와있다. 알고리즘 1 과 다르게 알고리즘 2 에서 추가적으로 사용된 변수는 비결함을 표시하는 *negativeLabel*, CLAMI Training set 으로 예측 한 예측 값의 배열인 *CLAMIIdx*, CLABI Training set 으로 예측 한 예측 값의 배열인 *CLABIIdx*, 두개의

예측 값의 확률을 저장한 *probabilityOfCLABIIIdx* 와 *probabilityOfCLAMIIIdx* 가 있다.

알고리즘 2 의 1, 2 번 줄에서 사용된 *getProbabilityIdx()* 함수는 모든 인스턴스에 대해서 예측 값이 비결함이 될 확률과 결함이 확률 중 더 큰 값을 배열에 저장한다. CLAMI Test set 과 CLABI Test set 에 *getProbabilityIdx()* 함수를 적용해서 각 모델의 값의 확률값을 *probabilityOfCLABIIIdx* 와 *probabilityOfCLAMIIIdx* 에 저장한다.

각 예측 값을 저장한 *CLAMIIIdx* 와 *CLABIIIdx* 값이 다르면 해당 예측값에 대한 *probabilityOfCLAMIIIdx* 와 *probabilityOfCLABIIIdx* 를 비교해서 더 큰 확률값을 가진 예측값을 인스턴스의 최종 예측값으로 변경한다. *CLAMIIIdx* 와 *CLABIIIdx* 값이 같으면 기존의 예측값을 최종 예측값으로 설정한다. 이와같은 방안으로 모든 인스턴스에 대해서 최종 예측값을 구한다.

알고리즘 1. CLABI 에서 MVS 값이 높은 메트릭으로 Training set 과 Test set 생성

Require:

- all instances of the data *instances*;
- a list of percentile cutoff of the metrics *cutoffs*;
- a buggy label *positiveLabel*;
- a list of metrics with same MVS values *MVS*;
- a list of MVS values in descending order *keys*;
- instances to use as training set *trainingInstances*;
- instances to use as test set *testInstances*;
- instances to remove from all instances *instIndicesNeedToRemove*;
- metrics to be kept from all metrics *selectedMetricIndices*;

CLABI:

```

1: MVS = getMVS();
2: keys [] = MVS.keySet().toArray;
3: Arrays.sort(keys, Collections.reverseOrder());
4: for (Object key: keys) do
5:     selectedMetricIndices = MVS.get(key) +
                               (instancesByCLA.classIndex() + 1);
6:     testInstances, trainingInstances =
                               getInstanceByRemovingMetrics();
7:     cutoffs = getHigherCutoffs();
8:     instIndicesNeedToRemove = getSelectedInstances();
9:     trainingInstances = getInstanceByRemovingMetrics();
10:    if trainingInstances.numInstances() != 0 then
        break;
    
```

알고리즘 2. CLABI 두 개의 예측 모델에서 충돌나는 예측 값 해결방안

Require:

- all instances of the data *instances*;
- a buggy label *positiveLabel*;
- a clean label *negativeLabel*;

a list of predicted labels probabilities *probabilityOfCLAMIIIdx*;

a list of predicted labels probabilities *probabilityOfCLABIIIdx*;

a list of predicted labels of CLAMI *CLAMIIIdx*;

a list of predicted labels of CLABI *CLABIIIdx*;

Conflict:

```

1: probabilityOfCLAMIIIdx = getProbabilityIdx();
2: probabilityOfCLABIIIdx = getProbabilityIdx();
3: negativeLabel = getNegLable();
4: for i = 0 to instances.numInstances() do
5:     if !(CLAMIIIdx.get(instIdx).equals (CLABIIIdx.get(instIdx)) then
6:         if probabilityOfCLAMIIIdx.get(instIdx) <
           probabilityOfCLABIIIdx.get(instIdx) then
7:             if CLAMIIIdx.get(instIdx) == positiveLabel then
8:                 instancesByCLA.instance(instIdx).
                               setClassValue(negativeLabel);
9:             else if CLAMIIIdx.get(instIdx) == negativeLabel then
10:                instances.instance(instIdx).
                               setClassValue(positiveLabel);
11:            else instances.instance(instIdx).
                               setClassValue(CLAMIIIdx.get(instIdx));
12:            else instances.instance(instIdx).
                               setClassValue(CLAMIIIdx.get(instIdx));
    
```

3.3. 사용한 데이터

결함 예측 연구에서 자주 사용되는 데이터인 AEEEM 과 PROMISE 프로젝트를 사용했다[4]. 실험에서 사용된 데이터는 기존 비지도 학습 결함 예측 모델을 비교하는 논문[2]에서 사용된 데이터이다. AEEEM 데이터 5개[10]와 PROMISE 데이터 9개[11]에서 여러 버전을 선택하여 총 27 개의 데이터를 사용했다. 메트릭은 각 데이터별로 process metric 7 개, complexity metric 11 개, network metric 24 개를 추출해서 총 81 개를 사용한다. 표 2. 에서는 각 프로젝트의 이름 (project name), 프로젝트 버전 (version), 인스턴스 수 (#instance), 결함 비율 (%Def.)을 나타내고 있으며 모든 프로젝트는 자바를 사용해서 구현됐다.

3.4. 사용한 평가지표

다양한 접근법의 성능을 비교하기 위해 정밀도(precision), 재현율(recall), f-측정(F1), AUC(Area Under the Curve), MCC(Matthews Correlation Coefficient)를 사용했다. 정밀도는 결함으로 예측된 인스턴스 중에 얼마나 많은 인스턴스가 정확히 예측됐는지를 나타내는 지표이며, 재현율은 실제 결함이 있는 인스턴스 중에 얼마나 많은 인스턴스가 정확히 예측됐는지 나타낸다. F1 은 정밀도와 재현율의 조화평균으로, 결함 예측 연구에서 성능 측정할 때 자주 사용된다. MCC 는 실제 결함/비결함 비율과 예측된 결함/비결함 비율의 상관계수이다.

표 2. 실험에 사용된 데이터

Project	Version	# Instance	% Def.
ant	1.3	125	16.00%
	1.4	178	22.47%
	1.5	293	10.92%
	1.6	351	26.21%
camel	1.0	339	3.83%
	1.2	608	35.53%
	1.4	872	16.63%
	1.6	965	19.48%
ivy	2.0	352	11.36%
jedit	3.2	272	33.09%
	4.0	306	24.51%
	4.1	312	25.32%
	4.2	367	13.08%
	4.3	492	2.24%
log4j	1.0	135	25.19%
poi	2.0	314	11.78%
synapse	1.0	157	10.19%
	1.1	222	27.03%
	1.2	256	33.59%
Velocity	1.6	229	34.06%
xerces	1.2	440	16.14%
	1.3	453	15.23%
Equinox Framework	3.4	324	39.81%
Eclipse JDT Core	3.4	997	20.56%
Apache Lucene	2.4.0	691	9.26%
Mylyn	3.1	1862	13.16%
Eclipse PDE UI	3.4.1	1497	13.96%

MCC 는 -1 에서 1 사이의 값을 가지며 1 에 가까울수록 모델의 예측이 우수하다고 판단되고, 0 인 경우에는 모델의 성능이 랜덤 예측과 같다. CLAMI, CLAMI+, CLABI, CLABI+에서 MCC 를

이용해서 더 정확히 성능을 검증했다. AUC 는 모든 임계값에서 성능을 보여주는 그래프인 ROC 의 곡선 아래 영역을 의미한다. AUC 가 높다는 것은 모델의 성능이 우수하다는 것을 나타낸다.

3.5. 연구 질문

본 논문에서 CLA/CLAMI 를 확장해서 새로 제안된 CLABI 의 성능과 특성을 비교 분석하기 위해 다음과 같은 연구 질문(RQ)을 설정한다.

- RQ1. 새로 제안된 CLABI 의 성능이 CLA/CLAMI 의 성능을 향상 시키는가?
- RQ2. 어떤 요소들이 CLABI 의 성능에 영향을 미치는가?

RQ1 에서, 실험데이터에 대한 CLABI 의 성능을 다른 비지도 학습 모델의 성능과 비교한다.

RQ2 에서, CLABI 에 사용된 데이터와 CLABI 의 특성을 정리해서 CLABI 의 성능에 영향을 끼치는 데이터의 요소들을 찾고 분석한다.

4. 연구 결과

기존 비지도 학습 결함 예측 모델을 비교하는 논문[2]에서 사용된 27 개의 데이터를 사용해서 베이스 라인인 CUC, CLA, CLA+, CLAMI, CLAMI+로 CLABI 와 CLABI+의 성능을 함께 비교했다.[2]. 대부분의 성능 지표를 모든 모델에 적용할 수 있었다. 하지만 AUC 값을 계산하기 위해서는 머신러닝 classifier 를 사용해야 하기 때문에 머신러닝을 사용하지 않는 CLA, CLA+, CUC 는 AUC 를 계산할 수 없었다. 표 3. 는 27 개의 데이터를 사용해서 실험한 성능 값의 평균을 계산한 것이다.

CLA, CLA+, CLAMI, CLAMI+, CLABI, CLABI+는 81 개의 메트릭을 사용하여 실험을 할 수 있지만 CUC 는 메트릭 값이 모두 0 인 메트릭을 가진 데이터에 대해서는 예측을 하지 못한다. CUC 는 메트릭의 중앙값과 표준편차를 구해서 실험을 하기 때문에 메트릭 값이 모두 0 인 메트릭에 대해서는 중앙값과 표준편차가 0 이 되므로 실험에서 결과를 얻을 수 없다[4]. 따라서 CUC 에 대해서는 메트릭 값이 모두 0 인 linesdel_max (maximum number of deleted lines) 메트릭을 제거하고 80 개의 메트릭을 사용해서 실험을 진행했다.

또한 실험과정에서 교차 검증 (cross validation) 을 하지 않았다. 본 연구에서 실험 비교 대상 기법이 모두 비지도 학습 기반 기법이므로 Training set 과 Test set 을 구분하여 교차 검증을 할 필요가 없고 전체 데이터를 사용해서 실험을 하는 것이 공정하다.

표 3. 비지도 학습 결합 예측 성능 평균

	Precision	Recall	F1	AUC	MCC
CLA	0.320	0.751	0.428	-	0.285
CLA+	0.292	0.819	0.410	-	0.261
CLAMI	0.304	0.758	0.411	0.695	0.261
CLAMI+	0.295	0.757	0.402	0.694	0.246
CLABI	0.283	0.745	0.390	0.685	0.221
CLABI+	0.297	0.757	0.403	0.690	0.248
CUC	0.300	0.778	0.410	-	0.259

표 4. 는 비지도 학습 결합 예측 방법들 사이의 유의성을 측정할 p-value 값을 나타낸 것이다. 비모수 통계검정 방법인 Wilcoxon Signed Rank Test[12]을 사용해 두 집단 사이의 p-value 를 계산했고 p-value 와 기준값(0.05)을 비교해서 유의성을 판단한다. Wilcoxon Signed Rank Test 를 통해서 얻은 p-value 값이 0.05 이하 일 때 CLA 와 유의미한 성능 차이가 있다고 판단했다. 또한 비모수 효과크기 측정방법인 Cliff’s δ [13] 를 사용해 유의성의 크기를 분석했다. 효과크기는 negligible ($|\delta| < 0.147$), small ($|\delta| < 0.33$), medium 963 ($|\delta| < 0.474$), and large ($0.474 \leq |\delta|$) 기준으로 분류했다[13]. 표 4. 에서 각각의 p-value 옆에 N: Negligible, S: Small, M: Medium, L: Large 로 표시했다. CLA는 5개의 성능 평가 지표 중 3개의 지표에서 가장 높은 값을 냈기 때문에 CLA 를 통계 검증의 기준으로 잡았다. 5 개의 평가 지표 중 정밀도, 재현율, F1, MCC 을 비교하여 각 모델과 CLA 의 유의성을 검증했다.

표 4. 비지도 학습 결합 예측 방법 통계검증

	Precision	Recall	F1	MCC
CLA+	< 0.001(S)	< 0.001 (M)	0.001(N)	0.001 (N)
CLAMI	0.001(N)	0.374 (N)	0.003 (N)	0.007 (S)
CLAMI+	< 0.001(N)	0.786 (N)	< 0.001 (N)	0.001 (S)
CLABI	< 0.001(S)	0.095 (N)	< 0.001 (S)	< 0.001 (M)
CLABI+	< 0.001 (N)	0.530 (N)	< 0.001 (N)	0.001 (S)
CUC	0.003 (N)	0.427 (S)	0.039 (N)	0.077 (N)

정밀도와 F1 을 기준으로 했을 때는 6 개의 모든 결합 예측 방법에서 p-value 가 0.05 를 넘는 값이 없기에, 모든 모델이 CLA 와 유의미한 차이가 있다. 재현율을 기준으로 했을 때는 CLA+를 제외한 나머지 5 개의 예측방법의 p-value 값이 모두 0.05 이상이기에 CLA 와 유의미한 차이가 없다. MCC를 기준으로 했을 때는 CUC 의 p-value 값이 0.05 이상이므로 CLA 와

유의미한 차이가 없으며, CUC 를 제외한 모든 모델은 CLA 와 유의미한 차이가 있는 것으로 보인다.

대부분의 경우 효과크기는 Negligible 또는 Small 이지만 CLA+와 CLABI 는 Medium 으로 나타난다. 이를 통해 대부분의 모델이 CLA 와 비교했을 때 효과크기가 무시할 만하거나 작은 수준으로 유의미한 차이가 있다는 것을 알 수 있다. 또한 CLA+의 재현율과 CLABI 의 MCC 는 효과크기가 Medium 으로 유의미한 차이가 있다.

RQ1. 새로 제시된 CLABI 의 성능이 CLA/CLAMI 의 성능을 향상 시키는가?

총 27 개의 데이터 중 5 개의 지표에 대해서 대부분 CLAMI 의 성능이 더 우수하게 나왔지만 7개의 데이터 (ant-1.3, camel-1.2, Eclipse JDT Core, Eclipse PDE UI, jediit-3.2, 4.0, Lucence)에서 5 개의 지표 중 3 개 이상의 지표가 CLABI 가 더 우수하게 나왔다.

표 3.에서 확인할 수 있는 것처럼 CLABI 의 성능이 CLA, CLAMI 보다 높을 것이라 예상했지만 예상과 다르게 낮은 성능을 얻었다. CLABI는 정밀도, 재현율, F1, AUC, MCC에서 가장 낮은 값을 보였다. CLABI+는 정밀도에서 CLA+, CLAMI+에 비해 높은 값을 가졌으며, 재현율에서는 CLA 보다 높은 값을 냈고 CLAMI+와 같은 값을 가졌다. 뿐만 아니라, CLABI+는 F1 과 AUC 에서 CLABI 다음으로 낮은 값을 냈으며, MCC 에서는 CLAMI+보다 높은 값을 결과로 얻었다.

각 지표를 기준으로 가장 높은 값을 살펴보면 정밀도, F1, MCC 에서 CLA 가 가장 높은 값을 냈고, 재현율에서는 CLA+가 가장 높은 값을 냈다. AUC는 CLAMI 가 가장 높은 값을 가졌다. CLA 를 기준으로 모든 모델의 p-value와 effect size 를 구한 표 4. 를 살펴보면 정밀도, F1, MCC 에서 CLABI 의 p-value 값이 0.001 이하로 CLA 와 통계적으로 유의미한 차이를 보인다. 하지만 CLABI 의 effect size 는 MCC 를 제외한 나머지 지표에서 small 이나 negligible 을 가지므로 유의미한 차이가 크지 않다.

RQ2. 어떤 요소들이 CLABI 의 성능에 영향을 미치는가?

CLABI 의 성능에 영향을 미치는 요소가 CLABI 의 가정에 맞는 메트릭의 존재여부와 전체 인스턴스 수에 대한 가장 높은 MVS 값의 비율이다. CLABI 는 표 1. 에서 설명한 것처럼 두개의 모델을 생성한다. 첫번째 모델은 CLAMI 가 생성하는 모델과 동일하지만 두번째 모델은 CLAMI 의 가정과 반대되는 모델이다. CLABI 의 두번째 모델은 가장 높은 MVS 값을 사용한다. 가장 높은 MVS 을 사용할때 해당 MVS 가 전체 인스턴스의 90%도 되지 않을 수 있는 문제점이 있다. 대부분의 데이터로 CLAMI 모델을 만들때 사용되는 가장 낮은 MVS 는 전체 인스턴스 수의 10%이하로 낮은 값을 가진다. 그러므로 가장 높은 MVS 값도 전체 인스턴스의 90%이상이어야만 CLAMI 와 반대되는 모델이라고 할 수 있으며 그렇지 않은 경우 CLABI 의 성능이 CLAMI 보다 낮을 가능성이 높다. CLABI 의 성능에 영향을 미치는 두번째 요소는 데이터의 트레이트이다. 데이터에 CLAMI 의 가정과 반대되는 ‘메트릭 값이 낮으면서 결합이 있는 메트릭’이 존재하면 가장 높은 MVS 값의 비율이 높아지고 CLABI 의

성능이 높아질 가능성이 있지만 그렇지 않으면 CLABI 의 성능이 CLAMI 의 성능보다 낮을 가능성이 높다. 예를 들어 그림 1 에 나열된 Equinox Framework 메트릭 중에는 결함과 비결함의 경계를 정확히 나누면서 메트릭 값이 낮을수록 결함이 있는 메트릭이 거의 없기 때문에 해당 데이터에 대한 CLAMI 의 성능이 CLABI 보다 더 높을 수 밖에 없다. 5.1 절을 통해 두 요소에 대한 더 자세한 내용을 확인 할 수 있다.

5. 논의

5.1. 결과 분석

CLABI 는 양방향 메트릭 선택 기반 비지도 학습 결함 예측 방법으로 CLAMI 를 확장해서 제안하는 접근법이다. 따라서 결과 분석을 위해 CLAMI 와 CLABI 를 주로 비교했다. CLABI 는 메트릭 값이 높을수록 결함이 있을 것이라는 CLAMI 의 가정과 반대되는 메트릭으로 만든 모델을 포함시킴으로써 더 세밀한 예측을 할 수 있을 것이라고 예상했다. 즉, 예측 모델을 데이터에 최대한 적합하게 만들려는 시도였다는 점에서 의미 있다. 하지만, CLABI 는 CLAMI 보다 낮은 성능 결과를 얻었고, 그 이유는 다음과 같이 분석된다. CLABI 는 Metric selection 부분에서 MVS 값이 가장 높은 메트릭을 사용해서 예측을 한 후 CLAMI 와 다르게 예측한 값을 수정해 주는 방식이다. 그런데 MVS 값이 가장 높은 메트릭이라고 해서 *메트릭 값이 낮을수록 결함이 있는 성향*을 보여주는 것은 아니다. 반대의 성향을 뚜렷하게 보여주기 위해서는 MVS 값이 메트릭의 총개수에 가까워야 하기 때문이다. 예를 들어 메트릭 값이 총 10 개인 데이터에서 가장 큰 MVS 값이 6 인 경우, 해당 메트릭이 CLAMI 와 반대의 성향을 따라간다고 할 수 없다. MVS 값이 6 이라는 의미는 10 개의 메트릭 값 중에 6 개의 메트릭 값이 CLA 가 예측한 값과 반대의 의미를 가진다는 것이기 때문이다.

표 5. Minimum MVS, Maximum MVS 와 F1 측정 성능

data	Min MVS (%)	Max MVS (%)	F1 of CLAMI	F1 of CLABI
Equinox Framework	5.86	66.98	0.6828	0.6797
log4j-1.0	5.93	75.56	0.6122	0.5607
Mylyn	6.87	50.27	0.2913	0.2894

실험에 사용한 데이터의 대부분이 CLABI 가 CLAMI 에 비해 F1 이 낮은 것을 확인했고, 그 중 세 개 데이터의 Minimum MVS 와 Maximum MVS 를 표 5. 에 정리했다. 표 5. 에서 각 MVS 값을 전체 인스턴스의 수로 나눠서 백분율로 나타냈다. 표 5. 에 명시된 데이터의 Min MVS 는 5%에 가깝고, Max MVS 는 80%도 되지 않는 것을 볼 수 있다. Min MVS 를 사용해서 Metric selection 을 한 CLAMI 가 CLABI 보다 성능이 더 높으므로 메트릭 값이 높을수록 결함이 있다는 CLAMI 의 가정에 가깝다는 것을 알 수 있다. 반대로 Max MVS 가 전체

메트릭의 약 95%에 가까운 메트릭을 가진 데이터를 CLABI 에 적용한다면 CLAMI 의 모델을 보완할 가능성이 있다.

또한 실험에 사용한 데이터 중 하나인 Equinox Framework 의 모든 메트릭을 정규화한 값에 대해 결함, 비결함 분포를 상자 그림으로 그려 실제 메트릭의 분포를 확인해 보았다 [그림 1]. 각 메트릭 값에서 메트릭 값의 평균을 뺀 후 표준편차로 나눠서 정규화했다. M72, M74 처럼 메트릭 값에서 결함과 비결함 범위가 비슷한 메트릭은 있으나, 메트릭 값이 높으면 비결함이고 낮으면 결함인 것으로 명확히 나뉘는 메트릭은 없었다. 대부분의 메트릭이 M35, M47 과 같이 값이 높으면 결함이고 반대이면 비결함 인 것으로 보였다.

플러스(+) 버전들의 성능 또한 원래의 버전에 비해 떨어지는 것을 확인할 수 있었다. CLA+, CLAMI+, CLABI+는 원래의 버전들에 비해 정밀도가 모두 떨어지지만 재현율은 CLA+, CLABI+에서 올랐다. 이는 원래의 버전에서 보다 더 많은 인스턴스를 결함으로 예측하기 때문이다. 이와 같은 이유로 재현율이 오른 경우는 있지만 정밀도와 F1 은 떨어졌다.

5.2. 향후 연구

기존의 CLA/CLAMI 의 성능을 높이기 위한 시도로 CLABI 라는 새로운 방법을 제안했다. 하지만 앞서 설명한 것처럼 성능 향상은 없었다. 성능을 높이기 위해서는 프로젝트 데이터에 맞는 모델을 생성하기 위한 개선이 필요하다. 프로젝트 데이터마다 가장 적합한 예측 모델을 만들기 위해 percentile cutoff 를 조정하는 방법이 있다. 현재는 Clustering 과정에서 기본 percentile cutoff 인 50% 기준으로 median 보다 큰 값과 작은 값을 구분한다. 하지만, 데이터마다 특성이 다르기 때문에 각 프로젝트 데이터마다 가장 적합한 percentile cutoff 가 존재할 것이고 이것을 찾으면 성능을 더 향상시킬 수 있는 방법으로 쓰일 수 있을 것이다. CLA/CLAMI 의 성능 향상을 위한 또 다른 방법으로 예측을 하기 위한 모델을 생성하기 전에 프로젝트 데이터의 특징을 미리 찾는 방법이 있다. 해당 프로젝트 데이터의 특징이 특정 모델을 사용해서 예측을 하기 적합한 지 확인하는 것이다. 이렇게 함으로써 프로젝트 데이터의 특성을 사전에 파악하고 그 특성에 적합한 모델을 사용하도록 할 수 있을 것이다.

이를 위해 메트릭과 라벨의 상관관계를 파악하고, 상관관계가 높은 것으로 측정된 메트릭들 간의 상관관계도 계산하여 선택적으로 접근법을 적용하는 방법이 있다. 하지만 메트릭과 라벨의 상관관계를 파악하기 위해서는 사전에 라벨 정보를 알고 있어야 한다는 한계가 있다. 비지도 학습 결함 예측에서는 라벨 정보를 미리 알 수 없기 때문에 이 방법을 사용하기 위해서는 더 많은 연구가 필요하다.

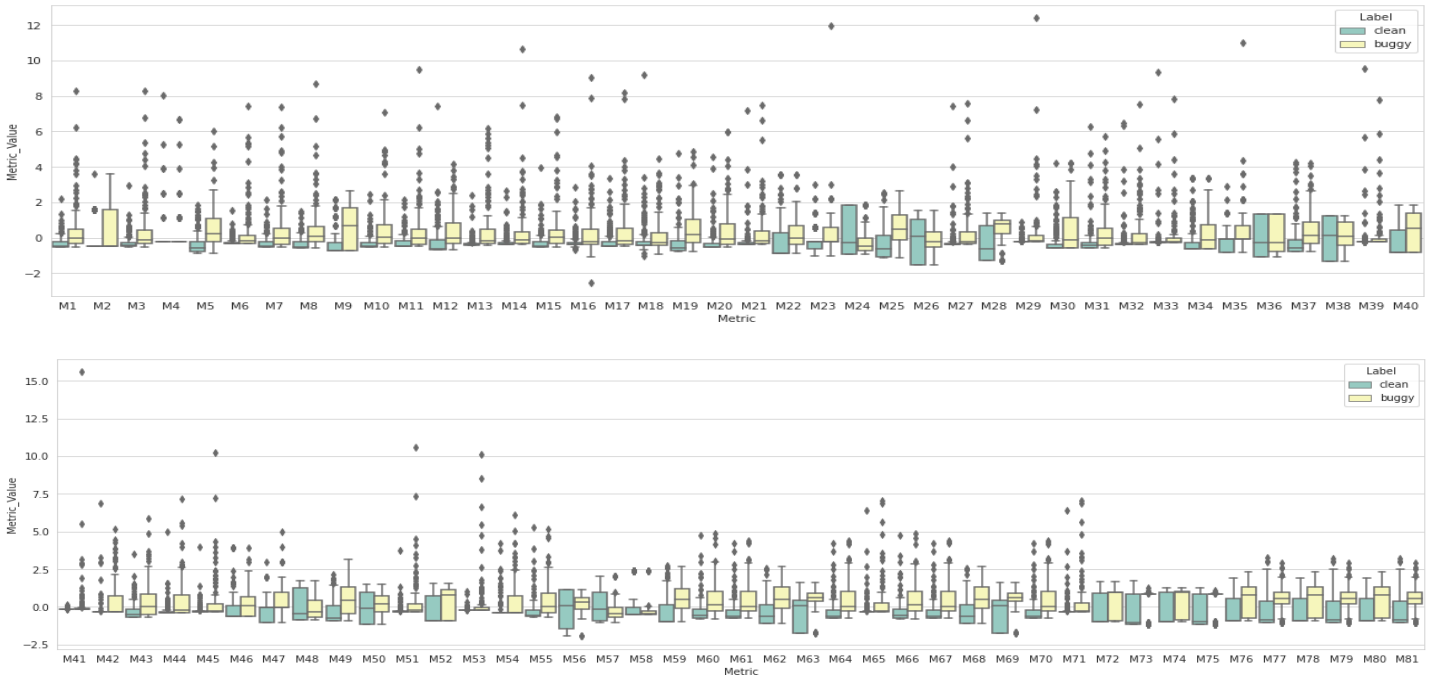


그림1. Equinox Framework 메트릭의 결함, 비결함 분포 상자그림

5.3. 타당성 검증

1. **데이터의 타당성:** 실험에 사용된 데이터는 14 개의 오픈 소스 프로젝트 중 27 개의 버전을 사용했다. 모든 데이터는 자바 언어로 구현된 프로젝트로 생성되었으며, 다른 언어(C, C++, Python 등)는 사용되지 않았다는 점이 실험 결과의 편향성을 높일 수 있다. 뿐만 아니라 각 데이터 셋은 process metric, complexity metric, network metric 만 선택적으로 사용했다. 향후 연구에서는 데이터의 언어의 종류와 메트릭의 종류를 추가해서 더 보편화된 결과를 가져올 수 있을 것이다.
2. **모델 성능 평가 지표의 타당성:** 실험의 성능을 평가하는 지표로서 정밀도, 재현율, F1, AUC, MCC 를 사용했다. 이 지표들이 결함예측 성능 평가에 자주 사용되는 지표이긴 하지만 Cost-effectiveness 와 같은 다른 지표로 평가를 하면 다른 결론이 관찰될 수 있다.

존재한다는 것을 알 수 있었다. 그 사실을 바탕으로 CLABI 모델을 만들었지만 선행 모델들보다 뛰어난 결과를 얻지 못했다. CLABI 에서는 값이 낮을수록 결함이 있는 메트릭을 찾고자 했지만 조건에 맞는 메트릭이 많이 존재하지 않았다. 결론적으로 메트릭 값이 낮을수록 결함을 가진 메트릭이 있는 데이터를 가지고 실험을 한다면 CLABI 가 CLA/CLAMI 와 CLAMI+가 고려하지 못한 부분을 보충할 수 있을 것이다.

데이터의 특성을 고려해서 CLABI 라는 새로운 모델을 제안한 것처럼 앞으로 기존 CLA, CLA+, CLAMI, CLAMI+의 성능을 높이기 위한 더 나은 방법을 연구해볼 수 있다. CLA, CLA+ CLAMI, CLAMI+는 고정된 percentile cutoff 를 사용해서 예측 모델을 만들지만 데이터마다 알맞은 percentile cutoff 를 적용한다면 메트릭 내에서 결함과 비결함의 경계를 더 정확하게 구분할 수 있을 것이다. 결론적으로 앞으로의 비지도 학습 결함 예측 모델 개발은 메트릭의 성향과 데이터의 특징을 분석하는 것에 초점을 맞춘다면 더 성능이 좋은 결함 예측 모델이 개발될 것이다.

6. 결론

소프트웨어 공학의 연구분야 중 하나인 결함예측의 한계를 해결하기 위해 비지도 학습 기법들이 연구되고 있다. 여러 비지도 학습 결함 예측을 비교 분석하기도 하고[2], FRUGAL[9]과 같은 세미 지도 학습 모델이나 CLAMI+ 같은 비지도 학습 모델도 개발됐다. 본 논문은 기존에 알려진 다양한 비지도 학습 결함 예측방법들을 분석하고 새로운 예측방법인 CLABI 를 제안했다.

선행 연구인 CLA/CLAMI, CLAMI+는 메트릭 값이 높을수록 결함이 있을 것이라는 가정으로 연구가 진행됐다. 선행 연구를 분석하면서 선행연구의 가정에 반대된 성향을 가진 메트릭도

“이 논문은 과학기술정보통신부의 소프트웨어중심대학 지원사업(2017-0-00130)과 한국연구재단의 지원(No. 2021R1F1A1063049)을 받아 수행하였음.”

Reference

- [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] Xu, Zhou, et al. "A comprehensive comparative study of clustering-based unsupervised defect prediction models." *Journal of Systems and Software* 172 (2021): 110862.
- [3] Yan, Meng, et al. "Self-learning Change-prone Class Prediction." *SEKE*. 2016.
- [4] Zhang, Feng, et al. "Cross-project defect prediction using a connectivity-based unsupervised classifier." 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). IEEE, 2016.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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 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.
- [9] Tu, Huy, and Tim Menzies. "FRUGAL: Unlocking SSL for Software Analytics." *arXiv preprint arXiv:2108.09847* (2021).
- [10] D'Ambros, Marco, Michele Lanza, and Romain Robbes. "Evaluating defect prediction approaches: a benchmark and an extensive comparison." *Empirical Software Engineering* 17.4 (2012): 531–577.
- [11] Jureczko, Marian, and Lech Madeyski. "Towards identifying software project clusters with regard to defect prediction." *Proceedings of the 6th international conference on predictive models in software engineering*. 2010.
- [12] Woolson, Robert F. "Wilcoxon signed-rank test." *Wiley encyclopedia of clinical trials* (2007): 1–3.
- [13] Romano, Jeanine, et al. "Appropriate statistics for ordinal level data: Should we really be using t-test and Cohen'sd for evaluating group differences on the NSSE and other surveys." *annual meeting of the Florida Association of Institutional Research*. Vol. 13. 2006.

기업형 블록체인 플랫폼인 하이퍼레저 베수의 기술적 배경

임영택*, 박민하**, 주성호**, 배병일**

동국대학교 전자전기공학부*, 한전 전력연구원**

wyde00@dgu.ac.kr, [mhpark, shju-1052, b.i.bae]@kepcoco.kr

Technical background of Hyperledger Besu for enterprise blockchain platform.

Yeongtaek Lim*, Minha Park**, Seongho Ju**, Byeongil Bae**

Dongguk University*, KEPCO Research Institute**

요 약

최근 새롭고 획기적인 프로젝트들이 추가되면서 블록체인 기술은 끊임없이 확장되고 있다. 특히, 이더리움과 하이퍼레저 커뮤니티의 지속적인 협력이 기업형 블록체인 플랫폼 개발에 큰 성과를 거두고 있다. 현재 기업형 블록체인에서 가장 활발하게 기술 확산을 하고 있는 플랫폼 중 하나는 하이퍼레저 베수(Hyperledger Besu)이다. 하이퍼레저 베수는 퍼블릭 블록체인과 프라이빗 블록체인 네트워크 모두를 위해 기업 친화적으로 설계되었으며, 네트워크를 구축할 수 있는 자바 기반의 이더리움이면서 프라이빗 네트워크 구축은 이더리움보다 쉽다. 또, 모든 블록체인의 특징들은 이더리움을 그대로 계승하면서 이더리움의 명령어를 그대로 사용 가능하다는 장점이 있다. 본 논문에서는 하이퍼레저 베수를 중심으로 관련된 연구들을 조사하였다.

1. 서 론

2009년, 사토시 나카모토는 블록체인 기술에 기반한 최초의 탈중앙화 암호화폐인 비트코인[1]을 고안했다. 암호화폐는 블록체인을 활용한 최초의 애플리케이션이었지만 은행업무, 건강관리, 서플라이 체인 등과 같은 여러 분야에서 사용되어왔다[2]. 서로 다른 엔티티 간에 데이터를 저장하고 공유할 때, 중앙 집중화된 데이터 베이스에는 몇 가지 제한사항이 있었는데, 이러한 제한 중 하나가 바로 단일 장애 지점이다. 외부로부터 공격이 발생하면 전체 시스템에 장애가 발생할 수 있다는 것이다. 또한 소유권 문제, 투명성 문제 등으로 인해 데이터를 서드파티에 저장하는 것이 허용되지 않을 수도 있다[3]. 이를 해결하기 위한 방법 중 하나는 데이터를 저장하는데 있어서 신뢰할 수 있는 엔티티를 선택하는 방법이다. 그러나 각 엔티티들은 서로 다른 정책을 가지고 있기 때문에 데이터를 저장할 엔티티에 대해서 합의하기 어렵다는 문제점이 있다. 블록체인은 분산원장 기술을 활용하여 이러한 한계를 극복한다. 블록들이 서로 연결되어 체인을 형성한다는 것은, 새로운 블록을 기록할 때 이전 블록 정보에 대한 해시값이 포함된다는 의미이다. 해시값은 암호학적 해시 함수를 통해 계산되는데, 해시 함수는 결과값으로부터

입력값을 유추하기 어렵다. 따라서 블록체인에 기록된 내용은 위변조하기 매우 어려워지고 데이터를 신뢰할 수 있게 된다. 또한, 공격자가 블록을 변경하려고 하면 변경하려는 내용이 체인을 통해 전파되어 참여자들이 인식할 수 있다.

비트코인 이후 블록체인 기술을 단순 가상통화 거래 이외의 다른 분야로 확산 및 활용하고자 하는 움직임이 나타나기 시작하였다. 비트코인을 통해 신뢰성 보장 체계는 구축되었지만 시스템을 확장하거나 개선하기 위한 협력이 이루어지지 않는 등의 문제점이 계속 대두되었으며, 이에 따라 프로그래머 Vitalik Buteri이 ‘차세대 스마트 컨트랙트와 탈중앙화된 어플리케이션 플랫폼’이라는 제목의 이더리움 백서를 발표하게 된다. 2세대 블록체인이라 불리는 이더리움이 1세대인 비트코인과 결정적으로 다른 점은 스마트 컨트랙트 기능을 수행할 수 있다는 것이다. 스마트 컨트랙트는 트랜잭션 로직을 제어하며 중개자가 이를 실행할 필요가 없는 자동화된 프로그램이다[4]. 이를 통해 미리 작성된 절차 또는 프로그래밍된 규칙에 따라 자동으로 실행되며 분산 처리된다. 이때, 신뢰 보장 프로세스는 비트코인의 작업증명을 그대로 계승했기 때문에 별도의 신뢰 확인 절차는 필요치 않다. 블록체인에 스마트 계약 기능을 더하면 더 많은 명령어를 탑재하여 솔리디티(Solidity) 언어로

프로그래밍이 가능하다. 즉, 이더리움이 제공하려는 것은 튜링 완전(Turing-Complete) 프로그래밍 언어가 심어진 블록체인이며 설계가 가능한 모든 종류의 계약을 스마트 컨트랙트로 만들 수 있는 것이다. 이를 통해 모든 계약을 자동으로 하고, 이를 위한 다양한 분산 애플리케이션(DApp) [11] [12]도 만들고 실행할 수 있다. 이더리움 블록체인은 이러한 스마트 컨트랙트 기능을 더함으로써 간단한 컴퓨터 프로그램을 만들고 실행할 수 있는 플랫폼의 역할을 수행하고 있다.

블록체인은 분산원장 기술 또는 네트워크를 이용하는 방법이 사용처마다 다르고 활용 목적 또한 제각각이다. 이에 따라 크게 퍼블릭 블록체인(Public Blockchain)과 프라이빗 블록체인(Private Blockchain)으로 나뉘며 각 블록체인마다 특징이 있다. 퍼블릭 블록체인은 개방형 블록체인으로 누구나 트랜잭션을 생성할 수 있다. 통상적인 블록체인이라 하면 퍼블릭 블록체인을 지칭하며 비트코인과 이더리움이 대표적인 예이다. 퍼블릭 블록체인은 누구나 참여 가능하며 제한이 없고 별도의 허가(permission)도 없으며, 접속만 가능하다면 누구나 플랫폼에 가입해 노드가 될 수 있다. 트랜잭션 내역이 참여한 모든 노드에게 공개되며 네트워크내의 모든 노드가 이를 상호 검증하고 거래를 승인하기 때문에 신뢰도가 높다. 하지만 모든 참여자가 트랜잭션 내역을 남기고 공유하면서 처리 속도가 느려진다는 단점이 있다. 프라이빗 블록체인은 폐쇄형 블록체인으로 서비스 제공자의 승인을 받아야만 참여할 수 있다. 네트워크에 참여할 수 있는 사람을 제한하는 접근 제어를 기반으로 작동되며 거래에 참여하는 주체만이 해당 거래에 대한 내역을 갖게 되고 다른 주체들은 접근할 수 없다. 이러한 특성 때문에 주로 기업에서 활용하여 엔터프라이즈 블록체인(Enterprise Blockchain)이라고도 한다.

[표 1] 블록체인 네트워크 비교

	퍼블릭	프라이빗
접근성	누구나 참여 가능	필요에 따라 임의로 제한 가능
장점	높은 안정성, 신뢰성	높은 효율성, 확장성
대표 사례	비트코인, 이더리움	Hyperledger Fabric, R3 Corda

비트코인, 이더리움과 같은 퍼블릭 블록체인 외에도 하이퍼레저(Hyperledger)와 같은 프라이빗 블록체인에 관한 연구개발 또한 활발하게 이루어지고 있다.

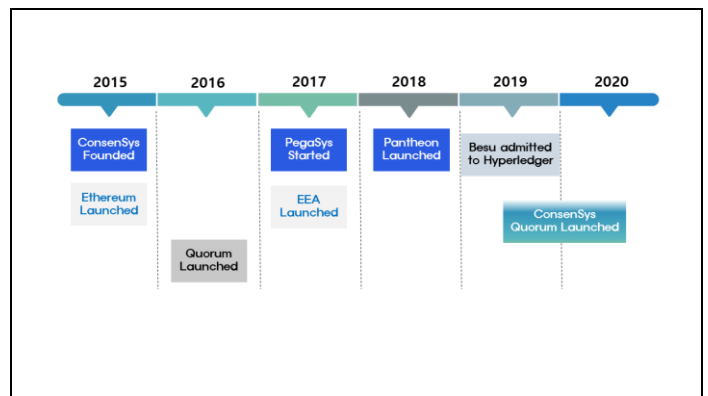
하이퍼레저는 현재 프라이빗 블록체인의 선두주자로서 대중적인 서비스가 가능하도록 다양한 프레임워크와 기능을 추가하고 있다. 그 중 가장 널리 알려진 프레임워크가 바로 하이퍼레저 패브릭(Hyperledger Fabric)이다. 현재, 엔터프라이즈 블록체인 분야에서 근본적인 변화가 일어나고 있는데 그 중심에는 하이퍼레저 베수(Hyperledger Besu)가 있다. 하이퍼레저 베수는 여러 기업용 프라이빗 프로젝트들을 뛰어넘을 잠재력을 지녔다는 평을 받으며 엔터프라이즈 블록체인 계에서 새롭게 인기를 끌고 있다.

2. 하이퍼레저 베수(Hyperledger Besu)

이전에 판테온(Pantheon)으로 알려졌던 하이퍼레저 베수 [5]는 개발자들이 합의 알고리즘, 네트워킹, 스토리지와 같은 분산 원장의 다양한 측면을 스스로 처리할 수 있도록 모듈형 아키텍처를 사용하여 설계된 자바 기반 이더리움 클라이언트이다. 이더리움 클라이언트는 특정 사용 사례를 다루기 위해 이더리움 프로토콜을 구현하기 위해 설계된 소프트웨어이다. 하이퍼레저 베수는 이더리움 네트워크를 사용하는 프로젝트 간 상호 운용성과 표준화된 개발 방식을 촉진하는 이더리움 기업 연합(Enterprise Ethereum Alliance, EEA) 규격 [13]을 준수한다.

2.1 베수의 등장배경

비교적 최근까지 프라이빗 DLT(Distributed Ledger Technology)의 세계는 국제 시장의 대부분을 차지하고 있는 몇 업체만을 중심으로 양분되었으며 Corda, Quorum 또는 Hyperledger Fabric과 같은 플랫폼이 기업 프로젝트에서 가장 널리 사용되었다.



[그림 1] 베수의 설립과정

[그림 1]에서는 베수가 채택되기까지의 연혁을 간략하게 표현했다. 2015년 초, 캐나다 출신 기업가인 Joseph Lubin은 이더리움 블록체인에서 동작하는

탈중앙화 소프트웨어 서비스와 애플리케이션을 개발하기 위한 소프트웨어 파운드리로서 ConsenSys를 설립했다[14]. Lubin은 사토시 나카모토의 비트코인 백서와 분산형 시스템이 "권력의 비대칭을 재조정할 수 있는" 잠재력에 영감을 받아 초창기 블록체인 커뮤니티를 파고들었다. 그는 동료 Vitalik Buteri와 함께 이더리움을 만들고 출시하는 데 주요한 역할을 했으며, 이후 ConsenSys를 설립함으로써 이더리움 기술을 중심으로 생태계를 구축하고자 했다. ConsenSys는 최고의 기업가, 컴퓨터 과학자, 프로토콜 엔지니어, 소프트웨어 개발자 및 엔터프라이즈 트랜스퍼 전문가를 고용하여 세계적인 입지를 다져왔다. 이후로 꾸준히 기업 및 정부를 위한 블록체인 산업의 개발자 도구, 분산형 애플리케이션 및 솔루션을 구축하였다. 2018년, PegaSys 프로토콜 엔지니어링 팀 (ConsenSys 소속)이 판테온을 개발하기 시작했다. 개발 당시, 판테온은 생산적인 기업 환경에 적합한 이더리움 클라이언트를 개발하는 것이 주요 목적이었다. 2019년 2월, 버전 1.0이 출시되었고 몇 달 뒤인 2019년 8월, 판테온은 베수라는 새로운 이름으로 하이퍼레저 에코시스템에 공식 채택되었다. 이 합병을 통해 세계에서 가장 큰 두 블록체인 개발 커뮤니티인 이더리움과 하이퍼레저 사이에 첫번째 협업 시스템이 구축되었다. 퍼블릭 네트워크와 프라이빗 또는 컨소시엄 네트워크 모두에서 엔터프라이즈급 애플리케이션 개발이 가능한 하이브리드 접근 방식을 통해 하이퍼레저 베수는 블록체인 계에서 빠르게 채택될 수 있었다.

2.2 Ethereum과 Hyperledger Besu

2015년에 출시된 이더리움은 분산형 애플리케이션 개발을 위한 오픈 소스 퍼블릭 블록체인 기반 분산 플랫폼이다[6]. 원래 이더리움은 PoW(작업증명) 기반 합의 프로토콜인 Ethash를 구현하는 퍼블릭 비허가형 블록체인 기반 플랫폼이다. 이더리움의 특징은 튜링 완전성을 갖춘 확장형 언어로 스마트 컨트랙트를 지원하며 이를 기반으로 다양한 응용 구현이 가능하다는 점이다. 이더리움 가상머신(EVM)을 도입하여 어떤 노드라도 프로그래밍 언어와 상관없이 어떤 프로그램이든 실행할 수 있게 한다.

이더리움 클라이언트인 하이퍼레저 베수의 엔터프라이즈 기능은 다양한 이더리움 엔터프라이즈 프로젝트 간의 공통 인터페이스 생성을 포함하여 EEA 클라이언트 사양의 요구사항을 준수하도록 설계되었다. 베수 클라이언트는 합의 알고리즘 등 블록체인 핵심 기능을 쉽게 구현하고 업그레이드할 수 있도록 모듈성이 뛰어난 것이 특징이다. 하이퍼레저 베수의 기능성 또한 큰 장점 중 하나이다. 웹소켓 및 HTTP 기반 API와 함께 CLI를 제공하여 이더리움

네트워크에서 노드의 적절한 실행, 유지보수, 모니터링을 보장한다. 클라이언트 API는 스마트 컨트랙트, DApp 개발, 운영 관련 사용 사례 및 배치를 포함한 일반적인 이더리움 기능을 적절하게 지원한다. 이더리움 노드에 연결하여 이더리움 네트워크와 상호 작용하는 데 사용되는 Web3j[15], 스마트 컨트랙트를 보다 쉽게 테스트하고 배포할 수 있게 해주는 프레임워크인 Truffle[16] 및 Remix[17]와 같은 도구는 이러한 활동을 지원하는 데 도움이 된다. 또한 베수의 클라이언트는 일반적인 JSON-RPC API[24] 구현에 대한 지원을 보장한다. 따라서 사용자는 기존 에코시스템 및 구축환경을 구성하기 위해 새로운 것을 익힐 필요가 전혀 없다. 클라이언트와 관련된 Zero knowledge[7] 프로토콜은 Aztec[8] 프로토콜과 함께 프라이버시를 보장하여 개인간 거래를 촉진할 수 있다. 그리고 각 노드는 Private Transaction Manager 역할을 하는 Orion[9] 노드를 가짐으로써 Private Transaction을 송신하거나 수신할 수 있다. Orion을 통해 트랜잭션을 이더리움 상에 공개할 때 원하는 대상으로만 배포를 제한하여 일부 노드만 정보를 가질 수 있도록 하는 것이다.

2.3 Quorum과 Hyperledger Besu

Quorum은 J.P. Morgan에 의해 금융 목적으로 개발되었지만 어떤 산업이든 사용할 수 있는 이더리움 기반 허가형 블록체인이다[10]. 이더리움 포크인 Quorum은 프라이버시와 효율적인 합의 수단을 위해 더 나은 옵션들을 혼합한 프로젝트이며 Privacy와 Permissioning 등과 같은 기능을 제공한다.

2020년 8월 25일 ConsenSys는 J.P. Morgan이 개발한 블록체인 Quorum을 인수한다고 발표했다. ConsenSys와 Quorum 모두 이더리움을 기반으로 블록체인을 운영하는데, 이는 스마트 컨트랙트를 최초로 활용한 프로토콜이다. 하지만 여기에 중요한 차이점이 있다. ConsenSys는 이더리움 블록체인을 중심으로 오픈소스 인프라를 구축한다. 즉, 누구나 소스 코드에 관여하거나 Ether 암호화폐처럼 누구나 사용할 수 있는 애플리케이션과 같은 프로젝트를 개발할 수 있다. 또한 Quorum은 이더리움을 기반으로 하지만 프라이빗 방식으로 구축된다. 이에 따라 사용자는 개발자 또는 그룹 내에 속한 사용자로 인정되며 강력한 보안 및 감사 기능을 제공받는다. ConsenSys의 Charles D'Haussy 이사는 Interoperability(상호운용성)을 위해 퍼블릭 블록체인과 프라이빗 블록체인을 결합하는 것이 이번 인수의 비전이라고 말했다. 오늘날 은행들은 5개, 10개 이상의 블록체인을 운영하고 있는데, 이러한 블록체인 간 많은 연결고리가 있고 이를 유지하는 것이 각 업계에게 굉장히 큰 부담이 되고 있는 상황이다. 이것이 ConsenSys가 프라이빗 블록체인과 퍼블릭

블록체인이 상호운용될 필요가 있다고 생각하는 이유 중 하나다. 현재 Quorum으로 실행되는 애플리케이션은 영향을 받지 않으므로 실제로 상호운용성이 무엇을 의미하는지는 아직 명확하지 않지만 근본적인 목표는 여러 블록체인을 위한 단일 액세스 지점을 구축하여 은행들이 업계 전반의 노드와 연결 시 어려움을 겪지 않도록 하는 것이다. 기업은 ConsenSys Quorum을 사용하여 각 상황에 맞게 필요한 엔터프라이즈급 네트워크를 제공할 수 있다. Apache 2.0 라이선스를 선호하거나 베수 기반의 다른 벤더들과 협력하는 기업들이 퀴럼 기반의 다양한 네트워크에 쉽게 가입할 수 있게 된 것이다. 이로 인해 타사 벤더 또는 내부 개발자의 제품 모듈을 통합하여 고성능 맞춤형 애플리케이션을 구축할 수 있다.

하이퍼레저 베수는 이더리움의 공공 인프라를 활용하려는 기업들을 위해 Quorum의 퍼블릭 메인넷과 완벽하게 호환되며 다양한 암호경제나 퍼블릭 데이터 중심 애플리케이션을 지원한다. 이 외에도 기존 금융기술과 분산형 금융기술의 융합을 주도해 어디서나 모두가 접근 가능하게 하는 Codefi[18] 제품군이나 이더리움에 접속해 DApp을 구축하는 블록체인 개발 툴 Infra[19] 등 기타 ConsenSys 소프트웨어 제품도 지원한다. 또한, 엔터프라이즈 Quorum 네트워크의 사용 권한과 개인정보 보호기능을 통해 컨소시엄은 체인에서 거래할 수 있는 것과 없는 것, 거래를 볼 수 있는 사람, 거래 수수료가 필요한지 등을 구분하기 위한 사용자 지정 규칙을 정의할 수 있다.

Quorum은 개인 트랜잭션의 암호화, 암호해독, 배포를 도와주는 Private Transaction Manager인 Tessera[20]를 사용하여 네트워크가 상용 네트워크의 수신 트랜잭션을 통해 일부 또는 모든 당사자와의 기밀 거래를 수행할 수 있도록 하며 프라이빗 트랜잭션의 수명과 유효성을 보장한다. ConsenSys가 Quorum을 인수하게 됨에 따라 Private Transaction Manager로 Orion만을 사용하던 베수에서도 Tessera를 지원하게 되었으며 최근에는 Orion의 기능들이 Tessera로 합병되었다.

3. 베수의 주요 기능

현재 하이퍼레저 베수는 EEA 표준에 따라 운영된다. EEA 사양은 이더리움 환경에서 서로 다른 폐쇄형 프로젝트와 오픈소스 프로젝트 간에 공통 인터페이스를 구축하기 위한 표준 벤치마크이다. 즉, 퍼블릭 블록체인인 이더리움에서 쓰이는 블록체인 기술을 기업 환경에서도 사용할 수 있게 하는 것이다. EEA의 프로토콜을 구현함으로써 이더리움에 Permissioning, Privacy와 같은 엔터프라이즈 기능들을 추가로 확장했다. 하이퍼레저 베수에서 사용할 수 있는 다른

기능들은 다음과 같다.

3.1 EVM

하이퍼레저 베수의 이더리움 가상 머신은 튜링 완성형 가상머신으로, 이더리움 블록체인에서 트랜잭션을 통한 스마트 컨트랙트 구축 및 실행을 지원한다. EVM은 코드를 실행하기 전에 높은 수준의 스마트 컨트랙트 코드를 EVM 바이트코드로 컴파일하는 역할을 한다.

3.2 Consensus Mechanisms

하이퍼레저 베수는 두 가지의 구별되는 알고리즘의 형태로 서로 다른 합의 메커니즘을 사용한다. 합의 알고리즘은 블록 생산, 블록 유효성 검사 및 트랜잭션 유효성 검사와 관련된 작업에 따라 다르게 쓰인다. 핵심적인 두 가지 합의 알고리즘은 바로 권한증명(PoA)과 작업증명(PoW)이다. 권한증명 프로토콜은 거래 참여자들이 서로를 알고 있는 경우에 이상적이고, 작업증명(또는 Ethash)은 메인넷 이더리움과 관련된 마이닝 활동에 초점을 맞춘 사용 사례에 적합하다.

[표 2] PoA와 PoW의 비교

	PoA	PoW
블록체인 종류	Permissionless	Permissioned
확장성	높음	낮음
장점	빠른 속도 및 높은 보안성	누구나 참여가능
단점	탈중앙성 부족	낮은 효율

3.3 Storage Features

스토리지는 로컬 노드 레벨에서 체인 데이터를 지속할 수 있는 RocksDB key-value 데이터베이스를 제공한다. RocksDB는 오픈소스 key-value 데이터베이스로 로컬 접근만 가능한데, 베수에서는 로컬 컴퓨터로만 접근할 수 있기 때문에 개인정보 보안상 이 데이터베이스를 채택했다. RocksDB에 의해 저장된 데이터는 두 부분으로 나뉘는데, 블록체인 데이터와 World state 데이터와 같은 특정 하위 범주로 분류된다. 블록체인 데이터에는 데이터 체인을 구성하는 블록 헤더가 포함되어 있어 블록체인 상태의 암호화 검증에 도움이 된다. 각 블록 헤더는 stateRoot 해시를 통해 World state를 가리키며 World state 데이터는 서로 다른 주소의 계정에 대한 매핑을 나타낸다. 외부 소유 계정에는 Ether 잔고가 있으며 스마트 컨트랙트 기반 계정에는 실행 파일과 코드가 포함될 수 있다.

3.4 P2P Networking

하이퍼레저 베수는 이더리움의 devp2p 네트워크 프로토콜을 활용하여 클라이언트 간의 통신을 가능하게 한다. 또한 합의 알고리즘 IBFT 2.0에 특화된 하위 프로토콜도 제공한다. 피어 검색에는 UDP 프로토콜이 사용된다[5].

3.5 User-facing APIs

EEA JSON-RPC API와 메인넷 이더리움 API의 형태로 된 User-facing API이다. 애플리케이션 측면에서 HTTP JSON-RPC, 웹소켓 JSON-RPC 또는 GraphQL을 사용하여 블록체인 네트워크와 상호 작용할 수 있다. 베수로 DApp을 개발할 때 Truffle, Remix 또는 Web3j와 같은 편리한 도구를 사용할 수 있다.

3.6 Monitoring

모니터링은 이더리움 퍼블릭 네트워크에서 작업을 가능하게 하는 하이퍼레저 베수의 기능에서 가장 중요한 강점 중 하나이다. 노드 및 네트워크의 성능을 모니터링하는 기능으로, 하이퍼레저 베수는 debug_metrics JSON-RPC API 메서드 또는 Prometheus를 사용하여 노드 성능을 모니터링한다. 반면, EthStats Network Monitor 및 Block Explorer와 같은 Alethio 툴은 네트워크 성능을 모니터링하는 데 도움이 된다[25]. 또한, Grafana를 사용하여 수집된 데이터를 시각화할 수도 있다. Prometheus와 Grafana 모두 오픈소스 툴로 다양한 데이터 소스를 자유롭게 시각화 할 수 있다.

3.7 Privacy

Privacy는 주로 거래에 관련된 당사자 간 프라이버시를 유지하는 것을 말한다. 외부 당사자가 거래에 관련되었거나 거래 내용 또는 직접 주고받는 거래 당사자의 목록에 접근할 수 없도록 보장한다. Private Transaction Manager는 거래의 프라이버시를 보호하는 데 있어 엄청난 강점이다. 이더리움 클라이언트의 다른 버전처럼 이더리움 메인넷과 테스트넷에서 정상적으로 동작할 수 있을 뿐만 아니라, 하이퍼레저 베수를 통해 사용자는 전용 네트워크를 만들고 해당 네트워크에서 전용 트랜잭션을 수행할 수도 있다. 본래의 프라이버시 그룹은 Static membership이었으며 그룹이 만들어지고 나면 기존 그룹에 구성원을 추가하거나 제거하는 옵션이 없었다. 구성원이 추가하거나 제거할 필요가 있을 때마다 새로운 개인 정보 그룹을 만들어야 했고, 그 후에는 이전 그룹의 모든 거래 내역이 손실되었다. 그러

나 베수가 업데이트됨에 따라 유연한 개인 정보 보호 그룹을 통해 새 구성원을 추가하고 기존 구성원을 제거할 수 있게 되었다. 구성원이 추가되면 이전에 있던 그룹의 모든 개인 트랜잭션은 그룹 기록에 액세스할 수 있도록 새 구성원에게 전송된다. 이를 통해 조직이 비즈니스 요구사항에 맞게 그룹에 가입하고 탈퇴함에 따라 그룹 구성원 자격을 새롭게 업데이트할 수 있게 되었다.

3.8 Permissioning

네트워크에 노드 허가(node permissioning) 또는 계정 허가(account permissioning)를 허용함으로써 허가된 네트워크에 의해 특정 노드와 계정만 참여할 수 있다. 사용자는 기본 설정에 따라 네트워크에서 계정 권한 또는 노드 권한을 활성화할 수 있다. 또한 하이퍼레저 베수를 통해 로컬뿐 아니라 on-chain permissioning도 받을 수 있다. 로컬 권한 부여는 노드 수준의 권한 부여에 구성파일을 활용하는 반면에, on-chain permissioning은 네트워크 수준에서 스마트 컨트랙트를 사용하는 것을 포함한다. 즉, 스마트 컨트랙트를 사용하여 노드, 계정 및 관리자 허용 목록을 저장하고 관리할 수 있게 되는 것이다.

4. 합의 알고리즘

하이퍼레저 베수는 PoW, PoA(IBFT, IBFT 2.0, Etherhash, Clique)를 포함한 여러 개의 합의 알고리즘을 포함한다. 이더리움 구현으로 PoW(작업 증명)은 당연한 것이지만, 기업 관련 프로젝트의 경우에는 PoA(권한 증명) 옵션이 더 사용 가능성이 높을 것이다. 베수에는 Clique와 IBFT 2.0이라는 두 가지 PoA 옵션이 있으며, 두 가지 모두 고유한 특성, 장점 및 단점이 있다.

IBFT 2.0은 Quorum과 베수의 새로운 IBFT 버전이다. 우선 IBFT 2.0은 유효한 모든 블록이 메인 체인에 포함되도록 하는 즉각적인 Finality를 가지고 있다. 즉각적인 Finality는 트랜잭션의 유효성이 확인되면 블록체인에 영구적으로 추가되어 나중에 폐기되지 않는다는 개념이다. 과거의 트랜잭션들이 절대 수정될 수 없다는 것이다. 이러한 특징으로 PoA 합의 과정에서 분기 없이 빠르게 체인에 블록을 추가된다. 즉, 트랜잭션의 최종 결정을 기다릴 필요가 없이 트랜잭션 처리 및 테스트가 간소화되므로 노드당 필요한 작업량이 적어지는 효과가 있다. 만약 Orion 노드의 트랜잭션을 검증한다고 하면, Finality가 필요하기 때문에 Private Transaction을 처리하는 데 IBFT 2.0이 유용하다. 이와 대조적으로 Clique는 즉각적인 Finality가 결여되어 있어 합의 프로토콜을 사용하는

구현체들은 포크에 취약하다. Clique는 블록 생성 속도에 관해서 IBFT 2.0을 능가한다. 그러나 검증자 수가 증가함에 따라 Clique 네트워크에서 포크가 발생할 위험이 커진다. 한편, IBFT 2.0 네트워크는 검증자가 더 추가될수록 블록 생성시간이 계속해서 증가하게 된다는 단점이 있다.

5. 하이퍼레저 베수 활용방안

최근 전산업에 걸쳐 주목받고 있는 기술 중 하나인 Non Fungible Token(NFT) [23]은 이더리움의 스마트 컨트랙트에서 파생된 일종의 암호화폐이다. NFT는 이더리움 개선 제안서 EIP-721 [21]에서 처음 제안되었고 EIP-1155 [22]에서 추가로 개발되었다. NFT는 비트코인과 같은 고전적인 암호화폐와 본질적인 특징이 다르다. 비트코인은 모든 코인이 동일하고 구별이 불가능한 표준 코인이다 반대로, NFT는 유사성(동일성, 비동일성)으로 교환될 수 없는 독특한 방식으로 사물이나 누군가를 식별하는데 적합하다. 구체적으로 말하면, 스마트 컨트랙트에 NFT를 활용하면 창작자가 동영상, 이미지, 아트, 이벤트 티켓 등의 형태로 디지털 자산의 존재와 소유권을 쉽게 증명할 수 있다. 하이퍼레저 베수와 스마트 컨트랙트 컴파일, 배포 및 이진 관리 기능이 내장되어 있는 Truffle [16]을 이용하여 손쉽게 ERC-721을 배포할 수 있을 것이다. 빠르게 기업 간의 컨소시엄을 구축하거나, 이를 통해 스마트 컨트랙트 내에서 NFT를 추적하고 전송하여 기업 간 컨소시엄을 빠르게 구축하거나 각 기업에서의 기자재 관리, 비품의 반입 및 반출 또는 Lifecycle의 관리 등 여러 방면으로 활용 가능할 것으로 기대된다.

게다가, 하이퍼레저 베수는 Privacy 기능을 제공해준다. 이를 통해 당사자 간의 거래를 비공개로 유지하여 당사자는 거래 내용, 발송 당사자 또는 참여 당사자 리스트에 액세스할 수 없도록 할 수 있다. 가령, 물류와 같은 Supply chain에서 고객들은 배송에 필요한 정보를 제외한 나머지 정보를 비공개로 하여 개인정보가 새어나가는 것을 막을 수 있을 것이다. 또한, Orion [9]을 통해 기업은 컨소시엄을 구성하더라도 정보은닉을 통해서 불필요한 정보유출을 막을 수 있다.

6. 결론

본 논문에서는 하이퍼레저 베수가 등장하게 된 배경부터 특징 및 기능까지 조사하였다. 하이퍼레저 베수는 Quorum과 연관되어 복잡한 발전과정을 거쳤다. 하이퍼레저 베수가 도입되면 기업은 기존 이더리움 네트워크만의 장점을 퍼블릭 네트워크 상에서 활용할 수 있다. 무엇보다도 베수는 일반적인 시스템 요구사항을 통해 간단하고 유연한 설정과 설치를

보장한다. 네트워크 사양을 명확하게 파악하면 하이퍼레저 베수에 대한 시스템 요구 사항을 보다 쉽게 설정할 수 있을 것이다. 하이퍼레저 베수는 이미 세계 수준의 엔터프라이즈급 기술을 구축하는 강력한 도구이자 인상적인 기술이다. 향후 허가형 네트워크에서의 퍼블릭 이더리움 원장이 다수의 서로 다른 클라이언트(Besu, Geth, Parity 등)과 연결되는 것과 같은 방식으로 서로 다른 클라이언트와 함께 실행될 것으로 기대된다. 허가와 합의 메커니즘이 표준화되면 각기 다른 기업 블록체인 클라이언트로 구성된 허가형 분산원장 운영이 가능해질 것이다.

참고문헌

- [1] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 21260.
- [2] B. A. Tama, B. J. Kweka, Y. Park, and K. H. Rhee, "A critical review of blockchain and its current applications," in 2017 International Conference on Electrical Engineering and Computer Science (ICECOS), 2017, pp. 109-113.
- [3] X. Xu et al., "A Taxonomy of Blockchain-Based Systems for Architecture Design," in 2017 IEEE International Conference on Software Architecture (ICSA), 2017, pp. 243-252.
- [4] Singla, V., Malav, I. K., Kaur, J., & Kalra, S. (2019, January). Develop leave application using blockchain smart contract. In 2019 11th International Conference on Communication Systems & Networks (COMSNETS) (pp. 547-549). IEEE.
- [5] Hyperledger Besu, September 2, 2021, [Online]. Available: <https://besu.hyperledger.org>
- [6] A next-generation smart contract and decentralized application platform, Ethereum Whitepaper, December 16, 2021, [Online]. Available: <https://ethereum.org/en/whitepaper>
- [7] Goldreich, O., & Oren, Y. (1994). Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1), 1-32.
- [8] Aztec protocol Whitepaper, January 24, 2019, [Online]. Available: <https://github.com/AztecProtocol/AZTEC/blob/master/AZTEC.pdf>
- [9] Consensys. Orion. November 1, 2020, [Online]. Available: <https://docs.orion.consensys.net/en/stable>
- [10] Quorum whitepaper, August 24, 2018, [Online]. Available: <https://github.com/ConsenSys/quorum/blob/master/docs/quorum%20Whitepaper%20v0.2.pdf>

- [11] Lee, K., James, J. I., Ejeta, T. G., & Kim, H. J. (2016). Electronic voting service using block-chain. *Journal of Digital Forensics, Security and Law*, 11(2), 8.
- [12] Riemann, R., & Grumbach, S. (2017). Distributed protocols at the rescue for trustworthy online voting. *arXiv preprint arXiv:1705.04480*.
- [13] Enterprise Ethereum Alliance, February 2017, [Online]. Available: <https://entethalliance.org/wp-content/themes/ethereum/img/intro-eea.pdf>
- [14] Radziwill, N. (2018). Blockchain revolution: How the technology behind Bitcoin is changing money, business, and the world. *The Quality Management Journal*, 25(1), 64–65.
- [15] Etheruem. Web3j, [Online]. Available: <https://docs.web3j.io>
- [16] Ethereum. Truffle, [Online]. Available: <https://trufflesuite.com/docs/truffle/overview>
- [17] Ethereum. Remix, [Online]. Available: <https://remix-ide.readthedocs.io>
- [18] ConsenSys. Codefi, March 2, 2021, [Online]. Available: <https://consensys.net/blog/codefi/what-is-the-codefi-blockchain-application-suite>
- [19] ConsenSys. Infura, [Online]. Available: <https://infura.io/about>
- [20] ConsenSys. Tessera, [Online]. Available: <https://docs.tessera.consensys.net>
- [21] Entriken, W., Shirley, D., Evans, J., & Sachs, N. (2018). Erc-721 non-fungible token standard. *Ethereum Foundation*.
- [22] Radomski, W., Cooke, A., Castonguay, P., Therien, J., Binet, E., & Sandford, R. (2018). Eip 1155: Erc-1155 multi token standard. *Ethereum, Standard*.
- [23] Wang, Q., Li, R., Wang, Q., & Chen, S. (2021). Non-fungible token (NFT): Overview, evaluation, opportunities and challenges. *arXiv preprint arXiv:2105.07447*.
- [24] Ethereum. JSON-RPC, [Online]. Available: <https://ethereum.org/ko/developers/docs/apis/json-rpc>
- [25] Alethio, [Online]. Available: <https://besu.hyperledger.org/en/1.2.3/Concepts/AlethioOverview>

다중 피부병변 검출 심층학습을 위한 피부병변 합성 기반 데이터 증강 기법

박범진¹, 이영찬¹, 장현재², 유원상¹

¹ 선문대학교 정보통신공학과, ² ㈜에프앤디파트너스

parkbj24@sunmoon.ac.kr, lyc19960618@sunmoon.ac.kr,

frank.zhang@fndpartners.info, wyou@sunmoon.ac.kr

A method of data augmentation based on skin lesion synthesis for deep learning of multiple skin lesion detection

Bumjin Park¹, Youngchan Lee¹, Hyunjae Zhang², Wonsang You¹

¹Department of Information and Communications Engineering, Sun Moon University

²F&D Partners Inc.

요 약

피부 확대경 영상 진단을 위한 심층학습 모델에서 피부병변의 자동 검출은 필수적인 전처리 과정이다. 기존의 피부병변 검출 기법은 대부분 단일 병변에 적용되어 왔고, 여러 질환의 다중 병변이 혼재하는 피부 임상사진에서는 검증된 바 없다. 그러나, 다중 병변의 피부영상 데이터의 부족으로 인하여 심층학습 모델의 훈련에 어려움이 있다. 본 논문에서는 생성적 적대 신경망(GAN)을 사용하여 다중 피부병변 합성 영상을 생성함으로써 다중 병변 검출 심층학습 모델 훈련을 위한 학습 데이터를 증강하는 기법을 제안한다. 제안된 기법은 (1) 피부영상에서 병변영역을 분할하는 단계와 (2) 분할된 병변영역을 피부에 혼합한 합성영상을 생성하는 단계로 구분된다. 실감성 등급 스코어(MOS)에 따른 정성적 평가 결과, 병변-피부 혼합 합성영상은 GP-GAN과 DIB를 사용하였을 때 MOS는 각각 3.72 ± 1.21 와 3.27 ± 1.34 로 유의미한 차이를 보였다($p=0.015$). YOLO를 사용한 다중 병변 검출 실험에서, DIB를 사용하여 생성된 다중 병변-피부 합성영상에 대해 평균 정확도(IoU)와 평균 정밀도(mAP)는 각각 91.6%와 77.8%였다. 실험결과는 제안된 GAN 기반 병변-피부 합성 기법이 피부영상 다중 피부병변 검출 심층학습을 위한 데이터 증강에 사용될 수 있는 가능성을 보여준다.

1. 서 론

피부 확대경 영상 진단을 위한 심층학습 모델은 최근 급속도로 발전해 왔다. 피부병변의 질환분류를 위한 영상진단 모델에서 피부병변의 자동 검출은 필수적인 전처리 과정이다. 영상처리에 의한 방법에서 최근 기계학습에 의한 방법에 이르기까지 다양한 피부병변 검출 알고리즘이 제안되어 왔다[1-8]. 이러한 방법들은 주로 단일 질환의 단일 병변을 포함하는 확대경 피부영상에 적용되어 왔다.

피부질환 환자 중에는 종종 다중 피부질환 또는 다중 피부병변을 가진 경우가 있는데, 저해상도의 임상 피부영상으로부터 다중 병변영역의 동시 검출은 피부영상진단 모델의 임상적 실용성을 향상시킬 수 있다. 그러나, 기존 피부병변 검출 방법은 여러 질환의 다중 병변이 혼재하는 저해상도의 임상 피부영상에서

성능이 검증된 바 없다. 반면, 체계적으로 수집된 다중 병변 임상 피부영상 데이터셋의 부족으로 심층학습 모델의 훈련과 검증에 어려움이 있다.

본 연구에서는, 다중 피부병변 검출 심층학습 모델을 학습 및 검증하기 위한 데이터를 증강하는 방법으로서, 생성적 적대 신경망(GAN)을 사용하여 다중 병변을 가진 피부 합성영상을 생성하는 기법을 제안한다.

2. 관련 연구

피부병변 검출을 위한 기존 방법은 영상처리에 의한 접근방법과 심층학습에 의한 접근방법으로 구분할 수 있다. 영상처리에 의한 접근방법은 임계값, 클러스터링, 퍼지, 양자화, 능동 윤곽선(active contour), 패턴 클러스터링 등의 방법이 있다[1-4]. 최근에는 피부병변 검출을 위한 심층학습 모델이 발전되어 왔는데, CDCNN

(Convolutional-Deconvolutional CNN), Densely Linked CNN (DermoNet), Mask R-CNN, YOLO v3 등을 사용한 모델이 제안되었다[5-8]. 그러나 기존 피부병변 검출 심층학습 모델은 고해상도의 확대경 영상 데이터에 주로 적용되어 온 반면, 본 논문에서 제안된 방법은 저해상도의 임상 피부영상에서 다중 병변을 검출하는 것을 목적으로 한다.

영상합성을 통한 데이터 증강 기법에 관하여, 흉부 X-ray 합성영상을 사용하여 훈련된 모델이 실제 X-ray 영상을 사용하여 훈련된 모델보다 성능이 20% 향상되었다는 연구결과가 최근 보고되었다[9]. 최근에는 GAN을 사용하여 피부질환 합성영상을 생성하는 기법이 제안되었다[10]. 기존 영상합성 기반 데이터 증강 기법이 단일 피부병변의 합성영상을 이미지 전체로 생성하는 방식인 반면, 제안된 방법은 다중 피부병변의 합성영상을 생성하는 방법으로서, 단일 병변을 가진 피부영상으로부터 병변영역을 분리한 후 GAN 모델을 사용하여 여러 병변영역을 타겟 피부영상에 혼합하는 방식이라는 점에서 독창성이 있다.

3. 제안 방법

본 논문에서 제안된 다중 피부병변 검출 모델의 훈련을 위한 영상합성 기반 데이터 증강 기법은 그림 1에 요약되어 있는 바와 같이 (1) 기존 피부영상 데이터셋으로부터 병변영역 분할을 통해 피부병변 딕셔너리(dictionary)를 생성하는 단계와 (2) 생성적 적대 신경망(GAN)을 사용하여 딕셔너리의 여러 피부병변을 타겟 임상피부영상에 혼합하는 단계로 구성된다.

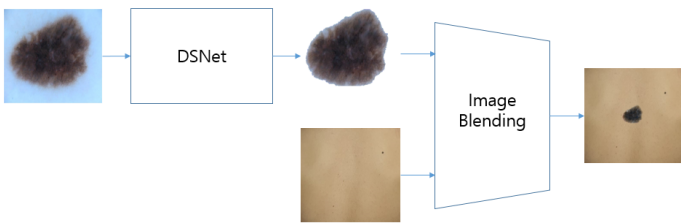


그림 1. 피부병변 합성 기반 데이터 증강 기법

피부병변 딕셔너리 생성 단계에서는, 피부병변 분할을 위한 심층학습 모델로서 DSNet을 사용하였다[11]. DSNet은 U-net과 비슷하게 인코더와 디코더의 구성요소로 이뤄져 있다. 인코더는 DenseNet121의 구조를 도입하였고 디코더는 인코더에 의해 출력된 낮은 해상도의 특징맵(feature map)을 높은 해상도의 픽셀 공간에 투영하여 세밀한 픽셀 단위 분류를 얻는다. 또한 풀링으로 인해 손실된 정보를 복구할 수 있도록 인코더와 디코더 사이에 skip connection이 존재한다. 학습 데이터로는 고해상도의 확대경 피부영상을 임상 피부영상 수준의 저해상도로 다운샘플링(downsamp-

ling)한 데이터가 사용되었다.

병변-피부 혼합 단계에서는, 타겟 피부영상으로부터 임의의 위치로부터 추출된 피부 패치(skin patch)에 딕셔너리의 피부병변을 혼합하는 과정으로 구성된다. 영상혼합 모델로서, Deep Image Blending(DIB) 모델과 Gaussian-Poisson GAN(GP-GAN) 모델의 성능을 비교하였다. 두 모델은 gradient 기반 방법과 GAN을 결합한 프레임워크이다. DIB 모델은 Poisson gradient loss와 사전학습된 VGG-16으로부터 계산된 style loss 및 content loss를 함께 최적화하는 방식인 반면[12], GP-GAN은 gradient 정보와 GAN으로부터 얻어진 색상 정보를 바탕으로 Gaussian-Poisson 방정식을 최적화하는 방식이다[13]. 위 과정은 그림 2의 의사코드에 요약되어 있다.

그림 2. 다중 피부병변 합성영상 생성 알고리즘

```

다중 피부병변 합성영상 생성 알고리즘
-----
Input a target skin image
N = the number of lesions which will be blended

for i = 1 to N
    Extract a skin patch from the target skin image
    Get a random lesion from dictionary
    Blending the lesion to the skin patch
    Attach the skin patch with lesion to the target
    
```

마지막으로, 생성된 다중병변 피부 합성영상을 사용하여 단일병변 피부영상으로 사전학습된 YOLO v4의 다중 피부병변 검출 성능을 시험하였다[14]. 본 연구팀은 다중 병변을 가진 실제 피부영상에 대하여 YOLO v4의 다중 피부병변 검출 실험을 보고한 바 있다[15]. 본 논문에서는 GAN을 사용하여 생성된 다중병변 피부 합성영상에 대하여 YOLO v4의 다중 피부병변 검출 성능을 검증한다는 점에서 차이가 있다.

3. 결 과

피부병변 딕셔너리 생성 단계에서, 피부병변 분할을 위한 DSNet의 학습 데이터는 ISIC 2016/2017 데이터셋의 확대경 영상으로부터 임상 피부영상 수준의 화질인 256x192 해상도로 down-sampling된 2,150 장의 데이터를 사용하였다. 학습, 검증, 테스트 데이터 로는 각 1,850, 150, 150장으로 분할되었고, 데이터셋에 포함된 피부병변에 대한 이진 정답 마스크가 사용되었다. 모델 가중치의 초기값은 ImageNet 모델을 사용하였고, 모델의 최적화 기법(optimizer)은 Adam이 사용되었고, 학습의 일괄 처리량(batch size)는 5로, 반복횟수(epoch)는 200으로 설정하였다.

그림 3은 DSNet을 사용한 피부병변 분할의 예시를 보여준다. 테스트 데이터에 대한 성능 지표는 정확도 (accuracy)는 94.9%, 민감도(sensitivity) 93.2%, 특이도 (specifi-city) 95.1%, 평균 IoU (intersection over union) 84.2%를 보였다. 그림 4는 피부병변 영역분할의 결과를 요약하는 혼동 행렬(confusion matrix)을 보여 준다.

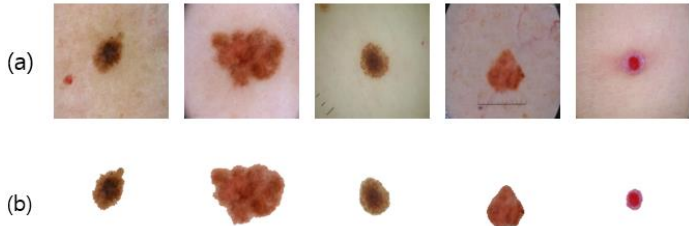


그림 3. DSNet을 사용한 피부병변 영역분할. (a) 원본 피부영상, (b) 분할된 피부병변

		Prediction	
		P	N
Ground truth	P	13198 TP	731 FN
	N	1771 FP	33451 TN
		P	N

그림 4. 피부병변 영역분할에 대한 혼동 행렬. (TP: True positive, FP: False positive, FN: False negative, TN: True negative)

병변-피부 혼합 단계에서는, DSNet에 의해 추출된 피부병변 이미지 6장을 테스트 샘플로 사용하였다. 그림 5는 DIB와 GP-GAN의 사전 학습된 모델을 사용하여 DSNet으로 추출된 피부병변과 사람 피부 이미지를 혼합한 결과를 보여준다.

합성된 이미지의 실감성에 대한 정성적 평가를 위해, 20대 한국인 14명을 대상으로 6장의 병변-피부 혼합영상 샘플에 대하여 1~5 사이의 실감성 등급을 매기게 하여 평균의견점수(mean opinion score, MOS)를 계산하였다 [16]. 정성적 평가 결과, 병변-피부 혼합 합성영상은 GP-GAN과 DIB를 사용하였을 때 각각 3.72 ± 1.21 와 3.27 ± 1.34 의 MOS 수치를 보였다. 대응표본 t-검정 (paired sample t-test) 결과 p-value는 0.015로 0.05의 유의수준보다 작으므로 통계적으로 GP-GAN의 MOS가 DIB와 비교하여 유의미한 차이를 보였다.

YOLO v4를 사용한 다중 피부병변 검출 실험에서는, 임상 피부영상으로부터 3개의 사각형 패치를 임의의 위치에 설정한 후 패치 이미지에 피부병변을 혼합한 후 삽입하는 방식으로 다중 피부병변을 가진 합성영상

3장을 생성하여 테스트 샘플로 사용하였다. 단일병변을 가진 혼합영상 10장에 대해서도 병변 검출의 성능을 측정하였다. YOLO v4의 훈련을 위해 ISIC 2016/2017 /2019 데이터셋으로부터 20,000장의 피부영상 데이터를 사용하였다. 정답 bounding box는 DSNet을 이용해 병변영역을 분할하여 취득하였다. 모델 훈련을 위한 반복횟수는 100으로 설정되었고, 병변 영역의 유무만 확인하기 위해 1개의 클래스만 정의하였다. 그림 6은 DIB 기반으로 합성된 다중 피부병변 합성영상에서 학습된 YOLO v4를 사용하여 다중 피부병변을 검출한 결과를 보여준다.

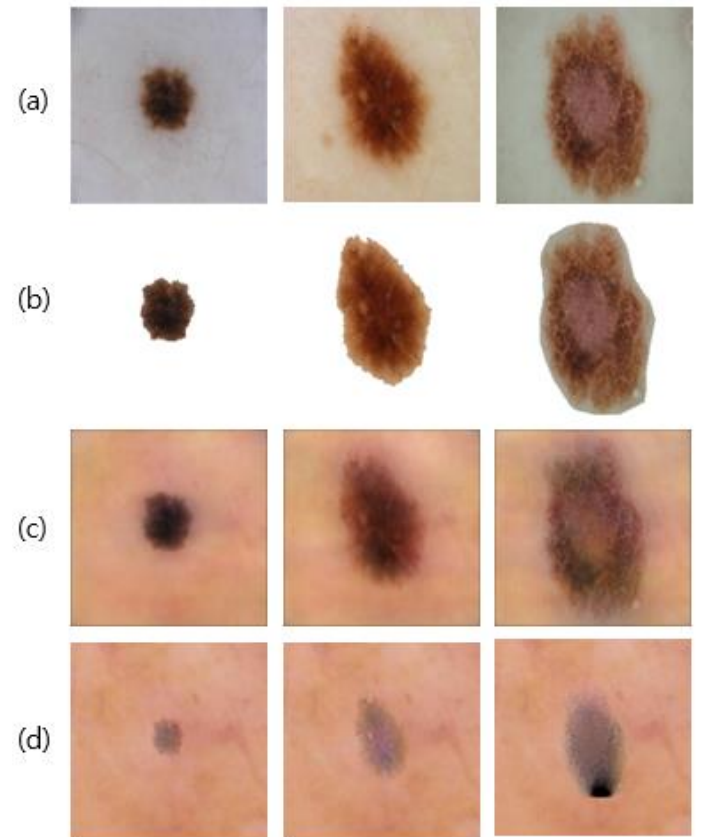


그림 5. 병변-피부 혼합 영상의 예시. (a) 원본 소스 피부영상, (b) 분할된 피부병변, (c) GP-GAN으로 생성된 피부-병변 혼합영상, (d) DIB로 생성된 피부-병변 혼합영상.

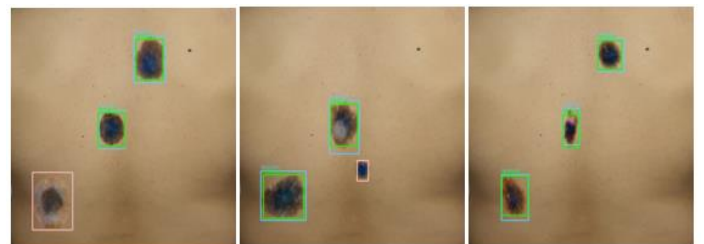


그림 6. 다중 피부병변 검출 결과. (파랑/오렌지) 정답 바운딩 박스, (녹색) 예측된 바운딩 박스

병변 검출에 대한 정량적 평가 결과는 표 1에 요약되어 있다. 단일 병변에 대한 검출에 대해서는, DIB와 GP-GAN 모두 100%에 가까운 mAP를 보였다. 그러나, 다중 병변에 대한 검출에서는 DIB로 합성된 피부병변 영상에 대해서 mAP가 77.8%로 감소하였고, GP-GAN으로 합성된 영상에 대해서는 피부병변 검출이 실패하였다.

표 1. YOLO v4에 의한 병변영역 검출 결과

종류	모델	IoU	mAP
단일병변	DIB	86.3%	100%
	GP-GAN	88.2%	100%
다중병변	DIB	91.6%	77.8
	GP-GAN	-	-

제안된 방법은 몇 가지 위협요소를 안고 있다. 첫째, 합성영상에 대한 정성적 평가를 위한 설문조사는 참여자의 주관적 편향성에 좌우될 수 있다. 이러한 문제를 경감시키기 위해 방법과 내용이 참여자에게 블라인드(blind) 되도록 하였다. 둘째, 병변분할 및 병변-피부 혼합의 성능은 피부질환의 종류에 따라 좌우될 수 있다. 이러한 위험을 줄이기 위해, 2가지 피부질환(seborrheic keratosis, melanoma)으로 구성된 피부영상 데이터를 고르게 사용하여 병변분할 및 병변-피부 혼합영상을 구성하였다.

5. 결 론

본 연구에서는 다중 병변 검출 심층학습 모델의 훈련을 위한 학습 데이터를 증강하는 기법으로서 피부 병변이 있는 혼합 영상을 생성하는 GAN 모델을 비교 분석하였다.

본 연구는 몇 가지 기술적 한계를 갖고 있다. 병변-피부 혼합영상에 대한 실감성 평가는 GP-GAN과 DIB를 사용하였을 때 각각 74%와 65%에 그쳐 성능이 개선되어야 한다. 다중 병변의 검출에서 GP-GAN이 성공적이지 못했던 것은 영상혼합 과정에서 피부의 색상과 질감이 상당히 변형된 점도 영향을 준 것으로 분석된다.

또한, 병변-피부 혼합영상에 대한 정성적 평가와 아울러 Inception Score (IS), Frechet Inception Distance (FID) 등을 사용한 병변-피부 혼합영상에 대한 정량적 평가가 보완되어야 한다. 한편, 제안된 방법으로 생성된 다중 피부병변 합성영상을 사용하여 학습 데이터를 증강하여 학습된 모델에 대한 훈련과 평가가 후속 연구과정을 통해 이루어져야 할 것이다.

이러한 한계에도 불구하고, 제안된 기법은 GAN을 사용한 다중 피부병변 합성 기법이라는 점에 독창성이

있다. 다중 피부병변 검출 심층학습 모델을 위한 학습 데이터가 부족한 현실에서, 합성 피부영상 기반의 데이터 증강 기법은 피부진단 기술의 발전을 위한 획기적인 돌파구가 될 것으로 기대된다.

Acknowledgments

본 논문은 2021년도 중소벤처기업부의 지원을 받아 수행된 지역특화산업육성과제의 연구결과로 수행되었음 (피부임상영상기반의 병변부위 자동검출 및 초해상화 시스템개발).

참고문헌

- [1] P. Pereira, R. Fonseca-Pinto, R. Paiva, P. Assuncao, L. Tavora, L. Thomaz, S. Faria, "Dermoscopic skin lesion image segmentation based on Local Binary Pattern Clustering: Comparative study," Biomedical Signal Processing and Control, Vol. 59, 101924, 2020.
- [2] L. Huang, M. Wang, "Image thresholding by minimizing the measures of fuzziness, Pattern Recognition," Pattern Recognition, Vol. 28 Issue 1, pp. 41-51, 1995.
- [3] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pp. 603-619, 2002.
- [4] S. Lankton and A. Tannenbaum, "Localizing Region-Based Active Contours," in IEEE Transactions on Image Processing, vol. 17, no. 11, pp. 2029-2039, 2008.
- [5] Y. Yuan, M. Chao and Y. Lo, "Automatic Skin Lesion Segmentation Using Deep Fully Convolutional Networks With Jaccard Distance," in IEEE Transactions on Medical Imaging, vol. 36, no. 9, pp. 1876-1886, 2017.
- [6] S. Baghersalimi, B. Bozorgtabar, P. Schmid-Saugeon et al., "DermoNet: densely linked convolutional neural network for efficient skin lesion segmentation," EURASIP Journal on Image and Video Processing, no. 71, 2019.
- [7] Cheng Huang and Anyuan Yu, "Skin lesion segmentation based on R-CNN," 2020 International Conference on Virtual Reality and Visualization (ICVRV), Nov. 2020.
- [8] Halil Murat Ünver and Enes Ayan, "Skin Lesion Segmentation in Dermoscopic Images with Combination of YOLO and GrabCut Algorithm," Diagnostics, 9:72, 2019.

- [9] H. Salehinejad et al., "Generalization of Deep Neural Networks for Chest Pathology Classification in X-Rays Using Generative Adversarial Networks", in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP), pp. 990–994, 2018.
- [10] 정현석, 하성민, 심학준, 오상호, 장혁재, "데이터 불균형 해결을 위한 조건부 심층 적대적 생성 신경망 기반 피부질환 영상생성 기법," 대한전자공학회 하계학술대회, 2020.
- [11] Md. Kamrul Hasan, Lavsén Dahal, Prasad N. Samarakoon, Fakrul Islam Tushar, Robert Marti Marly, "DSNet: Automatic Dermoscopic Skin Lesion Segmentation", arXiv:1907.04305, 2020.
- [12] Lingzhi Zhang, Tarmily Wen, Jianbo Shi, "Deep Image Blending", WACV, arXiv:1910.11495v1, 2019.
- [13] Huikai Wu, Shuai Zheng, Junge Zhang, Kaiqi Huang, "GP-GAN: Towards Realistic High-Resolution Image Blending", arXiv:1703.07195v3, 2019
- [14] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv:2004.10934, 2020.
- [15] 박범진, 이영찬, 유원상, "임상 피부영상에서 YOLO 기반 다중 병변영역 검출에 관한 예비 연구", KICSP, 2021
- [16] Aaron van den Oord et al., "Parallel WaveNet: Fast High-Fidelity Speech Synthesis," Proceedings of the 35th International Conference on Machine Learning, PMLR 80:3918–3926, 2018.

텍스트 기반의 CAPTCHA 보완 모델

김태호^o, 고유라, 한종대

상명대학교

kok8454@gmail.com, koyura18398229@gmail.com, elevenwhite@smu.ac.kr

Text-based CAPTCHA supplementary model

Taeho Kim^o, Yura Ko, Jongdae Han

Sangmyung University

요 약

본 연구에서는 텍스트 기반의 CAPTCHA 보완 모델을 생성하고자 한다. 최근 인공지능의 발전으로 CAPTCHA를 우회하는 경우가 많아지면서 다양한 형태의 노이즈가 기존 텍스트 기반의 CAPTCHA에 추가되고 있다. 하지만 이는 실제 사용자의 가독성을 떨어뜨리며 점차 자동화 공격 방지 솔루션으로서 기능을 상실하게 된다. 이를 해결하기 위해 노이즈 추가를 대신해 텍스트 구성 성분 자체에 처리를 가하고 CAPTCHA 내 텍스트 글자 수, 출력 서체의 종류, 텍스트 구성 성분의 조합을 기준으로 CAPTCHA challenge loss 값을 파악했다. 그 결과, 3가지 기준의 변인을 모두 임의로 설정하고 생성한 CAPTCHA를 학습한 모델에서 CAPTCHA challenge 성공률을 83%로 확인했다. 이에 맞춰 CAPTCHA 해독 성공률을 낮출 수 있는 텍스트 기반의 CAPTCHA 보완 모델을 제안한다.

1. 서 론

1.1 연구 목적 및 의의

1.1.1 연구 목적

CAPTCHA(Completely Automated Public Turing test to tell Computers and Human Apart, 완전 자동화된 사람과 컴퓨터 판별, 캡차)는 사람과 컴퓨터 프로그램을 구별하기 위해 사용되는 기술이다. CAPTCHA는 사람은 구별할 수 있지만 컴퓨터 프로그램은 구별하기 어렵도록 의도적으로 왜곡된 텍스트 또는 이미지를 보여주고 이로부터 왜곡되기 전의 기존 텍스트 또는 이미지를 유추할 수 있는지를 확인하는 방식으로 사람과 컴퓨터 프로그램을 구별한다. 사람과 달리 컴퓨터 프로그램은 변형된 텍스트나 이미지를 인식하지 못하므로 CAPTCHA 인식 단계를 통과하지 못하고 이를 통해 흔히 웹사이트 회원가입 시 자동 가입 방지 프로그램으로 활용된다. 하지만 텍스트 기반의 CAPTCHA의 경우, 이미지로 제공되는 텍스트를 자동으로 인식하는 기술이 발전하면서 이를 활용해 인공지능 기반의 ‘봇(Bot)’으로 컴퓨터 프로그램이 CAPTCHA를 우회해 계정을 생성하는 CAPTCHA challenge 성공률이 점차 높아지고 있다. 텍스트 기반의 CAPTCHA가 자동화 공격 방지 보안 솔루션이라는 본래의 목적을 달성해지기 어려워지면서 이를 극복하기 위해 텍스트의 회전 또는 점, 선 등의 노이즈를 과도하게 활용하면서 점차 인식하기 어려운 텍스트 기반의 CAPTCHA를 생성하지만 이는 사람 또한 육안으로 식별하기 힘든 정도에 달했다.¹ 텍스트 기반의 CAPTCHA의 낮은 사용성을 극복하고 CAPTCHA challenge 성공률을 낮춰 보안성을 강화할 수 있는 텍스트 이미지의 특성을 찾아내

고 이를 기반으로 텍스트 기반의 CAPTCHA 보완 모델을 생성한다면 본래의 보안 솔루션으로서 기능을 되찾을 수 있다.

1.1.2 연구 의의

최근 많은 웹 사이트에서 자동화 공격을 예방 및 방어하기 위해 사용되는 텍스트 기반의 CAPTCHA의 보안성과 사용성의 현황에 대해 이해한다. 기존의 텍스트 기반의 CAPTCHA가 갖는 문제점을 파악하고 파악한 취약점을 보완할 수 있는 데이터 생성을 통해 자동화 공격 방어 보안 솔루션의 특성과 취약점 보완 방법에 대해 연구하며 컴퓨터 과학 기술 발전에 따른 보안 대책에 관해 심도 깊은 이해를 돕는다. 텍스트 기반의 CAPTCHA의 보안성과 사용성을 동시에 고려하여 봇(Bot)이 학습하기 힘든 텍스트 데이터의 특성을 연구하여 자동화 공격 방지 보안 솔루션으로서 텍스트 기반의 CAPTCHA가 갖는 안정성과 보안성, 사용성을 보완할 수 있다. 또한, 향상된 텍스트 기반의 CAPTCHA를 상용화하여 웹 사이트 상의 무분별한 웹 크롤링(Web Crawling)을 차단하는 근본적인 아이디어를 제공할 수 있다.

1.2 연구 방법

1.2.1 연구 범위 및 과정

다양한 유형의 CAPTCHA 중, 텍스트 기반의 CAPTCHA에 한해 보완 모델을 생성한다. 텍스트 기반의 CAPTCHA 이미지 내 구성 성분은 숫자, 영문 대문자, 영문 소문자로 제한한다. 연구 과정은 3단계로 구성했으며 자세한 과정과 내용은 다음과 같다.

1. 텍스트 기반의 CAPTCHA 생성 및 이미지 전 처리
 숫자는 0부터 9까지로 10개, 영문 대문자 26개, 영문 소문자 26개를 활용해 텍스트 기반의 CAPTCHA를 구성한다. 3가지의 구성 성분은 총 62개로 정형화된 형태를 피하고 봇(Bot)의 CAPTCHA 인식률을 낮출 수 있는 형태로 전처리를 거친다. 전 처리 후의 이미지는 사람이 육안으로 파악할 수 있는 정도로 유지하면서 각 성분별로 정의한 기준을 기반으로 잘라낸다. 기준은 다음과 같다.



그림1. 숫자의 기본 이미지와 전 처리 후 이미지

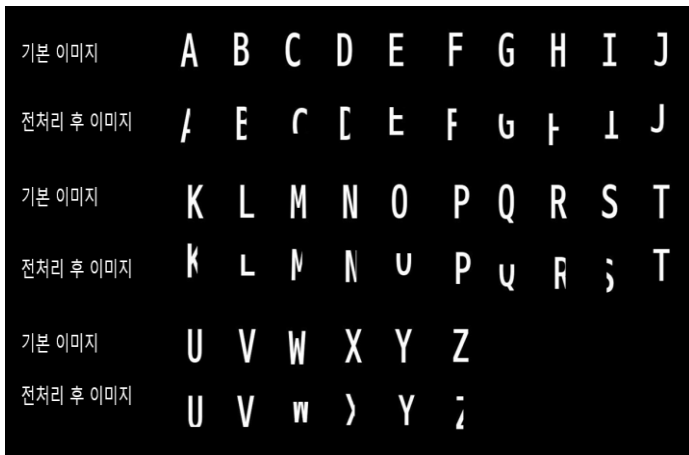


그림2. 대문자의 기본 이미지와 전 처리 후 이미지

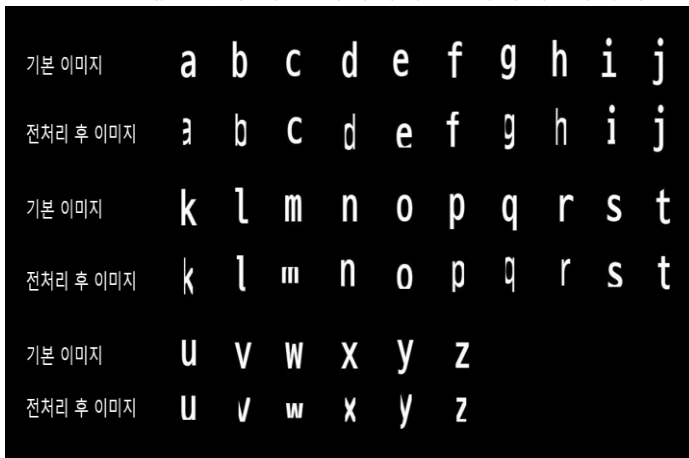


그림3. 소문자의 기본 이미지와 전 처리 후 이미지

잘라낸 이미지를 임의로 이어 붙여 텍스트 기반의 CAPTCHA를 생성한다. 그 후, 노이즈를 추가하는 등 기존의 CAPTCHA를 생성하는 것과 같은 방식으로 왜곡을 더해 텍스트 기반의 CAPTCHA를 생성한다.

2. CAPTCHA 보안 모델 학습

결과 분석 기준을 설정하고 각 기준에 따라 일만 개의 CAPTCHA를 생성한다. 각 기준 별로 CAPTCHA의 텍스트를 인식하는 CRNN 모델을 만든다. 또한 CTC 손실 함수를 사용해 훈련한다.

3. CAPTCHA challenge

각 기준에 따른 텍스트 기반의 CAPTCHA 학습을 거친 CAPTCHA 보안 모델에 대해 CAPTCHA challenge를 진행하고 CAPTCHA challenge loss 값을 그래프로 나타낸다. 각 기준에 따른 CAPTCHA challenge의 loss 그래프를 '어떠한 특징을 가진 텍스트 기반의 데이터가 텍스트 기반의 CAPTCHA의 보안성을 높이는 데 효과적인 것인가'를 주제로 비교하며 CAPTCHA challenge에 실패한 CAPTCHA가 이미지 및 구성 성분 측면에서 갖는 특징을 선별해낸다.

1.2.2 용어 정의

본문에서 CAPTCHA challenge란 CAPTCHA에 대한 우회, 즉 봇(Bot)의 우회 시도를 지칭한다. CAPTCHA challenge에 성공했다는 것은 봇(Bot)이 텍스트 기반의 CAPTCHA 내 텍스트를 정확히 인식했다는 뜻이다.

2. 본 론

2.1 CAPTCHA의 정의와 사전 지식

2.1.1 CAPTCHA의 정의

CAPTCHA(Completely Automated Public Turing test to tell Computers and Humans Apart)는 사용자가 실제 사람인지 컴퓨터 프로그램인지를 구별하기 위해 사용되는 방법으로 크게 3가지로 텍스트 기반의 CAPTCHA, 음성 기반의 CAPTCHA, 이미지 기반의 CAPTCHA가 있다. CAPTCHA가 주로 활용되는 분야는 광고성 게시물 방지, 아이디 자동생성 방지, 무차별 대입 방법에 의해 비밀번호를 찾는 공격 방지 등 악의적인 목적으로 부당한 이득을 취할 가능성이 있는 곳이다.

2.1.2 사전 지식

본문에서 다루는 텍스트 기반의 CAPTCHA는 컴퓨터는 읽기 어려운 반면 사람은 읽기 쉬운 글자들의 조합을 만들기 위해 OCR(Optical Character Recognition)의 한계를 이용하기 위해 각각의 글자에 보편적으로 사용하는 저항 메커니즘을 적용한다. 저항 메커니즘의 유형은 다음과 같다.



그림4. 글자 겹침



그림5. 글자 왜곡



그림6. 글자 회전



그림7. 호 삽입



그림8. 점 삽입



그림9. 가변 크기

1. 글자 겹침(Overlapping): 글자끼리 겹치게 한다.
2. 글자 왜곡(Distortion): 글자를 뒤틀리게 한다.
3. 글자 회전(Rotation): 글자를 회전한다.
4. 호(Arc) 삽입: 글자를 가로지르는 호를 삽입한다.
5. 점(Dot) 삽입: 무작위로 점을 삽입한다.
6. 가변 크기: 글자의 크기를 다양화한다.

2.2 CAPTCHA challenge 현황

CAPTCHA는 OCR의 한계를 이용하여 만든 것이기 때문에 한계를 극복할 수 있는 방법을 고안하면 인공 지능으로 하여금 자동으로 CAPTCHA challenge를 성공하게 할 수 있다. 사람보다 인공 지능이 하나의 글자를 더 정확하게 인식하기 때문에² 텍스트 기반의 CAPTCHA에 사용되는 글자들은 하나의 글자로 보이지 않도록 저항 메커니즘이 적용되어 있다. 이에 따라 글자들에 대해 인공 지능을 활용하여 각 글자를 분리하는 기술이 만들어졌으며 시간이 지남에 따라 성공률도 높아지는 추세이다. 글자 분리 기술의 성공률이 높아짐에 따라 글자를 인식할 확률도 덩달아 높아지므로 텍스트 기반의 CAPTCHA challenge 성공률도 상승하게 되고 악성 봇(Bot)의 활동도 증가하게 된다. 2020년을 기준으로 전체 인터넷 트래픽 중 악성 봇(Bot)이 차지하는 비율이 약 25%이다.³ 악성 봇(Bot)은 인터넷을 떠돌며 개인 정보를 무단으로 수집 및 악용하기 때문에 무단으로 수집된 정보로 인해 실제 인터넷 사용자가 피해를 볼 가능성이 증가하고 특정 사이트에 대한 트래픽을 높이고 이 트래픽을 유지하여 사이트를 마비시키는 등 웹 생태계에 악영향을 많이 끼친다. 또한 악성 봇(Bot)을 만들 때에 악성 봇(Bot)이 자동으로 CAPTCHA challenge를 통과하는 기술을 보유하지 않더라도 매달 일정 금액만 지불하면 봇(Bot)이 자동으로 CAPTCHA challenge를 시도하는 프로그램을 이용할 수 있다.

따라서 현재 텍스트 기반의 CAPTCHA는 악성 봇(Bot)을 차단하는 것에 대해 실패가 꽤나 빈번하기 때문에 본래의 목적을 달성하기 위해 분리 기술이 어렵도록 텍스트 기반의 CAPTCHA 보완 모델을 만들 필요가 있다. 하지만 분리 기술이 어렵다는 것은 저항 메커니즘들을 많이 적용한 것으로 실제 사람도 글자를 알아보기까지 오랜 시간이 걸리거나 알아보지 못하는 경우가 발생하고 있다.

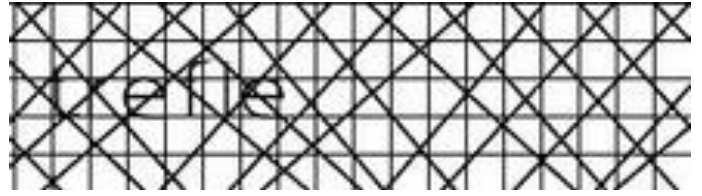


그림10. 악성 봇을 차단하기 위해 많은 저항 메커니즘을 적용해 사람의 가독성이 떨어진 경우

따라서 보안성이 증대된 텍스트 기반의 CAPTCHA의 필요성이 대두되었기 때문에 본문에서는 악성 봇(Bot)의 CAPTCHA challenge 성공률을 낮추고 사람의 인식률을 높여 CAPTCHA의 보안성과 사용성을 높이는 방법을 연구한다.

2.3 연구 과정

2.3.1 연구 기준

CAPTCHA challenge 성공률이 높지 않은 텍스트 기반의 CAPTCHA의 특징을 구별하는 것을 목표로 연구를 진행한다. 따라서 텍스트 기반의 CAPTCHA의 특징을 기준으로 연구 결과 분석을 진행한다. 텍스트 기반의 CAPTCHA의 특징은 크게 1. CAPTCHA 내 텍스트의 글자 수, 2. CAPTCHA 내 텍스트 서체의 종류, 3. CAPTCHA 내 텍스트 구성 성분의 조합으로 총 3가지로 나눈다.

1번 기준 (CAPTCHA 내 텍스트 글자 수)의 세부 항목은 3가지로 4글자, 6글자, 10글자로 구성한다. 2번 기준 (CAPTCHA 내 텍스트 서체의 종류)의 세부 항목은 3가지로 'D2CodingBold', 'Calista', '다행체'로 구성한다.



그림11. 'D2CodingBold'체



그림12. 'Calista'체



그림13. '다행'체

3번 기준 (CAPTCHA 내 텍스트 구성 성분의 조합)의

세부 항목은 4가지로 ‘숫자’, ‘숫자 + 영문 소문자’, ‘숫자 + 영문 대문자’, ‘숫자 + 영문 소문자 + 영문 대문자’로 구성된 이미지로 구성한다.

따라서, 본문에서 전체 3가지 기준 내 총 10가지 세부 항목으로 연구 결과를 분석한다.

2.3.2 학습 모델

다음의 단계를 사용해 구성된 텍스트 기반의 CAPTCHA 텍스트 인식 모델을 활용해 CAPTCHA 학습을 거친다.

1. CAPTCHA 데이터 셋 구성

앞에서 정의한 총 3가지 기준 별로 일만 개의 CAPTCHA 데이터 셋을 구성한다.

2. CAPTCHA 전 처리

구성하는 모델에 적합하도록 전 처리를 진행한다. 입력 이미지와 출력 레이블을 모두 전 처리한다. 입력 이미지의 전 처리를 위해 다음의 과정을 진행한다. 이미지를 읽고 회색조 이미지로 변환한다. 패딩을 사용해 각 이미지를 (128,32) 크기로 만든다. 모델의 입력 형태와 호환될 수 있도록 이미지 차원을 (128, 32, 1)로 확장하고 이미지 픽셀 값을 255로 나누어 정규화한다.

출력 레이블의 전 처리를 위해 다음의 과정을 진행한다. 이미지 이름에는 이미지 내부의 텍스트가 포함되어 있기 때문에 이미지 이름에서 텍스트를 읽는다. 함수를 생성하여 단어의 각 문자를 숫자 값으로 인코딩한다. 단어에서 최대 길이를 계산하고 모든 출력 레이블을 채워 최대 길이와 동일한 크기로 만든다.

3. 모델 구성

학습 모델은 CNN과 RNN, CTC 손실 함수로 구성된다. CNN(Convolutional Neural Network)은 시각적 이미지 분석을 담당하는 인공 신경망을 제공하는 딥 러닝 클래스이다. RNN(Recurrent Neural Network)은 이전 단계의 출력이 현재 단계의 입력으로 제공되는 신경망이다. 시퀀스에 대한 일부 정보를 기억한다. CTC 손실 함수는 각 단계의 출력을 예측하기 위해 사용되는 전사 레이어이다.

4. 모델 훈련

CAPTCHA 이미지를 훈련한다.

5. CAPTCHA challenge

기준에 따라 인식률을 확인하며 loss 값을 기반으로 기준 별 그래프를 작성하고 CAPTCHA challenge 성공률을 낮출 수 있는 CAPTCHA 내 텍스트가 갖는 특징을 선별한다.

2.4 연구 결과

2.4.1 CAPTCHA challenge loss 그래프

기준 1. CAPTCHA 내 텍스트의 글자 수를 4자, 6자, 10자로 나누어 글자 수에 따른 loss 값을 나타낸 그래프는

다음과 같다.

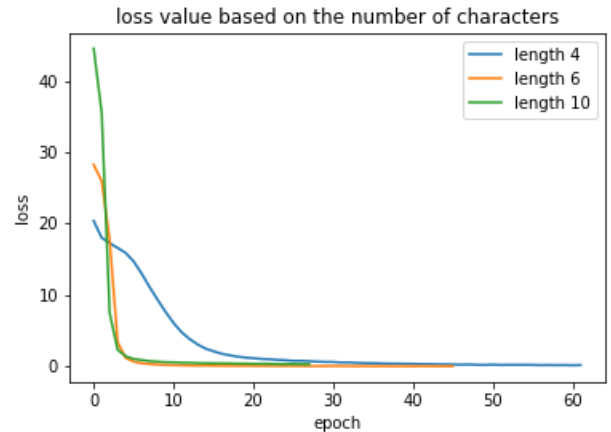


그림 14. CAPTCHA 내 텍스트의 글자 수(4자, 6자, 10자)에 따른 loss 값을 나타낸 그래프

CAPTCHA 내 텍스트의 글자 수가 많을 수록 초기의 loss 값이 커지는 것을 확인할 수 있었다.

하지만 학습 데이터 량이 각 1만 개로 충분히 많기 때문에 초기 loss값의 관계없이 모두 0에 근사한 쪽으로 수렴하는 것을 확인할 수 있었다. 이는 평균적으로 1에 가까운 확률로 CAPTCHA 내 텍스트를 인식한다고 볼 수 있다. 따라서 데이터가 충분하다면 글자 수에 따라 CAPTCHA 내 텍스트의 해독을 막기 어렵다는 결론을 얻는다.

기준 2. CAPTCHA 내 텍스트 서체를 ‘D2CodingBold’, ‘Calista’, ‘다행체’로 나누어 서체에 따른 loss 값을 나타낸 그래프는 다음과 같다.

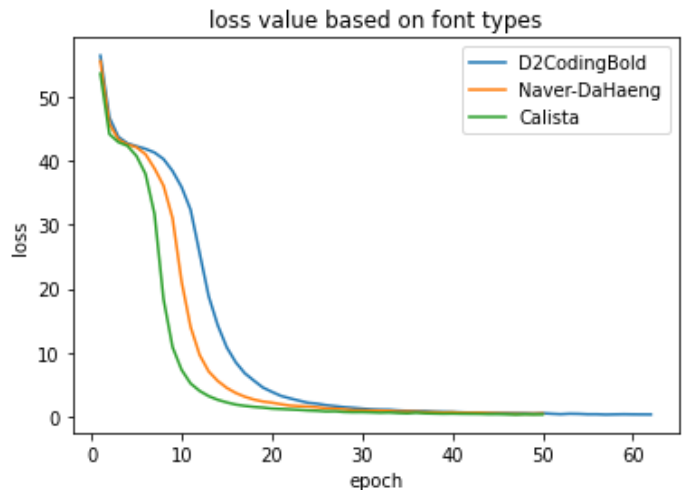


그림 15. CAPTCHA 내 텍스트의 서체(D2CodingBold, Calista, 다행체)에 따른 loss 값을 나타낸 그래프

CAPTCHA 내 텍스트의 서체가 육안으로 구별하기 어려운 필기체에 가까울수록 초기의 loss 값이 커지는 것을 확인할 수 있었다.

하지만 학습 데이터 량이 각 1만 개로 충분히 많기 때문에 초기 loss값의 관계없이 모두 0에 근사한 쪽으로 수렴하는 것을 확인할 수 있었다. 이는 평균적으로 1에 가까운

확률로 CAPTCHA 내 텍스트를 인식한다고 볼 수 있다. 따라서 데이터가 충분히 많다면 서체의 종류에 따라 CAPTCHA 내 텍스트의 해독을 막기 어렵다는 결론을 얻는다.

기준 3. CAPTCHA 내 텍스트 구성 성분의 조합을 ‘숫자’, ‘숫자 + 영문 소문자’, ‘숫자 + 영문 대문자’, ‘숫자 + 영문 소문자 + 영문 대문자’로 나누어 구성 성분의 조합에 따른 loss 값을 나타낸 그래프는 다음과 같다.

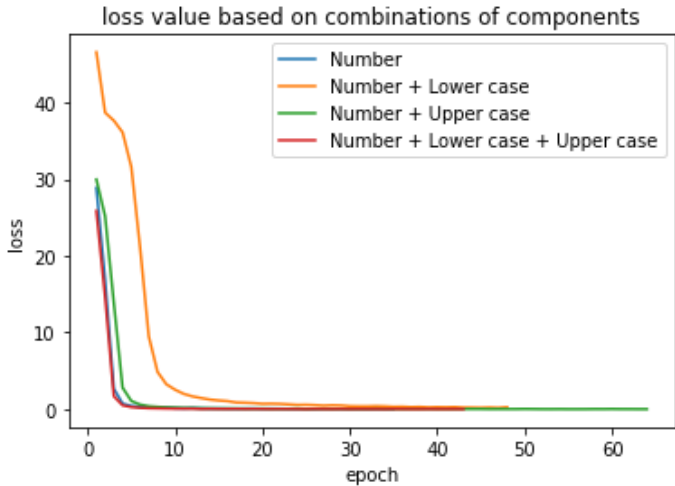


그림 16. CAPTCHA 내 텍스트의 성분 조합 유형(숫자, 숫자 + 영문 소문자, 숫자 + 영문 대문자, 숫자 + 영문 소문자 + 영문 대문자)에 따른 loss 값을 나타낸 그래프

CAPTCHA 내 텍스트 성분 조합 중 ‘숫자 + 영문 소문자’ 조합의 경우가 초기의 loss 값이 가장 컸다. 반면에 가장 다양한 성분을 갖는 ‘숫자 + 영문 소문자 + 영문 대문자’의 조합의 경우에서 초기 loss 값이 가장 작음을 확인할 수 있었다. ‘숫자 + 영문 소문자’의 경우가 초기 loss가 가장 큰데, 여기에 영문 대문자를 더한 것은 오히려 loss 값이 가장 낮아졌다. 하지만 학습 데이터 양이 각 1만 개로 충분히 많기 때문에 초기 loss값의 관계없이 모두 0에 근사한 쪽으로 수렴하는 것을 확인할 수 있었다. 이는 평균적으로 1에 가까운 확률로 CAPTCHA 내 텍스트를 인식한다고 볼 수 있다. 따라서 데이터가 충분히 많다면 텍스트를 구성하는 성분의 조합에 따라 CAPTCHA 내 텍스트의 해독을 막기 어렵다는 결론을 얻는다.

위의 각 기준들에 따라 데이터에 대한 예측을 하였을 때 성공률이 약 98%로 텍스트를 구성하는 성분의 조합에 따라 CAPTCHA 내 텍스트의 해독을 막기 어렵다는 결론을 얻을 수 있다. 이와 같은 결론이 나타난 이유는 글자에 조작을 가했을 때 서로 비슷한 글자들의 집합이 많지 않기 때문에 결과적으로는 62개의 글자를 분류한 작업과 비슷한 상황이기 때문이다. 즉 글자 분류 작업을 방해하는 기준이 필요하다는 결론을 얻는다.

따라서 새로운 기준을 기준 4로 설정하고 기준 4에 따라 텍스트 기반의 CAPTCHA를 생성해 학습을 시도했다. 기준 4는 다음과 같다. 기준 1, 기준 2, 기준 3에서 각각 변인으로 설정했던 CAPTCHA 내 텍스트의 글자 수, 글씨체, 구성 성분의 조합을 모두 임의로 설정했다. 또한, CAPTCHA

내 텍스트를 자르는 길이도 임의로 설정했다. 최대를 자를 수 있는 길이를 설정하고 무작위로 글자를 자른다. 이때 최대를 잘릴 수 있는 길이는 기존에 사용했던 전처리 할 때 사용한 길이와 같다. 기준 4에 따라 학습 데이터를 2천개를 생성하고 학습시켰다. 이 기준에 따른 loss 값을 나타낸 그래프는 다음과 같다.

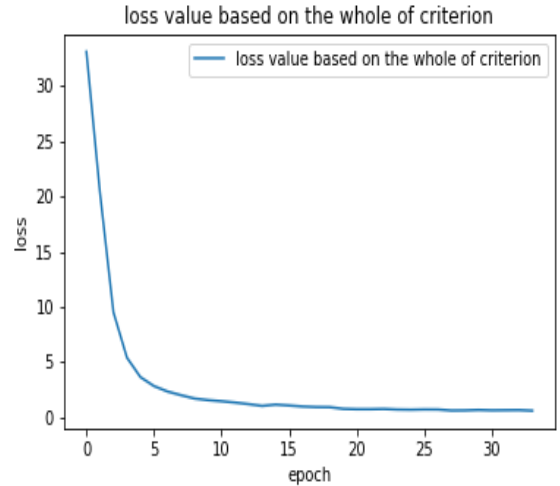


그림 17. 기준 4(기준 1, 2, 3의 모든 변인을 임의로 설정하고 CAPTCHA 내 텍스트를 자르는 길이를 다양화한 경우)에 따른 loss 값을 나타낸 그래프

마찬가지로 loss 값이 0에 근사하는 것을 볼 수 있다. 하지만 이 모델로 텍스트 기반의 CAPTCHA에 대한 예측을 하였을 때 CAPTCHA challenge 성공률이 약 83%로 측정되었다. 이는 기존에 알려진 CAPTCHA challenge 성공률 약 97%에 비해 낮은 수치이다. 다음은 예측에 실패한 CAPTCHA 예시이다.



그림 18. CAPTCHA 내 실제 데이터는 ‘d75y9’이나 예측한 데이터는 ‘o75y9’로 영문 소문자 ‘d’를 ‘o’로 잘못 분류한 경우

따라서 기준 4에 따라 CAPTCHA를 생성할 경우, CAPTCHA 내 텍스트의 해독을 방해할 수 있다는 결론을 얻는다.

3. 결 론

3.1 결론

CAPTCHA의 보안성과 사용성을 높이는 CAPTCHA 보안 모델의 특징을 살펴봤다. 기준 4에 따르면 기존 텍스트 기반의 CAPTCHA의 보안성을 높이기 위해 글자를 변형하는 방법을 제한할 수 있다. 이를 응용할 경우, 사람만 알아볼 수 있게 텍스트를 변형하는 방법은 무수히 많기 때문에 보안을 높이기 위한 한 가지 방법으로 쓰일 수 있는 가능성이 있다. 하지만 공격자가 CAPTCHA를 다운로드하여 변형 방

법과 변형 정도를 분석하고 이를 이용하여 CAPTCHA 해독 모델을 만든다면 보안성은 다시 취약해지기 때문에 이를 향후 연구에서 보완하고자 한다.

3.2 향후 연구 방향

공격자가 CAPTCHA를 즉시 해독하지 않고 다운로드하여 공격에 이용하는 것을 방지하기 위해 CAPTCHA를 자동화 코드를 통해 얻는 방법을 막는 환경을 구축하는 방법에 대해 연구할 계획이다. 1) N분할 shuffling, 2) 스크린 캡처를 통한 데이터 획득 방지와 같은 방법을 생각해 볼 수 있다.

3.2.1 N분할 shuffling

CAPTCHA가 저장되어 있는 서버에서 CAPTCHA를 요청한 웹 사이트에 이미지를 전송할 때에 균등하게 N조각을 낸 뒤 조각들을 서로 섞는다.

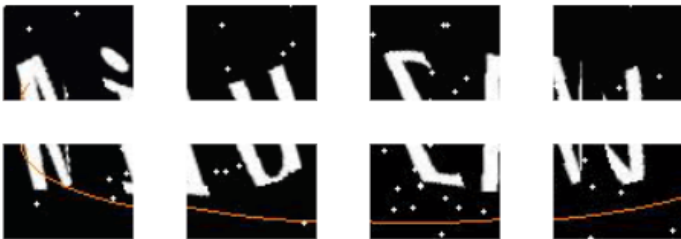


그림 19. N=8일 때 N분할 방식 사용 예시

순서를 섞은 후 올바른 순서에 관한 정보를 인코딩하고 웹 페이지는 순서가 섞여진 이미지를 태그에 올리도록 한다. 사용자에게 보여줄 이미지는 웹 페이지가 순서에 관한 정보를 디코딩하여 페이지에 로드한 후 파기한다. 이렇게 함으로써 공격자가 순서에 관한 정보를 디코딩 할 방법을 찾지 못하면 순서가 섞인 이미지를 다운로드 받을 수밖에 없다. 공격자가 순서가 섞인 이미지를 합쳐서 원본 이미지를 만들려면 2^N 가지의 경우를 모두 해볼 수밖에 없다. 결과적으로 N이 충분히 크다면 자동화 공격을 통하여 얻는 이득이 사람이 직접 수행하는 공격보다 작을 것으로 예상되기 때문에 CAPTCHA 데이터 획득을 방어할 수 있다고 생각된다.

3.2.2 스크린 캡처를 통한 데이터 획득 방지

스크린 캡처를 통한 데이터 획득은 공격자가 ROI 알고리즘을 이용하여 CAPTCHA영역을 확인한 후 해당 영역을 캡처 하는 것인데 본 연구에서 사용하지 않았던 adversarial attack 등 여러가지 방법을 적용하여 영역 식별을 방해하여 데이터 획득을 방지하려고 한다.

4. 참고문헌

[1] 조금환, 최주섭, 김형식, 보안성 및 사용성 측면에서의 CAPTCHA 동향, 정보보호학회지, 제27권, 제1호, 2017
 [2] Kumar Chellapilla, Kevin Larson, Patrice Simard and Mary Czerwinski, Computers beat Humans at Single Character Recognition in Reading based Human Interaction Proofs (HIPs), CEAS 2005 -

Second Conference on Email and Anti-Spam, Stanford University, California, USA, 2005
 [3] 김용대, 사람 인지 특성에 기반한 CAPTCHA 설계, 한국과학기술원, 2017
 [4] Zhong Wang, Peibei Shi CAPTCHA Recognition Method Based on CNN with Focal Loss, Hindawi, Volume 2021, Article 6641329, 2021