논문 2024-4-3 http://dx.doi.org/10.29056/jsav.2024.12.03

소스코드 유사성 평가에서 딥러닝 활용 방안

김유경*†

Application of Deep Learning in Source Code Similarity Assessment

Yukyong Kim*†

요 약

소스 코드 유사성을 평가하는 데 딥러닝을 적용하는 것은 최근 몇 년 동안 관심 분야로 떠오르고 있다. 이러한 접근 방식은 딥러닝 모델이 소스 코드의 다양한 표현에서 자동으로 학습하고 기능을 추출하는 능력을 활용하여 유사성 탐지 작업의 정확도와 효율성을 개선할 수 있다는 측면에서 매우 긍정적이기는 하지만, 딥러닝을 활용한 유사도 감정은 결과의 정확성, 공정성, 해석 가능성에 대한 문제를 안고 있다. 본 논문에서는 딥러닝 기반 소프트웨어 유사도 평가 기법과 관련된 문제를 생각해 보고, 기술적 관점에서 딥러닝 모델이 소프트웨어 감정에 활용되기 위해 필요한 개선 방안에 대해 논의해 본다. 본 연구는 딥러닝 기반 소프트웨어 유사도 감정이 산업적으로 효율적이면서도 법적 분쟁을 최소화하기 위한 실질적인 방안을 모색해 보는데 그 목적이 있다. 개선 방안으로 하이브리드 접근법 도입, 데이터셋 증강 및 레이블링 자동화, 모델 경량화 및 효율적 학습, 설명 가능 AI (XAI) 도입 등과 함께 신뢰성 향상을 위해 다양한 평가 지표의 활용 시나리오를 제시한다.

Abstract

Recentlry, the application of deep learning to assess source code similarity has been an area of interest. Although this approach is very promising in that it can improve the accuracy and efficiency of similarity detection tasks by leveraging the ability of deep learning models to automatically learn and extract features from various representations of source code, similarity assessment using deep learning has issues regarding the accuracy, fairness, and interpretability of the results. This paper considers the issues related to deep learning-based software similarity assessment techniques and discusses the improvements required for deep learning models to be used in software assessment from a technical perspective. This study aims to provide a practical method for deep learning-based software similarity assessment to be industrially efficient while minimizing legal disputes. Suggested improvements include introducing a hybrid approach, automating dataset augmentation and labeling, lightweighting and efficient learning of models, and introducing explainable AI. In addition, we present scenarios for utilizing various evaluation indicators to improve reliability.

한글키워드 : 딥러닝, 인공지능, 유사도 감정, 소스코드 유사성

keywords : Deep learning, artificial intelligence, similarity assessment, source code similarity

* 숙명여자대학교 기초공학부
† 교신저자: 김유경(email: ykim.be@sookmyung.ac.kr)

접수일자: 2024.11.12. 심사완료: 2024.12.02.
게재확정: 2024.12.20.

1. 서 론

소프트웨어 유사성 평가는 표절 방지, 코드 최적화, 품질 평가, 리팩토링 등의 분야에서 중요한 역할을 하고 있다. 다양한 분야에서 AI 기술의 사용이 급증함에 따라 소프트웨어의 유사성 탐지 및 평가 과정에서도 딥러닝(Deep Learning)과 자연어 처리(NLP) 모델이 활용되고 있다. 딥러닝의 발전으로 기존의 전통적 유사도 평가 방식을 넘어 문법 및 의미적 유사성 평가가 가능해진 것이다. 소스 코드 유사성을 평가하는 데 딥러닝을 적용하는 것은 최근 몇 년 동안 관심 분야로 떠오르고 있다. 이러한 접근 방식은 딥러닝 모델이 소스 코드의 다양한 표현에서 자동으로 학습하고 기능을 추출하는 능력을 활용하여 유사성 탐지 작업의 정확도와 효율성을 개선할 수 있다는 측면에서 매우 긍정적이기는 하지만, AI를 활용한 유사도 감정은 결과의 정확성, 공정성, 해석 가능성에 대한 문제를 안고 있다. 소프트웨어 감정 결과는 기업에 법적, 재정적 영향을 미칠 수 있으므로 이러한 단점을 해결하여 위험을 줄이고 AI 활용 감정 결과에 대한 신뢰를 높이는 것이 중요하다.

본 논문에서는 딥러닝 기반 소프트웨어 유사도 평가 기법과 관련된 문제를 생각해 보고, 기술적 관점에서 딥러닝 모델이 소프트웨어 감정에 활용되기 위해 필요한 개선 방안에 대해 논의해 본다. 딥러닝을 활용한 소프트웨어 유사도 감정이 높아진 정확성에도 불구하고 발생할 수 있는 문제를 분석하고, 딥러닝 모델이 학습하는 과정에서 발생하는 기술적 문제와 결과의 신뢰성 문제를 살펴본다. 본 연구는 딥러닝 기반 소프트웨어 유사도 감정이 산업적으로 효율적이면서도 법적 분쟁을 최소화하기 위한 실질적인 방법을 제시하는 데 그 목적이 있다. 개선 방안으로 하이브리드 접근법 도입, 데이터셋 증강 및 레이블링 자동화, 모델 경량화 및 효율적 학습, 설명 가능 AI (XAI) 도입 등과 함께 신뢰성 향상을 위한 다양한 평가 지표의 활용 시나리오를 제시한다.

2. 관련 연구

딥러닝은 머신러닝의 새로운 분야로, 데이터의 고수준 추상화를 모델링하여 인간의 의사결정 과정을 모방하는 신경망 개념을 사용한다[1]. 신경망의 각 노드 또는 뉴런은 입력 데이터 세트에서 패턴을 추출하는 함수로 작동한다. 일반적인 신경망은 세 개의 순차적인 층, 즉 입력 층, 은닉 층, 출력 층으로 구성되고, 다양한 기능적 특성의 필요에 따라 여러 개의 은닉 층이 포함될 수 있다. 신경망에는 심층 신경망(DNN), 순환 신경망(RNN/RtNN), 재귀 신경망(RvNN), 그래프 신경망(GNN), 합성곱 신경망(CNN) 등 다양한 유형이 있다.

딥러닝은 이런 다층 신경망을 활용하여 데이터를 학습하는 인공지능 기술이다. 특히 비정형 데이터(텍스트, 이미지, 코드 등)에서 뛰어난 성능을 보인다. 특히 최근 RNN의 한 종류로, LSTM (Long Short-Term Memory)은 장기 의존성을 학습하기 위해 설계된 신경망 구조로, 소스코드와 같은 시퀀스 데이터를 처리하는 데 사용되고 있다[2].

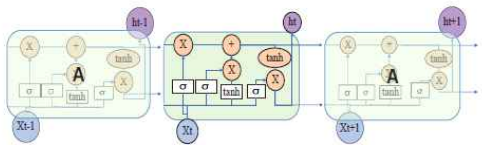

그림 1. LSTM 네트워크 구조 개략도[2]
Fig. 1. Schematic diagram of the LSTM network structure

Tree-LSTM은 트리 구조 데이터를 학습할 수 있도록 확장된 LSTM으로써 트리 구조 데이터에서 RNN을 학습하는 과제를 해결하기 위해 제안되었다. 표준 LSTM과 유사하게 Tree-LSTM의 각 셀 노드는 입력 게이트(it), 출력 게이트(ot), 셀 상태 선언(Ct) 및 숨겨진 출력(ht)으로 구성된다. 기존 AST(Abstract Syntax Tree) 기반 기법이 불필요한 노드가 포함되어 학습시간이 길어지고 과적합을 야기하던 문제를 해결하기 위해 도입된 Tree-LSTM은 AST와 같은 계층적 구조를 효율적으로 학습할 수 있으면서 코드의 의미적 유사성을 정교하게 탐색할 수 있다[3].

2022년 오픈AI가 ChatGPT를 공개한 이래로 생성형 모델 발전에 따라 프로그램 코드 분야에서는 이를 이용한 코드 생성, 다른 언어로의 번역, 코드 보정 등이 활발히 진행 중이다[4].

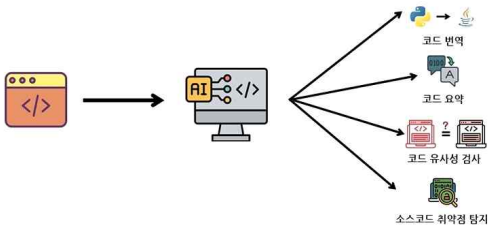

그림 2. 코드를 이해하는 언어모델을 활용한 주요 작업 (출처:카이스트 CSRC)
Fig. 2. Key work using language models that understand code (Source: KAIST CSRC)

딥러닝 기반 소스 코드 유사성 탐지 및 코드 클론 탐지 연구는 코드의 구문과 의미를 보다 정교하게 분석하기 위해 다양한 모델을 도입하여 발전하고 있다. 최근 연구에서는 Transformer와 같은 딥러닝 모델이 소스 코드 유사성 탐지에 효과적이라는 것이 입증되었다. 예를 들어, 코드와 자연어 설명을 같은 임베딩 공간으로 변환하여 유사성을 분석하는 CODEnn 모델은 자연어 질의에 기반한 코드 검색에서 높은 성능을 보여준다. 또한, Graph Neural Network (GNN) 기반 모델들은 코드의 그래프 표현을 벡터로 변환하여 코드의 구조와 의미를 보다 정교하게 분석할 수 있게 해준다. 대표적으로, Graph Matching Network를 사용하여 코드의 그래프 표현 간 유사성을 학습하는 방법이 사용된다[5][6].

대규모의 코드 데이터셋을 활용한 사전 학습(pre-training) 및 파인튜닝(fine-tuning)은 딥러닝 기반 코드 유사성 탐지 성능을 더욱 높이는 중요한 방법이다. CodeSearchNet 및 BigCloneBench 같은 대형 데이터셋은 코드 클론 탐지와 유사성 연구에서 널리 사용된다. 이들 데이터셋을 통해 대량의 코드 샘플을 학습한 모델은 높은 정확도와 재현율을 유지하며, 다양한 프로그래밍 언어와 코드 스타일에 대한 일반화 성능도 우수하게 나타나는 것으로 보고된다[7].

최근 딥러닝을 활용한 소스 코드 유사성 평가 연구는 Tree-LSTM, CNN 및 그래프 신경망과 같은 다양한 딥러닝 모델을 활용하여 코드의 의미적, 기능적 유사성을 포착하는데 초점을 두고 있다. 이들 연구 결과를 통해 딥러닝 기반 방법이 전통적인 기술보다 일관되게 뛰어난 정확도와 효율성, 그리고 다양한 프로그래밍 언어 및 플랫폼에 대한 적용 가능성을 보이고 있다.

딥 그래프 매칭을 통한 의미 기반 코드 검색 향상에 대한 연구에서는 쿼리와 코드 조각을 그래프 형식으로 변환하여 의미적 매칭을 촉진하는 신경 그래프 매칭을 사용하는 모델을 제안하였다[8]. 딥러닝 기반의 코드 기능 유사도를 탐지하는 방법으로 제안된 DeepSim은 코드 제어 흐름 및 데이터 흐름을 의미 행렬로 인코딩하여 코드 기능 유사도를 측정하는 것이다[9]. 유사 코드 세그먼트를 찾기 위한 딥 뉴럴 네트워크 기반 접근법으로 [10]에서 제안한 방법은 코드 클론을 식별하기 위해 동일한 두 개의 CNN을 사용하고 있

다. 이 모델은 코드 조각을 AST로 나타내고, CNN 기반 서브 네트워크를 사용해 이러한 AST에서 특징 벡터를 추출하도록 한다. 또 다른 연구로는 [11]과 같이 소스 코드 기능적 유사성 감지를 위한 그래프 기반 코드 표현 학습이 있다. 이 연구에서는 Tailor 모델을 제안하였으며, 이 모델은 기능적 유사성 감지를 위해 코드 속성 그래프기반 신경망(CPGNN)을 통해 그래프의 표현을 학습하도록 정의하였다.

이와 같이 코드 유사성 평가에 딥러닝 적용에 대한 연구가 활발하게 진행중이다. 이러한 방법들은 정확도와 일반화 성능을 높이며 특히 기능적으로는 동일하지만 문법적으로는 다르게 표현된 복잡한 코드 유사성을 탐지하는데 유용하게 활용 될 수 있을 것으로 보인다.

3. 소스코드 유사성 평가에서 딥러닝 적용의 문제

딥러닝 기반의 소스 코드 유사성 평가 방법들이 높은 정확도와 재현율을 보인다고 해도, 기본적으로 딥러닝 기술을 완전하게 신뢰하기 어려운 몇 가지 이유가 있다. 소프트웨어 유사도 감정에서 딥러닝를 활용하는 경우, 딥러닝에 내재된 기술적 문제들은 감정 결과에 영향을 끼치게 된다.

(1) 데이터셋 문제

레이블링 된 데이터의 부족이나 불균형한 데이터, 프로그래밍 언어 종속적인 특징과 관련된 문제이다. 딥러닝 모델을 훈련하려면 레이블링된 대규모 데이터셋이 필요하다. 하지만 의미적으로 유사한 코드를 수동으로 분류하는 것은 시간과 비용이 많이 들 뿐만 아니라, BigCloneBench와 같은 기존 데이터셋은 특정 유형의 코드에 편향될 수 있어 일반화가 어렵다. 특히 의미적으로 유사한 코드를 수집하는 것은 더욱 어렵기 때문에, 학습하기 어렵고 성능이 저하될 수 있다. 또한 대부분의 데이터셋은 특정 프로그래밍 언어에 종속되기 때문에, 다른 종류의 프로그래밍 언어나 멀티 프로젝트 상황에서는 모델 성능이 특히 낮아 질 수 있다.

(2) 코드 표현 및 특징 추출의 문제

AST나 CFG는 코드 구조를 학습하는 데 유용하지만, 실제 소스코드의 의미적 관계를 온전히 표현하지 못한다[12].

복잡한 코드에서는 AST와 CFG의 노드 수가 기하급수적으로 증가해 모델의 학습 시간이 길어지게 된다. 또한 문법적으로 다른 코드라도 기능적으로 동일할 수 있다. 이러한 의미적 유사성은 딥러닝 모델이 학습하기 어려운 영역이다. 단순한 토큰 기반 접근법이나 구조적 접근법으로는 해결이 어렵고, 심화된 의미 추출이 필요하다.

(3) 모델 학습 및 과적합 문제

소스코드 데이터셋은 다양한 코드 작성 스타일과 구조를 포함하게 된다. 모델이 훈련 데이터의 패턴에 과적합되면 새로운 코드에 대한 일반화 성능이 저하된다. 특히 데이터셋이 부족하거나 특정 언어에 편향된 경우 과적합이 더욱 심화될 수 있다. AST나 CFG 기반 모델은 데이터 구조가 복잡해질수록 연산량이 급격히 증가하는데 반해, 최근 많이 활용되고 있는 Tree-LSTM이나 GNN 같은 고급 딥러닝 모델은 학습 및 추론에 높은 계산 비용과 메모리 자원을 요구한다.

뿐만 아니라 동일하거나 매우 유사한 코드가 데이터셋에 포함될 경우, 모델이 이를 단순하게 학습하고 진짜 의미적 차이를 구분하지 못할 수 있게 되는데, 이것은 유사도 감정에서는 특히 문제가 될 수 있다.

(4) 모델 해석 가능성

CNN, LSTM, Transformer와 같은 딥러닝 모델들은 블랙박스처럼 동작하기 때문에, 유사도 평가 결과를 해석하고 설명하기 어렵다. 특히 코

드 유사성 감정에서는 결과의 신뢰성이 중요한데, 모델이 어떤 기준으로 유사성을 판단했는지 설명할 수 없다면 이는 신뢰하기 어렵다.

(5) 다중 언어 및 환경 지원 문제

현재 소프트웨어 개발 환경에서는 여러 프로그래밍 언어가 함께 사용되는 경향이 있다. 하지만 대부분의 딥러닝 모델은 단일 언어만 지원하고 있다. 다중 언어 코드나 크로스 프로젝트 상황에서 모델을 적용하려면 추가적인 훈련과 확장이 필요하다[13].

이와 같은 딥러닝의 기술적 문제로 인해, 소프트웨어 유사도 감정에서 딥러닝을 활용할 경우 감정 결과에 대한 법적 재정적 위험 요인을 반드시 고려해야 한다.

4. 소프트웨어 감정에서 딥러닝 활용을 위한 기술적 개선 방안

정확성이 담보 되지 않는 상황에서 딥러닝 기술이 사용되기 위한 당위성이 요구될 수밖에 없는데, 여타 다른 기술들과 같이 이러한 당위성을 도출할 수 있는 일차적 근거는 바로 기술적 효율성(efficiency)이다. 딥러닝 기술이 새로이 도입되는 다소 미성숙한 단계에서는 절대적 다수의 신뢰를 이끌어낼 수 있는 기술적 완성도를 확보하기는 어렵기 때문에, 이러한 고도의 효율성은 더욱더 강조될 수밖에 없다[14]. 그러나 소프트웨어 감정에서는 효율성만을 강조할 수 없으므로, 소프트웨어 유사도 감정에 딥러닝 기술을 적용하기 위해서는 정확성과 신뢰성을 향상 시킬 수 있는 기술적 개선이 필요하다.

4.1 기술적 개선 방안

(1) 하이브리드 접근법 도입

유사도 감정에서는 코드에 드러난 유사성 뿐만 아니라, 문법적으로 다르지만 기능적으로 동일한 코드들도 탐지해야 한다. CNN이나 RNN과 같은 기존 딥러닝 모델은 문법적 특징은 잘 학습하지만, 코드의 의미를 파악하는데 한계가 있다. 또한 소스코드의 구조를 표현한기 위해 사용하는 AST는 문법적 구조만 표현하고, 제어 흐름이나 데이터 흐름을 명확히 표현하지 못한다. CFG는 제어 흐름을 나타내지만 코드의 의미를 정량화하기에는 부족하다. CodeBERT, GraphCodeBERT 같은 사전 훈련 모델을 사용하면 코드의 의미적 표현을 더 효과적으로 학습할 수 있으므로, AST와 CFG를 결합해서 복합적 특징을 학습 할 수 있는 접근방법을 생각해 볼 수 있다. CodeBERT와 GraphCodeBERT는 코드의 문법적, 의미적 정보를 학습한 사전 학습 모델이다. CodeBERT는 BERT(Bidirectional Encoder Representations from Transformers) 모델을 기반으로 자연어와 소스코드를 함께 학습하는 사전 훈련 모델이다[15]. 단일 언어 코드(Python, Java 등)와 자연어 주석 간의 관계를 학습하여 코드의 의미적 표현을 효과적으로 학습한다. GraphCodeBERT는 CodeBERT를 확장한 모델로, 코드의 문법적 구조와 그래프 표현을 함께 학습한다. GNN의 개념을 도입해 코드의 AST와 같은 구문 구조 정보를 추가로 활용한다. 코드의 문법적 및 의미적 관계를 포괄적으로 평가하는 코드 유사성 탐지와 데이터 흐름 분석을 통해 코드의 오류를 탐지하는 버그 탐지 및 코드 분석에 주로 활용된다[16]. CodeBERT보다 구조적 정보(AST, 데이터 흐름)를 추가 학습하므로 코드의 의미적 유사성을 더 정확하게 평가할 수 있으므로, 이러한 모델들을 활용하면 코드 유사성 탐지의 일반화 성능도 높일 수 있을 것이다.

표 1. 사전훈련 모델 비교
Table 1. Comparison of pre-trained models

CodeBERT	GraphCodeBERT
BERT 기반, 코드와 주석 동시 학습	코드 구조와 의미 병합 학습
자연어와 코드의 의미적 관계 학습	데이터 흐름과 AST를 통한 구조 학습
코드 검색, 유사성 탐지	버그 탐지, 코드 분석

(2) 데이터셋 증강 및 레이블링 자동화

데이터 증강 기법을 활용해 다양한 유형의 코드 클론을 생성할 수 있다. 예를 들면, 변수명 변경, 코드 블록 재배열, 로직 변환 등을 통해 의미적으로 유사한 데이터를 자동 생성할 수 있다. 데이터 증강은 기존 데이터셋에서 새로운 데이터를 생성하여 훈련 데이터의 양과 다양성을 늘리는 방법이다. 소스코드 데이터에 적용하는 경우, 의미를 유지하면서 코드 형태를 변형하여 새로운 코드를 생성한다. 데이터 증강을 사용하면 희소한 데이터를 늘릴 수 있으므로 데이터 불균형을 해소할 수 있다. 이를 통해 다양한 형태의 코드를 학습해 모델이 새로운 코드에서도 잘 작동하도록 일반화 성능을 향상 시키고 과적합 문제를 해결할 수 있다[17].

이와 함께 자동 레이블링 도구를 도입한다면, 코드 실행 결과를 기반으로 의미적으로 동일한 코드를 판단하여 자동으로 라벨링하는 방법이 가능해 질 것으로 보인다. 레이블링 자동화는 대규모 소스코드 데이터셋을 자동으로 라벨링하는 기법이다. 수작업 레이블링은 비용이 많이 들고 시간이 오래 걸리므로, 자동화된 도구와 기법이 필요하다. 레이블링 시간을 단축해 대규모 학습 데이터셋을 신속하게 구축할 수 있다. 수작업 레이블링에서 발생할 수 있는 주관적 오류를 크게 줄일 수 있고, 실행 결과나 AST 기반 라벨링을 통해 코드의 의미적 관계를 더 정확하게 반영할 수 있다[18]. 또한 크로스 프로젝트 학습을 위해 다양한 소스코드 데이터셋에 적용하여, 일반화 성능을 개선하는 것도 필요하다.

(3) 모델 경량화 및 효율적 학습

모델 경량화를 통해 계산 비용을 줄이고 학습 시간을 단축하는 방안이 마련되어야 한다. 예를 들면, Tree-LSTM 대신 Transformer 모델을 최적화해 트리 구조 학습을 좀 더 효율화할 수도 있을 것이다. Attention 메커니즘을 활용해 코드의 중요한 부분만 집중적으로 학습시키는 방안도 고려해 볼 수 있다.

(4) 설명 가능 AI (XAI) 도입

모델이 어떤 기준으로 코드가 유사하다고 판단했는지 시각화 및 해석 도구를 도입하는 것이 필요하다. 예를 들면, Attention Map 시각화, 코드 트리 노드의 중요도 분석 등을 통해 판단 근거를 제시한다면, 모델의 신뢰성을 높이고 오류를 검증할 수 있을 것이다.

4.2 딥러닝 모델에 대한 평가 지표 활용

소스코드의 유사성을 평가하기 위해 딥러닝을 적용할 때 모델 성능과 코드 간 유사도를 측정하기 위해 다양한 평가 지표를 사용할 수 있다. 이 평가 지표는 유사성 탐지 모델의 정확성과 효과를 평가하는 중요한 요소이다. 특히 의미적으로 유사한 정도를 파악하기 위한 딥러닝 모델 성능 평가는 단순한 문법적 일치가 아닌 기능적 의미가 동일한지를 평가하는 것이 중요하다. 이를 위해 유사도 평가 메트릭과 모델 성능 평가 메트릭을 활용하여 평가 결과의 신뢰도를 증가시킬 수 있다.

기술적 문제를 완화할 수 있는 방안으로서 평가 지표의 활용은 매우 의미가 있다. 신뢰성 있는 평가 메트릭을 사용하면 모델이 실제 현장에

서 정확하고 일관된 결과를 제공한다는 것을 보장할 수 있다. 특히 실행 기반 유사도 메트릭은 두 코드가 논리적으로 동일한 기능을 수행하는지를 검정하기 때문에 현업 적용에는 필수적이다.

또한 성능 평가 메트릭은 여러 모델이나 방법론을 비교하고 모델을 지속적으로 개선하기 위한 객관적 기준을 제공한다. 따라서 최적의 모델 선택이나 개선방향을 도출할 수 있을 것이다. 이와 함께 불균형 데이터셋에서도 객관적이고 신뢰성 있는 평가를 제공할 수 있기 때문에, 다양한 메트릭을 조합하여 사용함으로써 모델 성능을 다각도에서 검증하고 유사도 평가 결과에 신뢰성과 객관성을 부여하는 것은 매우 중요하다.

- 코사인 유사도(Cosine Similarity)는 임베딩 벡터 간 각도를 기반을 유사도를 측정한다. GNN, Tree-LSTM 등 딥러닝 모델이 생성한 임베딩 벡터를 비교하기 위해 주로 사용한다.
- BLEU(Bilingual Evaluation Udeerstudy)는 본역 모델 성능 평가에 사용되던 BLEU 점수를 코드 유사성 평가에 적용하는 것이다. 코드 생성 모델이나 의미적 유사성을 평가할 때, 참조 코드와 모델이 예측한 코드 간 유사도를 평가한다.
- 코드 실행 기반 유사도(Execution-based similarity)는 소스코드의 실행 결과를 비교해 유사성을 평가하는 방법으로 코드의 의미적 유사성을 가장 정확하게 평가할 수 있다고 알려져 있다.
- F1 스코어는 정밀도와 재현율의 조화 평균으로, 두 지표의 균형을 평가한다. 불균형 데이터셋에서도 유용한 메트릭이다.
- ROC-AUC는 모델의 임계값 변화에 따른 예측 성능을 종합적으로 평가하는 메트릭으로써 유사성 탐지에서 모델의 분류 능력을 평가하는데 매우 유용하다.

표 2. 메트릭 사용 시나리오
Table 2. Metric usage scenarios

평가 목표	사용 가능 메트릭
코드 유사도 정량화	• 코사인 유사도 • 유클리드 거리
의미적 유사성 정확도 평가	• BLEU 점수 • 코드 실행 기반 유사도
모델의 분류 성능 평가	• 정확도, 정밀도 • 재현율, F1 스코어 • ROC-AUC

5. 결 론

딥러닝은 기능 추출을 자동화하고 다양한 코드 표현에서 유사성 탐지의 정확도를 개선함으로써 소스 코드 유사성 평가 분야에 기여하고 있다. 향상된 AST, 그래프 기반 표현과 같은 기술은 구문적 및 기능적 유사성을 모두 탐지하는 데 상당한 개선을 보이고 있다. 또한 딥러닝 모델은 바이너리 코드 유사성 분석 및 코드 검색 작업에서 효과적인 것으로 입증되었으며, 다양한 소프트웨어 엔지니어링 과제를 처리하는 데 있어 매우 유용하다. 즉, 딥러닝 기법을 통해 코드의 의미적 유사성을 보다 정확하게 파악하는 가능성을 확인할 수 있었고, 전통적인 문법적 비교에 비해 코드의 구조와 동작을 이해하는 데 큰 잠재력을 보였다.

본 논문에서는 딥러닝 기반 소프트웨어 유사도 평가 기법과 관련된 문제를 생각해 보고, 기술적 관점에서 딥러닝 모델이 소프트웨어 감정에 활용되기 위해 필요한 개선 방안에 대해 논의해 보았다.

앞으로 다양한 코드 표현 방법, 다중 언어 코드 유사성 평가 모델 개발, 설명가능한 모델 개

발, 효율성 개선, 그리고 대규모 공개 코드 데이터셋 구축과 같은 연구가 지속적으로 이루어져야 할 것이다. AST, CFG 등의 그래프 표현을 더욱 효율적으로 활용하거나, 코드와 주석, 문서화를 결합하여 더욱 풍부한 의미적 정보를 학습할 수 있는 모델 개발이 필요하다. 또한 여러 프로그래밍 언어 간의 유사성을 평가할 수 있는 다중 언어 학습 모델을 개발하여 코드 유사성을 언어 독립적으로 평가하는 연구로의 확장이 이루어져야 한다. 딥러닝 모델이 산출한 유사성 평가 결과에 대해 인간이 이해할 수 있는 설명을 제공하여 모델의 신뢰성을 높이는 연구와 함께 코드 유사성 평가의 정확도를 높이면서도 연산 비용을 줄일 수 있는 경량 모델이나 최적화 기법이 개발되어야 할 것이다. 특히 대규모 소스코드를 대상으로 하는 산업 현장에 적용할 수 있도록 모델을 경량화하는 연구가 필요하다. 마지막으로 다양한 언어와 도메인을 아우르는 대규모 코드 유사성 데이터셋이 부족하므로, 실효성 있는 데이터셋을 구축하고 공개하는 작업이 중요하다.

이와 같은 딥러닝을 통해 소스코드 유사성 평가의 정확도와 효율성을 높이기 위한 연구로의 확장이 코드 분석과 이해에 있어 인공지능의 역할을 확대하는 중요한 기초가 될 것으로 기대한다.

참 고 문 헌

[1] I. Goodfellow, Y. Bengio, A. Courville. Deep Learning. MIT Press(2016). ISBN: 0262035618

[2] X. Wu, W. Zhao, Z. Tan, X. Zhang, W. Chen, “Research and Implementation of Code Similarity Detection Technology Based on Deep Learning”, In Proceedings of International Conference on Cloud Computing and Intelligent Systems (CCIS), pp.235-239, 2023. DOI: https://doi.org/10.1109/CCIS59572.2023.10262836

[3] K. S. Tai, R. Socher, C. D. Manning, “Improved semantic representations from tree-structured long short-term memory networks”, In Proceedings of the international joint conference on Natural Language Processing, pp.1556-1566, 2015. DOI: https://doi.org/10.3115/v1/P15-1150

[4] S. Park, “AI language model that understands source code and detects vulnerabilities”, Security News Articles, avilable from https://m.boannews.com/html/detail.html?idx=130528

[5] V. Suttichaya, N. Eakvorachai, T. Lurkraisit, et al., “Source Code Plagiarism Detection Based on Abstract Syntax Tree Fingerprintings”, In Proceedings of International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), 2022. DOI: https://doi.org/10.1109/iSAI-NLP56921.2022.9960266

[6] S. Yang, L. Cheng, Y. Zeng, Z. Lang, H. Zhu, Z. Shi, “Asteria: Deep Learning-based AST-Encoding for Cross-platform Binary Code Similarity Detection”, In Proceedings of the International Conference on Dependable Systems and Networks, pp. 224-236, 2021. https://doi.org/10.1109/DSN48987.2021.00036

[7] M. Chilowicz, E. Duris, G. Roussel, et al., “Syntax tree fingerprinting for source code similarity detection”, In Proceedings of the International Conference on Program Comprehension, 2009. https://doi.org/10.1109/ICPC.2009.5090050

[8] N. Bibi, A. Maqbool, T. Rana, et al., “Enhancing Semantic Code Search With Deep Graph Matching”, IEEE Access, Vol. 11, pp.52392 - 52411, 2023. DOI:

https://doi.org/10.1109/ACCESS.2023.3263878

[9] G. Zhao, J. Huang, “DeepSim: deep learning code functional similarity”, In Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference, pp.141-151, 2018. DOI: https://doi.org/10.1145/3236024.3236068

[10] D. K. Kim, “A Deep Neural Network-Based Approach to Finding Similar Code Segments”, IEICE Transactions on Information and Systems, Vol. 103, no. 4, pp.874-878, 2020. DOI: https://doi.org/10.1587/transinf.2019edl8195

[11] J. Liu, J. Zeng, X. Wang, Z. Liang, “Learning Graph-based Code Representations for Source-level Functional Similarity Detection”, In Proceedings of the International Conference on Software Engineering, 2023. DOI: https://doi.org/10.1109/ICSE48619.2023.00040

[12] J. Zhang et al., “A novel neural source code representation based on abstract syntax tree”, in Proceedings of International Conference on Software Engineering, 2019. DOI: https://doi.org/10.1109/ICSE.2019.00086

[13] T. Vislavski et al., “LICCA: A tool for cross-language clone detection”, In Porceedings of the International conference on Software Analysis, Evolution and Reengineering (SANER), 2018. DOI: https://doi.org/10.1109/SANER.2018.8330250

[14] G. Nam, “Factors to be Considered for the Introduction of Artificial Intelligence Technology into Field of Legal Practice- with the Case of Similarity Assessment of Trademarks”, Korea University Legal Research Institute, Korea Law, Vol. 105, pp.129-159, 2022. DOI: https://dx.doi.org/10.36532/kulri.2022.105.129

[15] Z. Feng et al., “CodeBERT: A pre-trained model for programming and natural languages”, Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 1536-1547, 2020. DOI: https://doi.org/10.18653/v1/2020.findings-emnlp.139

[16] D. Guo et al., “GraphCodeBERT: Pre-training code representations with data flow”, in Proceedings of International Conference on Learning Representations (ICLR), 2021. DOI: https://doi.org/10.48550/arXiv.2009.08366

[17] ‘what is Data Augmentation?’, available at https://aws.amazon.com/ko/what-is/data-augmentation/

[18] S. Jeffrey et al., “Towards a Big Data Curated Benchmark of Inter-Project Code Clones”, In Proceedings of International Conference on Software Maintenance and Evolution (ICSME), 2014. DOI: https://doi.org/10.1109/ICSME.2014.77

저 자 소 개

김유경(Yukyong Kim)

2005.9-2006.8 UC Davis, Post-doc.
2006.9-2013.9 한양대학교 컴퓨터공학과 연구교수
2018.3-현재 숙명여자대학교 기초공학부 교수
<주관심분야> 웹서비스 QoS 평가, SOA 기반 IoT 신뢰 평가, 소프트웨어 품질평가