한국정보과학회
KOREAN INSTITUTE OF INFORMATION SCIENTISTS AND ENGINEERS

제 25 권 제 1 호
Vol. 25 No. 1

SOFTWARE ENGINEERING SOCIETY

2023

제 25 회 한국 소프트웨어공학 학술대회 논문집

Proceedings of the 25th Korea Conference on Software Engineering (KCSE 2023)

- 일시: 2023년 2월 8일(수) ~ 2월 10일(금)
- 장소: 강원도 평창 한화리조트(휘닉스파크점)

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

후원: ETRI, STA 테스팅컨설팅, KSTQB, SOLUTIONLINK
㈜비트컴퓨터, ㈜이에스지, ㈜다한테크,
㈜모아소프트, 브이플러스랩㈜, 슈어소프트테크㈜,
한국소프트웨어기술진흥협회(KOSTA),
한국정보통신기술협회, T3Q㈜, ㈜SPID

Bad Code 패턴의 지도 학습을 통한 Bad Code 식별 적용 사례

박찬솔 ○, 김장환, 문소영, 김영철
홍익대학교 소프트웨어공학연구실
{c2193102, janghwan.kim}@g.hongik.ac.kr, {whit2, bob}@hongik.ac.kr

Applied Practice on Identifying Bad Codes through Supervised Learning with Bad Code Patterns

Chansol Park○, Janghwan Kim, So Young Moon, R. Young Chul Kim
SELab., Hongik University

요 약

현재 고품질 소프트웨어를 위해 소프트웨어 가시화 기법으로 저품질 코드 영역을 식별하여 리팩토링 할 수 있도록 가이드한다. 기존의 소프트웨어 가시화 기법은 규칙을 질의어 화하여 이를 통해 품질 지표를 추출하고 가시화했다. 그러나 이제는 규칙을 통한 품질 측정보다 더 효율적인 방법을 찾고자 한다. 이를 위해 인공지능을 접목하는 시도가 필요하다. 현재 지도 및 비지도 학습 두 방향으로 접근하여 기존 규칙 기반의 품질 식별 및 측정 방법을 대체하려는 연구가 진행 중이다. 본 논문은 그 중 지도 학습을 통해 Bad Code 패턴을 학습하는 방법을 제안한다. 이를 통해 기존의 소프트웨어 공학적 접근 방법과 인공지능을 통한 접근 방법 중 더 효율적인 방법에 대한 파악이 기대된다. 또한 앞으로 기존의 순수 SE 품질 식별 방법과 인공지능의 지도 학습 및 비 지도 학습을 통한 패턴 분석 방법과 정확성 비교 및 검증이 필요하다.

1. 서 론

기존 순수 소프트웨어 가시화는 품질 지표에 대한 추출을 통해 저품질 코드 영역을 식별하고 이를 리팩토링 할 수 있도록 나타내어, 소프트웨어를 고품질화 하는 기법이다.

본 연구실에서는 소프트웨어 가시화 기법에 인공지능을 접목하여 고도화하는 연구를 진행하고 있다. 이를 위해 코드 품질에 대한 지도 학습 및 비지도 학습을 통해 기존의 규칙 기반의 품질 추출 방법의 한계를 극복하고자 한다. 본 논문에서는 그 중 Bad Code를 지도 학습하여 Bad Code의 패턴을 분석하는 모델을 제안한다. 해당 모델을 통해 규칙을 직접 구현하지 않고도 Bad Code를 식별하는 모델을 구현한다.

2. 기존 순수 코드 가시화 연구

코드 가시화는 코드를 분석하여 추출한 정보를 표와 다이어그램 등으로 가시화하는 것이다. 이를 통해 소프트웨어의 비 가시적인 특성을 극복하여 소프트웨어 개발 프로젝트에 관련된 이해 관계자가 알기 쉽게, 소프트웨어의 각 설계 및 복잡도, 결함 부위에 대한 가이드 등을 나타낼 수 있다[1,2].

본 연구실에서는 그림 1과 같이 3년간 소프트웨어 가시화에 인공지능을 접목하는 연구를 진행하고 있다. 연구의 1차년도에 소프트웨어의 품질 검증을 위한 정적 분석기 및 코드 가시화 기술을 개발하고, 2차년도는 BERT, GPT-3등의 자연어 처리 모델에 대한 학습을 통한 무결점 SW 코드 자동 생성 기술을 개발하고, 3차년도는 SW의 품질 역량 강화를 위한 무결점 코드 가시화 서비스를 구축한다.

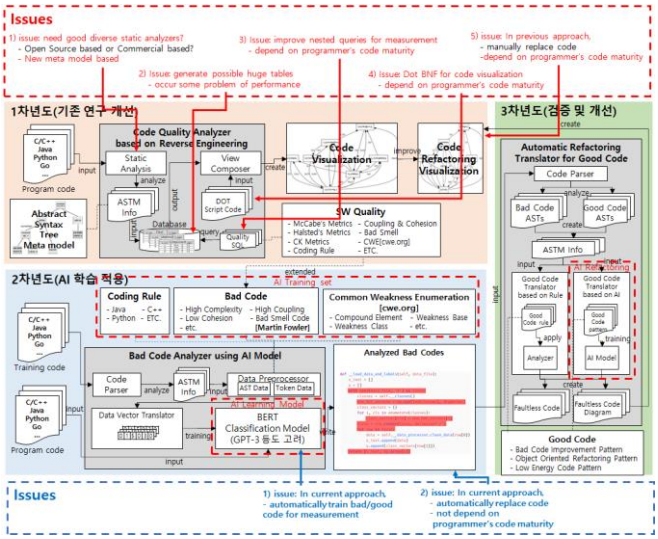

그림 1 코드 가시화 고도화 연구

본 연구실에서는 그림 1과 같이 3년간 소프트웨어 가시화에 인공지능을 접목하는 연구를 진행하고 있다. 연구의 1차년도에 소프트웨어의 품질 검증을 위한 정적 분석기 및 코드 가시화 기술을 개발하고, 2차년도는 BERT, GPT-3등의 자연어 처리 모델에 대한 학습을

통한 무결점 SW 코드 자동 생성 기술을 개발하고, 3차년도는 SW의 품질 역량 강화를 위한 무결점 코드 가시화 서비스를 구축한다.

3. 지도 학습 기반 Bad Code 패턴 학습

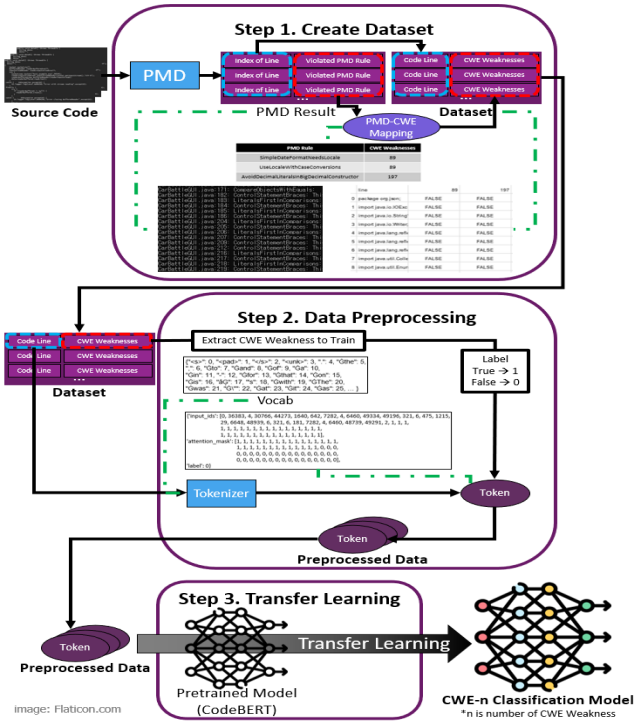

그림 2 CWE 식별 모델의 전이 학습 과정

전이 학습 과정에서는 크게 데이터 셋을 만드는 과정, 학습하기 위해 데이터 셋을 전 처리하는 과정 그리고 전이 학습을 진행하는 과정이 있다. 그림 3은 CodeBERT 모델에 CWE의 취약점을 전이 학습하기 위한 과정에 대한 구조도이다.

데이터 셋 생성: 인공 지능 모델에 CWE 취약점 항목을 학습하기 위해서는 각 코드 라인에 대해 CWE 위반 여부가 라벨링 된 데이터가 필요하다. 우선 PMD도구를 이용해 코드들을 분석하여 각 코드에 대한 PMD 규칙 위반 항목을 검출한다.

데이터 전 처리: 각 CWE 취약점 항목에 대한 전이 학습을 위해 앞선 과정에서 생성된 데이터 셋으로부터 코드 라인과 코드 라인에 대한 학습할 CWE 취약점 여부를 추출한다.

4. 적용 사례

적용사례로서 9280개의 Java 프로젝트로부터 50923개의 Java 코드에 대해 'CWE-400: Uncontrolled Resource Consumption' 항목에 대해 데이터 셋으로 만들어 검출 여부를 학습하였다. 학습에 이용한 데이터 셋에 대한 정보는 표 1과 같다. 취약점이 검출된 코드 라인은 전체 표본 코드 라인 수의 2.1%에 불과하다. Juliet Test Suit 데이터 셋[3]을 이용하여 모델의 정확도를 측정하였다. 모델의 식별 결과에 대한 혼동 행렬은 표 2와 같다.

CWE-400을 학습한 모델의 정확도는 41.9%에 불과하다. 이러한 낮은 정확도의 이유는 Juliet Test Suite의 구조와 연관이 있다. Juliet Test Suite는 Bad case와 Good case가 존재한다. Bad case의 경우 Bad Source로부터 Bad Sink로 데이터를 전달하는 경우이며, Good case의 경우 Bad Source로부터 Good Sink로 데이터를 전달하거나, Good Source로부터 Bad Sink로 데이터를 전달하거나 하는 경우가 포함된다. 따라서 Good case의 함수인 경우라도 Bad Source를 이용하거나 Bad Sink로 데이터를 전달하는 경우 취약점이 검출된 함수라고 식별하였다. Bad case에 대한 식별 정확도는 80.2%이다.

표 1 학습에 사용한 데이터의 정보

CWE Entry ID	Number of Labeled Data (line)		Number of TotalData (line)
400	False	11,642,208	11,893,126
	True	250,918	

표 2 CWE-400 식별 모델의 혼동 행렬

		예측 결과	
		미 검출	검출
실제 정답	미 검출	1,596	4,440
	검출	482	1,953

5. 결론 및 향후 연구

Bad Code 패턴을 지도 학습하여 Bad Code를 식별하는 방법을 제안하였다. 또한 CWE 취약점에 적용하여 Bad Code Pattern을 학습한 모델을 Juliet Test Suite의 분류를 통해 정확도를 검증하였다. 추후 더 많은 데이터에 대한 학습을 통해 검출 가능한 Bad Code의 종류를 늘이고 정확도를 향상하여, 규칙 기반의 도구와 비교를 통해 성능을 비교해 볼 예정이다.

ACKNOWLEDGMENT

이 논문은 교육부 및 한국연구재단의 4단계 두뇌한국21 사업의 지원(F21YY8102068)과 2022년도 정부(교육부)의 재원으로 한국연구재단의 지원(No. 2021R1I1A3050407, No. 2021R1I1A1A01044060)과 행정안전부 재난안전산업 기술사업화 지원 사업(RS-2022-00155579)의 지원을 받아 수행된 연구임.

참고 문헌

[1] Jung, Se Jun, et al. "Automatic UML Design Extraction with Software Visualization based on Reverse Engineering." International journal of advanced smart convergence 10.3 (2021): 89-96.

[2] 이원영, and 김영철. "코드 가시화 툴체인 기반 UML 설계 추출 및 검증 사례." 인터넷정보학회논문지 23.2 (2022): 79-86.

[3] Black, Paul E., and Paul E. Black. Juliet 1.3 test suite: Changes from 1.2. US Department of Commerce, National Institute of Standards and Technology, 2018.