초거대 언어모델을 활용한 소프트웨어 결함 예측 연구 동향

전북대학교 ▌류덕산*

1. 서 론

소프트웨어 신뢰도를 평가하는데는 다양한 방법들이 존재한다. 실행을 통해 실패수 (또는 실패 사이 시간)를 기반으로 소프트웨어 신뢰성 성장 모델 (Software Reliability Growth Model, SRGM) 같은 수학적 통계 모델을 적용하여, 직접적인 신뢰도를 측정하는 방법 외에도, 소프트웨어 개발 프로세스, 즉 요구사항, 설계, 구현, 테스팅 과정에서 수집한 산출물의 품질을 기반으로 소프트웨어의 신뢰도를 측정하는 간접적인 방법이 있다.

소프트웨어 결함 예측 연구는, 1950년대부터 시작된, 구현 단계에서의 품질을 측정하는 방법으로서, 소프트웨어 산출물을 기반으로 얻은 통찰력으로 효과적인 의사결정을 추구하는 소프트웨어 애널리틱스의 대표적인 적용 사례이다.

소프트웨어 결함 예측은, 소프트웨어 개발 생명 주기 (Software Development Life Cycle, SDLC)중의 구현/테스팅 단계에서, 가장 유용한 신뢰도 평가 활동 중의 하나로서, 결함도가 높은 개체 (소스코드파일, 바이너리, 모듈, 코드변경 등)를 식별하여, 품질 보증 활동(코드리뷰, 테스팅 등)이 결함도가 높은 개체들에 집중되도록 가이드함으로써, 소프트웨어 품질 제고 및 품질 보증 자원의 효과적 할당을 돕는다.

결함 예측을 위해서는, 소프트웨어의 구현 산출물에서, 결함과 연관지을수 있는 데이터 수집이 요구되며, 이때 Goal-Question-Metric (GQM) 기법이 유용하다. Goal은 측정의 목표이고, Question은 목표의 달성을 결정짓는 요소를 질문의 형태로 구성하고, Metric은 각 질문에 대해 정량적인 답안을 제시할 수 있는 척도를 도출한다.

소프트웨어 데이터는, 크게 문맥 데이터 (프로젝트

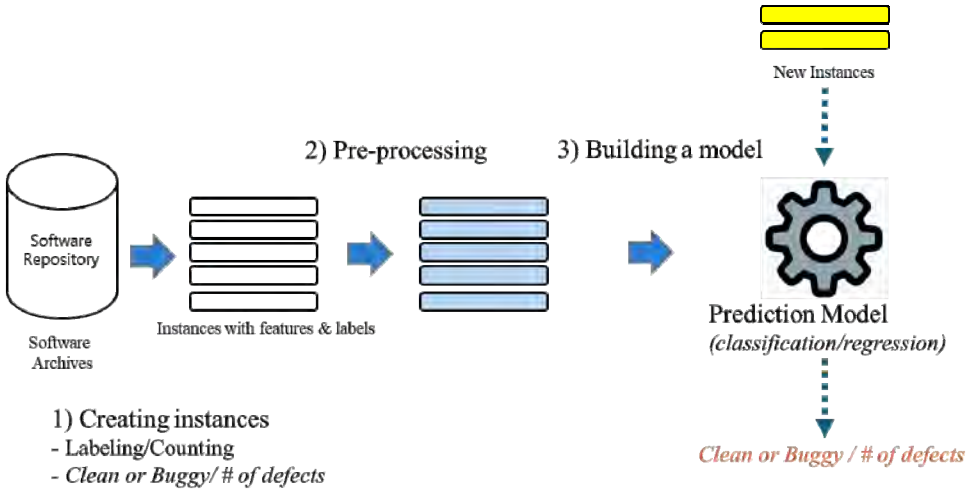

그림 1 소프트웨어 결함 예측 프로세스

* 종신회원

† 본 연구는 원자력안전위원회의 재원으로 한국원자력안전재단의 지원을 받아 수행한 원자력안전연구사업(No. 2105030)과 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(NRF- 2022R1I1A3069233)과 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업 (IITP-2024-2020-0-01795)의 연구결과로 수행되었음

목표, 조직구조등), 제약 데이터(호환성 척도, 성능 척도등), 개발 데이터의 3가지로 구분하는데, 결함 예측에 가장 많이 사용되는 데이터는 개발 데이터이다. 개발 데이터는, 코드 규모, 복잡도, 의존도, 코드의 추가/변경/삭제 내역 등을 포함하는 코드 자체 정보외에도 개발자들 간의 메일링 리스트, 버그 리포트 정보도 포함한다.

소프트웨어 결함 예측 프로세스는 그림1과 같이, 3단계로 구성되어 있다. 1단계는 인스탄스 생성으로, 소프트웨어 산출물로부터 소프트웨어 척도와 결함/비결함 라벨 또는 결함의 개수를 추출한다. 2단계 전처리 과정에서는, 모델의 성능 향상을 위해 데이터 정규화, 특징 선택, 노이즈 제거를 수행한다. 3단계 모델링 단계에서, 분류 모델은 결함/비결함을 예측하고, 회귀 모델은 결함의 수를 예측한다. 결함 예측을 위해서, 과거 전통적인 기계학습 기법부터 최신의 딥러닝 기법들까지 폭넓게 적용되고 있다.

소프트웨어 결함 예측은 소프트웨어 개발 생명주기의 단계중, 개인 개발자의 단위 시험이 완료된후, 소프트웨어 시스템을 통합후에, 통합테스팅의 직전 단계에서 소스코드의 정적분석 결과로 얻은 척도들을 사용하는 방법이 먼저 개발되었다. 이후, 개인 개발자가 커밋 수준(코드의 추가/변경/삭제 내역)으로, 결함을 예측하는 적시결함예측 (Just-In-Time Defect Prediction, JIT-DP) 방법이 고안되었다.

또한 예측 모델에 훈련할 데이터의 가용성 여부에 따라, 내부프로젝트 결함예측(Within-Project Defect Prediction, WPDP) 과 교차프로젝트 결함예측(Cross-Project Defect Prediction, CPDP)으로 구분된다. WPDP는 동일 프로젝트에 충분한 과거 결함 데이터가 존재

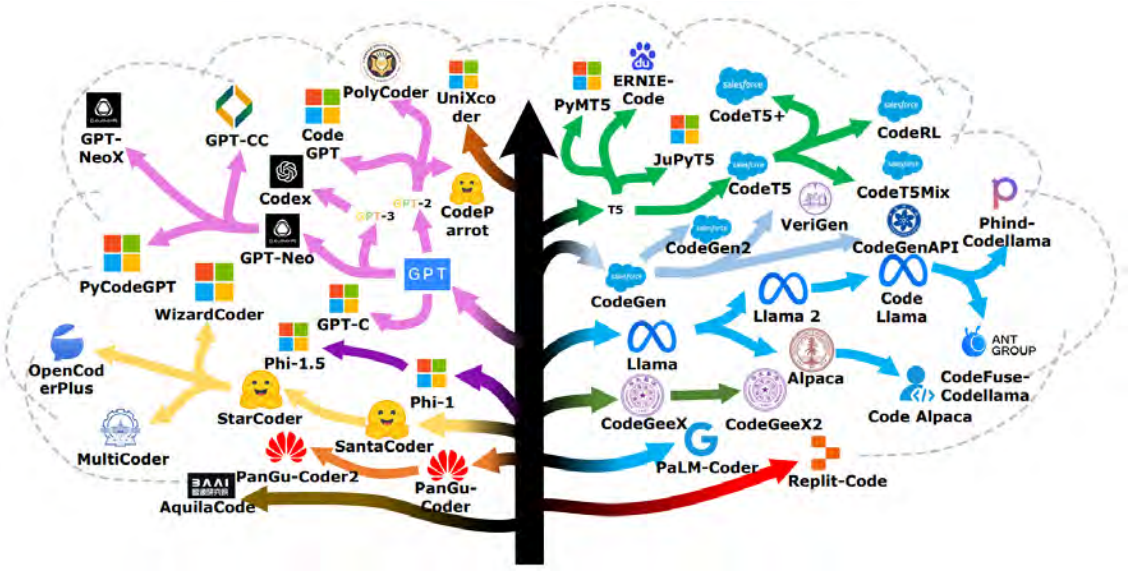

그림 2 Large Language Models for Code [1]

표 1 LLM4SE 주요 작업 [20]

작업	정의	LLM의 역할
코드 생성	사용자 요구사항 및 지정된 제약조건을 기반으로 소스코드 자동생성	(보조) 코드를 생성하거나 개발자에게 아이디어와 프로그래밍 '시작점'을 제공
코드 요약	개발자가 코드를 이해하고 유지관리하는데 도움이 되는 명확하고 정확하며 유용한 주석을 자동으로 생성	다양한 세부사항(예: 함수)을 지원하거나 코드의 의도 등을 설명하는 코드 요약
코드 번역	기능이나 논리를 변경하지 않고 다른 프로그래밍 언어간에 코드를 변환	보조 코드 변환, 리버스 엔지니어링 등
취약점 탐지	프로그램 충돌, 성능 저하 또는 보안 문제를 일으킬 수 있는 코드 오류를 식별하고 수정	코드 등의 잠재적인 취약점을 확인
코드 평가	잠재적인 문제와 개선점을 식별하기 위해 코드에 대한 정적분석을 수행	테스트케이스 또는 테스트 코드 성능, 사용성 및 기타 지표 등을 생성
코드 관리	코드 버전, 개발자 등 정보관리	팀 협업 개발, 버전 관리 등
Q&A 상호작용	소프트웨어 개발자와 LLM간의 상호작용	프로그램 보조, 프롬프트 엔지니어링

하는 경우로, 훈련 및 테스트에 모두 로컬 프로젝트의 데이터가 이용된다. 이때 한 개 프로젝트에서 버전이 업그레이드될 때 적용될 경우, 이를 명확히 하기 위해 교차버전 결함예측(Cross-Version Defect Prediction, CVDP)으로 불린다. 이와달리, CPDP는, 과거 결함 데이터가 없는, 최초 버전 프로젝트에 적용하기 위해, 다른 프로젝트 데이터를 이용하여 모델의 훈련을 하고, 테스트에는 로컬 프로젝트 데이터가 이용된다. 이 경우, 훈련 데이터와 테스트 데이터 사이의 분포가 다르기 때문에, 분포차를 줄일 수 있는 전이학습 기법을 적용한 모델을 고안하는 것이 주요한 연구 방법이다. CPDP는 훈련 데이터와 테스트 데이터 사이의 분포가 다르지만, 두 데이터간의 특징 공간은 동일하다. 이러한 CPDP보다 일반화된 기법으로는, 이종결함예측(Heterogeneous Defect Prediction, HDP)이 있으며, 훈련 데이터와 테스트 데이터간의 특징 공간이 다른 경우이다. 이 경우, 비슷한 척도끼리 매칭시키는 척도 매칭이나, 척도를 다른 차원으로 변환시키는 척도 변환방법이 주로 연구에 사용된다.

초거대 언어모델 (Large Language Model, LLM)은 다양한 소프트웨어공학 문제에 적용되고 있다. [1]은 LLM을 코드에 적용한 연구에 대해 종합적으로 조사/분석하였다. 그림2는 코드용으로 만든 LLM 모델들의 현황을 분석한 결과를 보여준다. Codex, UniXCoder, CodeT5+, Code Llama, Code Alpaca 등 다양한 코드용 LLM 이 개발되고 있음을 확인할 수 있다. 또한 [1]은 코드 생성, 테스트케이스 생성, 코드 요약, 취약점 교정, 코드 번역 등의 작업에 대해 LLM 모델의 성능 평가 및 벤치마킹 결과를 제시하고 분석하였다.

[20]은 소프트웨어공학 (Software Engineering, SE) 문제에 LLM이 적용된 연구들의 현황 및 평가 결과를 분석한다. 참고문헌들을 분석후, LLM4SE (Large Language Model for Software Engineering) 주요 작업을 7개로 분류하였다. 표1은 LLM4SE 주요 작업들을 나타낸다. 주요 작업들은 (1) 코드 생성, (2) 코드 요약, (3) 코드 번역, (4) 취약점 탐지, (5) 코드 평가, (6) 코드 관리, (7) Q&A 상호작용이다.

[8]은 소프트웨어공학분야에서 초거대 언어 모델의 적용 현황을 조사하였다. 요구사항, 설계, 코드 생성, 코드 완성, 소프트웨어 테스팅, 유지보수 배포, 문서 생성 등의 작업에서 적용된 LLM의 사례를 보였다. 해결해야 할 연구 주제들로는 다음을 제시했다: (1) 소프트웨어 공학용 LLM 모델 구축 및 튜닝, (2) 동적 프롬프트 최적화, (3) 기존 SE 프레임워크에 LLM 통합, (4) 환각문제, (5) 신뢰성있는 평가방안, (6) 포괄적인 테스팅, (7) 장문의 텍스트 입력 처리, (8) 요구사항공학, (9) 설계 및 리팩토링 같은 다루어지지 않은 소프트웨어 공학 태스크.

본 연구에서는, 초거대 언어모델이 소프트웨어 결함 예측에 활용되고 있는 최신 연구 동향을 알아본다. 2장에서 기계학습 및 딥러닝 모델이 활용된 연구들을 확인하고, 3장에서 초거대 언어모델이 활용된 연구들을 확인한다. 4장에서는 미래 연구 방향을 제시하고, 5장에서 결론으로 마무리한다.

2. 기계학습 및 딥러닝 모델을 활용한 연구들

[15]는 소프트웨어공학 분야에서 인공지능의 활용 사례를 분석한다. 15개의 연구분야에 대해 다음과 같이 5개의 테마로 분류하였다 : (1) 소프트웨어공학의 자연어 처리, (2) 소프트웨어 개발 생명주기 관리에 인공지능 사용, (3) 결함예측 및 노력 추정에 기계 학습 사용, (4) 지능형 소프트웨어 공학 및 코드 관리에 딥러닝 사용, (5) 품질 향상을 위한 소프트웨어 저장소 마이닝. 이 연구의 결과를 통해, 소프트웨어 결함 예측이 인공지능의 주요 적용 사례임을 확인할 수 있다.

[2]는 소프트웨어 결함 예측 분야에서 기계학습기법 대비 딥러닝 기법의 적용 사례를 분석하였다. 최다빈도의 5개 성능 지표는 F-measure, Recall, Accuracy, Precision, AUC 이고, 5개 모델은 컨볼루션 신경망(Convolution Neural Network, CNN), 심층신경망(Deep Neural Network, DNN), 장단기 기억 네트워크(Long Short Term Memory, LSTM), 심층신뢰망(Deep Belief Network, DBN), 스택 디노이징 (Stacked Denoising AutoEncoder, SDAE)가 사용되었다. 최다빈도의 데이터셋은 PROMISE 와 NASA 데이터셋이고, 딥러닝 모델의 Accuracy, F-measure, AUC 성능은 일반적으로 전통적인 기계학습 모델보다 우수한 것으로 나타났다.

3. 초거대 언어모델을 활용한 연구들

[17]은 소프트웨어 결함예측에 LLM의 적용가능성을 알아보기 위해, SDP가 가지고 있는 특성과 LLM이 적용된 연구성과를 분석하여 서로 일치하는점을 확인한다. 소프트웨어 결함 예측(Software Defect Prediction, SDP)은 이상 탐지와 유사하게 시스템의 비정상적인 상황을 감지하는 것을 목적으로 가지고 있다. SDP에 주로 사용되는 데이터셋은 정형데이터셋이다. 이러한 특성들을 기반으로, BERT, GPT-2와

같은 모델들이 이상탐지에 활용된 사례가 존재하며, 정형데이터에 LLM을 적용하는 사례들을 확인할 수 있었다. 이를 통해, LLM을 SDP에 적용하는 연구들이 유망함을 확인할 수 있었다.

[7]은 자율주행 소프트웨어 (Appollo, Donkeycar 등)를 대상으로, 딥러닝 기법을 적용하여, 적시결함예측을 수행하였다. 커밋 메시지와 코드 변화 정보를 임베딩하기 위해, UniXCoder 기법을 적용하고, 문맥과 의미 학습을 위해 양방향 LSTM (Bi-directional LSTM, Bi-LSTM)을 적용하였다. 랜덤포레스트 (Random Forest), 그래디언트 부스팅 결정 트리 (Gradient Boosting Decision Tree), 로직스틱 회귀 (Logistic Regression), 익스트림 그래디언트 부스팅 (xTreme Gradient Boosting, XGBoost) 등의 기계학습 기법외에도 DeepJIT, CC2Vec, JITLine, CodeBERT4JIT 기법과 비교하여, 노력인식 성능 지표 (effort-aware performance metric)와 노력 비인식 성능지표(effort-unaware performance metric)에서 모두, 효과크기 검정 (effect-size test)을 통해 우수한 성능을 검증하였다.

[12]는 엣지컴퓨팅에서 사용되는 소프트웨어 (EdgeX-go, Kubeedge, Openshift, Traefik)를 대상으로, 사전학습 모델을 적용하여, 함수 수준의 적시결함예측 모델을 평가한다. CodeBERT, GraphCodeBERT, UniXCoder 모델들을 적용하여, 내부프로젝트 및 교차프로젝트 환경에서 비교한다. AUC 성능지표를 기준으로, 내부프로젝트 환경에서, UniXCoder 가 우수한 성능을 보였지만, 교차프로젝트 환경에서는 CodeBERT 가 상대적으로 우수한 성능을 보였다.

[18]은 소프트웨어 결함 예측을 위해 CodeBERT 모델을 사용한 지도형 대조 학습 기법을 제안한다. 이 기법은 CodeBERT 모델을 적용하여, 소스코드로부터 의미 특징을 포착하고, 대조 학습을 통해, 유사한 데이터쌍 간의 유사성을 최대화하고, 서로 다른 데이터쌍 간의 유사성을 최소화하여 데이터에서 가치 있는 정보를 추출한다. 제안한 모델은, PROMISE 데이터셋에 대해 F-measure 성능 지표를 기준으로 DBN, CNN, LSTM 보다 우수한 성능을 보였다.

[3]은 엣지 클라우드 환경에서 사용되는 Ansible 스크립트의 코드 냄새 (잠재적인 결함을 암시하는 코드 패턴)를 LLM 프롬프트에 통합하여 결함 감지를 향상하는 코드냄새 안내 프롬프팅 (Code-Smell-Guided-Prompting, CSP)을 제안한다. 프롬프트 기법으로, 제로샷 (zero shot), 퓨샷 (few shot), 제로샷-CSP (zero shot-CSP), 생각의사슬-CSP (Chain-of-Thought-CSP)를 적용하여, ChatGPT, GPT-4, Gemini 1.0 모델에 적용하여, F-measure, Accuracy 성능을 비교하였다. LLM 및 프롬프트 기법 선택에 따라, 성능이 다르므로, 일관적인 성능이 나올수 있는 기법 연구가 필요함을 보였다.

[4]는 코드 변경 학습에서 매개변수 효율적 미세조정 (Parameter Efficient Fine-Tuning, PEFT)의 성능을 연구하였다. 이 연구는 적시결함예측 작업에 대해, PEFT기법이 전체모델 미세조정 (Full-Model Fine-Tuning, FMFT)보다 더 적은 계산 비용으로 더 나은 성능을 제공할 수 있음을 보였다. 어댑터 조정 (Adapter Tuning, AT)와 낮은순위조정 (LowRank Adaptation, LoRA)을 CodeBERT, GraphCodeBERT, PLBART, UniXCoder, CodeT5 모델에 적용하였으며, 코드 변환 표현 정보외에도 전문가 특징 정보를 추가 입력으로 주었을 때, 예측 성능이 향상됨을 보였다.

[5]는 사전학습된 코드 지능 언어 모델에 대해 유전적 자동 프롬프트 (Genetic Auto-Prompt, GenAP) 학습 방법을 제안한다. GenAP를 결함 예측, 코드 요약 및 코드 번역 작업에 수행하여, 유용성을 검증하였다. 특히, 결함 예측에는 CodeBERT 모델의 입력 프롬프트에 GenAP기법을 적용하여, 다른 프롬프트 기법들에 비해 성능이 우수함을 보였다. GenAP를 사용하면 비전문가도 꼼꼼하게 수동으로 디자인한 프롬프트에 비해 우수한 프롬프트를 쉽게 생성할 수 있는 장점을 가지고 있다.

[9]는 메서드 수준에서 버그 심각도를 예측하기 위해 소스 코드 척도와 LLM을 사용하는 연구이다. LLM은 CodeBERT를 사용하였다. 입력으로, 버그가 있는 메소드 만을 사용하여, 버그의 심각도를 예측한다. 또한, 버그가 있는 메소드 외에도 메소드 수준 소스코드 척도를 같이 통합하여, 입력으로 사용할 경우, 예측 성능이 향상되는 결과를 보였다. 이를 통해, 사전학습 모델은 특정 태스크에서 최고의 성능을 수행하기 위해서는, 추가적인 정보/문맥과 미세조정이 요구됨을 확인하였다.

[10]은 소프트웨어 결함 예측에 4가지 버전의 CodeBERT 모델을 적용하여 예측 성능을 분석한다. 교차버전 및 교차프로젝트 환경에서 실험 결과, 사전학습된 CodeBERT 모델이 예측 성능 향상 및 시간 비용을 절감할 수 있음을 보인다. 문장 기반 및 키워드 기반 예측 패턴을 활용할 때, CodeBERT 의 결함 예측력에 향상을 가져왔다.

[11]은 코드 인텔리전스 작업에서 프롬프트 튜닝의

사용법과 효과를 평가한다. 사전 훈련된 모델은 자동 코드 요약 및 결함 예측과 같은, 코드 인텔리전스 작업에서 효과적인 것으로 나타났다. 이러한 모델은 라벨이 지정되지 않은 대규모 코퍼스에서 사전 훈련된 다음, 다운스트림 작업에서 미세 조정된다. 그러나 사전 훈련 및 다운스트림 작업에 대한 입력의 형태가 다르기 때문에 사전 훈련된 모델에 대한 지식을 완전히 탐색하기는 어렵다. 게다가 미세 조정 성능은 다운스트림 데이터의 양에 크게 좌우되지만 실제로는 데이터가 부족한 시나리오가 일반적이다. 자연어 처리(Natural Language Processing, NLP) 분야의 최근 연구에 따르면 튜닝의 새로운 패러다임인 프롬프트 튜닝(Prompt Tuning)이 위의 문제를 완화하고 다양한 NLP 작업에서 유망한 결과를 달성하는 것으로 나타났다. 프롬프트 튜닝에서는 튜닝 중에 삽입된 프롬프트가 작업별 지식을 제공하며, 이는 상대적으로 데이터가 부족한 작업에 특히 유용하다. [11] 연구에서는, 사전 훈련된 CodeBERT 및 CodeT5에 대해 프롬프트 튜닝을 수행하고, 이를 결함 예측, 코드 검색, 코드 요약 및 코드 번역 작업에 적용한다. 실험결과, 프롬프트 튜닝이 미세조정보다 성능이 우수함을 보여준다. 결함 예측을 포함하는 코드 인텔리전스 작업에 대해 미세 조정 대신 프롬프트 튜닝을 적용하여 특히 작업별 데이터가 부족한 경우 더 나은 성능을 얻을 수 있음을 보였다.

[13]은 사전 학습된 LLM을 기반으로한 매개변수 효율적인 다중 분류 소프트웨어 결함 탐지 방법을 제안한다. 이 기법은 사전학습된 CodeT5+와 $(IA)^3$을 활용하였다. 사전학습된 CodeT5+는 결함이 발생하기 쉬운 특징들을 포착하는 동시에 코드 표현을 생성한다. 사전 학습된 LLM의 높은 오버헤드를 고려하여 $(IA)^3$ 벡터를 특정 레이어에 주입하고, 주입된 매개변수만 업데이트하여 학습 비용을 줄인다. 또한 사전 학습된 CodeT5+의 속성을 활용하여 소스 코드와 자연어기반 전문가 척도를 결합하여 입력 데이터를 풍부하게 하는 새로운 기능 시퀀스를 설계한다. 실험결과, F1-가중치, Recall-가중치, Precision-가중치, 매튜스상관계수(Matthews Correlation Coefficient, MCC) 성능 지표에서, 모두 우수한 성능을 보였다.

[14]는 코드 언어 모델을 활용하여 소프트웨어 결함 예측을 수행하는 연구를 다루고 있다. 소프트웨어 공학 영역에서는 코드 언어 모델이 광범위한 코드 관련 다운스트림 작업에 크게 도움이 되는 것으로 나타났다. 이 논문에서는 결함 예측을 위해 고유한 아키텍처를 갖춘 코드 언어 모델을 사용한다. 의미 정보의 이해를 높이기 위해 통합된 이중 모드 입력 표현을 제안한다. 또한 PROMISE 저장소와 엔지니어링 파일들을 기반으로 새로운 이중 모드 데이터 세트가 제안되었다. 세 가지 아키텍처 (CodeBERT, CODEGEN, CodeT5+) 중에서 인코더 전용 아키텍처 (CodeBERT)가 가장 효과적이고 효율적이었다.

4. 미래 연구 방향

[19]는 소프트웨어공학의 미래를 재구상하고 다음 세대를 위한 연구 및 개발 로드맵을 제시한다. 6가지 주요 연구 초점 영역중의 하나인, 인공지능으로 증강된 소프트웨어 개발 영역에서, 인공지능을 적극적으로 활용해야 할 세부 연구 주제로 결함 식별을 들었다. 결함 분석 및 예측을 통해, 개발자들이 적시에 정확한 의사 결정을 수행할 수 있음을 강조했다.

[16]은 인공지능을 소프트웨어공학에 적용하는 현황과 미래방향을 제시하고 있다. AI4SE (Artificial Intelligence for Software Engineering)가 중심이 되는 소프트웨어공학 2.0의 미래를 제시하며, 두가지 전략적 로드맵으로, (1) 신뢰할 수 있는 AI4SE와 (2) 협력적인 AI4SE를 제시한다. 특히 LLM4SE (Large Language Model for Software Engineering)의 효과를 높이기 위해서, LLM의 입력 및 출력을 강화하는 방향을 제시한다. 예를 들어, 코드 그래프 (콘트롤 플로우 그래프 및

표 2 LLM4SDP 미래 연구 방향

연구주제	내용	참고문헌
결함예측을 위한 프롬프트 엔지니어링	LLM 입력강화, 프롬프트 설계, 설계 자동화	[5]
다양한 데이터 소스 활용	LLM 입력강화, 결함도와 연관성 있는 다양한 입력데이터 활용	[3][7][9][14]
계산 비용 절감	매개변수 효율적 미세 조정, 모델 압축, 입력데이터 압축	[4][13]
새로운 LLM 출력 결과 적용	LLM 출력강화, 새로운 출력 결과 (결함예측용 코드생성)	[6]
데이터 부족문제 해결	프롬프트 튜닝, LLM으로 추가 데이터 생성	[11]
정형데이터에 적합한 LLM	결함예측용 데이터셋의 특성 (정형데이터)에 효과적인 LLM 적용	[17]

프로그램 의존도 그래프 등)를 LLM에 추가 입력으로 사용하는 방안이 있다. 또한, LLM에 적합한 입력이 되도록 하기 위해, 소스코드를 중간 표현 (Intermediate Representation, IR)으로 변경하는 방법도 있다.

지금부터는 참고문헌을 분석한 결과를 바탕으로, LLM 기반 소프트웨어 결함 예측 분야에서의 미래 연구 방향을 제시한다. 표2는 LLM4SDP (Large Language Model for Software Defect Prediction)의 미래 연구 방향을 요약한 결과이다.

4.1 결함 예측을 위한 프롬프트 엔지니어링

LLM 적용시 성능에 큰 영향을 주는 요소중 하나가, LLM의 입력인 프롬프트를 어떻게 구성하느냐이다. 따라서, 소프트웨어 결함 예측에 효과적인 프롬프트를 설계하는 것은 결함도 예측력 향상에 중요하다. 특히 [5]는 유전알고리즘을 활용한 자동 프롬프트 설계 기법을 제시하였는데, 이처럼 자동으로 설계하는 기법은 비전문가가 프롬프트 설계에 대한 전문지식이 없이도 LLM을 적용하여 결함 예측을 하는데 유용하다.

4.2 다양한 데이터 소스 활용

[3][7][9][14]의 예처럼, LLM의 입력으로 추가정보/문맥을 활용하는 방식은 결함 예측력 향상에 유용하다. 코드 척도, 코드 자체, 코드 주석, 코드의 내부 구조 정보 (데이터 플로우, 컨트롤 플로우 등)외에도, [3]의 코드 냄새등 다양한 추가 정보를 LLM의 입력으로 사용했을 때, 결함 예측력을 향상시킬 가능성이 높다.

4.3 계산 비용 절감

어댑터 조정(AP), 낮은순위적응(LoRA)과 같은 매개변수 효율적 미세 조정 (PEFT), 모델의 압축, 입력 데이터 압축 등을 활용하여, 학습 및 운영시의 계산 비용을 절감한다면 결함예측분야에서의 LLM 적용을 확산시킬 수 있다. [4]는 PEFT가 적시결함 예측모델에도 효과적임을 실험적으로 검증하였다.

4.4 새로운 LLM 출력 결과 적용

[6]은 LLM으로 코드 모델을 생성하는 귀납적-편향 학습(Inductive-Biased Learning, IBL) 기법을 제안하고 있다. 이 기법은 두단계 (학습 및 추론)로 이루어진다. IBL의 학습 단계에서는 주어진 훈련 데이터를 사용하여 설명 변수에서 목적 변수를 출력하는 Python 함수로 표시되는 코드를 생성한다. 이 생성된 Python 코드를 “코드 모델”이라고 부른다. IBL의 추론 단계는 학습에서 생성된 Python 코드를 사용하여 수행된다. 설명 변수를 입력으로, 확률 값을 출력하는 Python 코드가 실행된다. IBL은 상황내 학습 (In-Context Learning, ICL) 과 코드 생성 기법을 결합한 새로운 학습 기법이다. 해석능력과 추론 속도 측면에서, 기존 기계 학습 기법을 대체할 수 있을 것으로 예상된다.

4.5 데이터 부족문제 해결

라벨링 데이터를 구축하는 데는 매우 많은 비용이 요구된다. 오직 숙련된 개발자들만이 고품질의 라벨을 생성할 수 있기 때문이다. 또한 소프트웨어 프로젝트에 특화된 데이터가 필요한 경우도 있다. 프롬프트 튜닝에서는 튜닝 중에 삽입된 프롬프트가 작업별 지식을 제공하며, 이는 상대적으로 데이터가 부족한 작업에 특히 유용하다. [11]은 프롬프트 튜닝 기법을 결함예측에 적용하여, 우수성을 실험적으로 검증하였다. 또한 데이터 부족문제를 해결하는데, LLM이 결함 예측에 효과적인 데이터를 생성하는데 활용될 수 있다.

4.6 정형데이터에 적합한 LLM

[17]에서 소개했듯이, 소프트웨어 결함 데이터셋은 대부분 정형데이터로 구성된 표 데이터이다. 표 데이터셋에 대해 LLM을 적용하는 연구가 활발히 이뤄지고 있고, 이러한 표 데이터에 효과적인 LLM 모델을 SDP 에 적용하는 연구도 필요하다.

5. 결 론

소프트웨어 결함 예측은 소프트웨어 품질 제고를 위한 기법으로 활발히 연구되고 있다. 결함도가 높은 소스코드파일, 모듈, 커밋 등을 식별하여 품질 보증 활동을 집중하게 가이드함으로서 소프트웨어 품질을 높이고 자원의 효과적인 할당을 돕는다. 소프트웨어 결함 예측은 다른 산업 분야의 이상치 검출 연구와 동일하게, 최신 인공지능 기술이 가장 적극적으로 활용되는 분야이다. 결함 예측력을 향상시키기 위해, 전통적인 기계학습 모델 (Logistic Regression, Support Vector Machine, Random Forest 등)외에도, 딥러닝 모델 (CNN, DNN, DBN, LSTM 등) 이 광범위하게 연구되었다. LLM은 출현이후, 소프트웨어 공학의 다양한 문제들, 즉 코드 생성, 테스트 케이스 생성, 코드 요약, 취약점 교정 등에 폭넓게 적용되고 있다.

본 논문에서는 LLM이 소프트웨어 결함 예측 분야에 활용된 연구 동향을 조사하고, 미래 연구방향을 제시한다. LLM으로는 소스코드에 특화된 CodeBERT, GraphCodeBERT, UniXCoder, CodeT5+ 가 주로 활용

되었고, 범용의 ChatGPT, GPT-4, Gemini 모델을 SDP에 적용하는 연구도 확산되고 있다. 결함 예측에 효과적인 프롬프트 엔지니어링 방식을 찾거나, 다양한 데이터 소스를 활용하여, 성능을 개선하려는 연구가 수행되었다. 매개변수 효율적 미세조정이 SDP의 효율/효과성을 개선하는지 여부를 확인하는 연구, 프롬프트를 자동으로 설계하는 연구, 데이터 부족시에 유용한 프롬프트 튜닝 기법의 적용 사례 연구등이 제시되었다.

이러한 적용 사례를 분석한 결과를 바탕으로, LLM 기반 소프트웨어 결함예측 분야에서의 유망한 미래 연구 방향은 다음과 같다.

1) 결함 예측을 위한 프롬프트 엔지니어링
2) 다양한 데이터 소스 활용
3) 계산 비용 절감
4) 새로운 LLM 출력 결과 적용
5) 데이터 부족문제 해결
6) 정형데이터에 적합한 LLM

본 연구의 결과를 바탕으로, 후속 연구자 및 실무자들이, LLM4SE 및 LLM4SDP분야에 더 많은 관심을 가지고, 연구를 수행하기를 기대한다.

참고문헌

[1] Zheng, Z., Ning, K., Wang, Y., Zhang, J., Zheng, D., Ye, M., & Chen, J. (2023). A survey of large language models for code: Evolution, benchmarking, and future trends. arXiv preprint arXiv:2311.10372.

[2] Zain, Z. M., Sakri, S., & Ismail, N. H. A. (2023). Application of deep learning in software defect prediction: systematic literature review and meta-analysis. Information and Software Technology, 158, 107175.

[3] Hong, H., Lee, S., Ryu D., & Baik, J. (2024). Enhancing Software Defect Prediction in Ansible Scripts using Code-Smell-Guided Prompting with Large Language Models in Edge-Cloud Infrastructures. The 4th International Workshop on Big Data-driven Edge Cloud Services (BECS)

[4] Liu, S., Keung, J., Yang, Z., Liu, F., Zhou, Q., & Liao, Y. (2024). Delving into Parameter-Efficient Fine-Tuning in Code Change Learning: An Empirical Study. arXiv preprint arXiv:2402.06247.

[5] Feng, C., Sun, Y., Li, K., Zhou, P., Lv, J., & Lu, A. (2024). Genetic Auto-prompt Learning for Pre-trained Code Intelligence Language Models. arXiv preprint arXiv:2403.13588.

[6] Tanaka, T., Emoto, N., & Yumibayashi, T. (2023). Inductive-bias Learning: Generating Code Models with Large Language Model. arXiv preprint arXiv:2308.09890.

[7] Choi, J., Kim, T., Ryu, D., Baik, J., & Kim, S. (2023). Just-in-Time Defect Prediction for Self-driving Software via a Deep Learning Model. Journal of Web Engineering, 22(2), 303-326.

[8] Fan, A., Gokkaya, B., Harman, M., Lyubarskiy, M., Sengupta, S., Yoo, S., & Zhang, J. M. (2023, May). Large language models for software engineering: Survey and open problems. In 2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE) (pp. 31-53). IEEE.

[9] Mashhadi, E., Ahmadvand, H., & Hemmati, H. (2023, October). Method-level bug severity prediction using source code metrics and LLMs. In 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE) (pp. 635-646). IEEE.

[10] Pan, C., Lu, M., & Xu, B. (2021). An empirical study on software defect prediction using codebert model. Applied Sciences, 11(11), 4793.

[11] Wang, C., Yang, Y., Gao, C., Peng, Y., Zhang, H., & Lyu, M. R. (2023). Prompt tuning in code intelligence: An experimental evaluation. IEEE Transactions on Software Engineering.

[12] Kwon, S., Lee, S., Ryu, D., & Baik, J. (2023). Pre-trained model-based software defect prediction for edge-cloud systems. Journal of Web Engineering, 22(2), 255-278.

[13] Wang, X., Lu, L., Yang, Z., Tian, Q., & Lin, H. (2024). Parameter-Efficient Multi-classification Software Defect Detection Method Based on Pre-trained LLMs. International Journal of Computational Intelligence Systems, 17(1), 152.

[14] Song, W., Gan, L., & Bao, T. (2023, November). Software Defect Prediction via Code Language Models. In 2023 3rd International Conference on Communication Technology and Information Technology (ICCTIT) (pp. 97-102). IEEE.

[15] Kokol, P. (2024). The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of the Recent Research Literature. Information, 15(6), 354.

[16] Lo, D. (2023, May). Trustworthy and synergistic artificial intelligence for software engineering: Vision and

roadmaps. In 2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE) (pp. 69-85). IEEE.

[17] 이정화, 주은정, 류덕산 (2024). 소프트웨어 결함 예측에서의 LLM 적용 가능성에 대한 고찰, 2024 한국컴퓨터종합학술대회 (Korea Computer Congress 2024)

[18] Sahar, S., Younas, M., Khan, M. M., & Sarwar, M. U. (2024). DP-CCL: A Supervised Contrastive Learning Approach Using CodeBERT Model in Software Defect Prediction. IEEE Access.

[19] Carleton, A., and Klein, M., and Robert, J., and Harper, E., and Cunningham, R., and de Niz, D., and Foreman, J., and Goodenough, J., and Herbsleb, J., and Ozkaya, I., and Schmidt, D., and Shull, F., Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research & Development, Carnegie Mellon University, Software Engineering Institute's Digital Library, https://insights.sei.cmu.edu/library/architecting-the-future-of-software-engineering-a-national-agenda-for-software-engineering-research-development/

[20] Zheng, Z., Ning, K., Chen, J., Wang, Y., Chen, W., Guo, L., & Wang, W. (2023). Towards an understanding of large language models in software engineering tasks. arXiv preprint arXiv:2308.11396.

약 력

류 덕 산

2011 미국 카네기멜론대 소프트웨어공학과정 졸업 (석사)
2012 한국과학기술원 소프트웨어공학과정 졸업 (석사)
2016 한국과학기술원 전산학부 졸업 (박사)
2016~2018 한국과학기술원 전산학부 연구조교수
2018~현재 전북대학교 소프트웨어공학과 부교수
관심분야: 소프트웨어공학, 인공지능, AI4SE, LLM4SE, SE4AI, SE4LLM
Email : duksan.ryu@jbnu.ac.kr
Web : https://sites.google.com/view/aiselabjbnu