KIPS Trans. Softw. and Data Eng. 온라인 저지 시스템 지원을 위한 Feature-Wise Linear Modulation 기반 소스코드 문맥 학습 모델 설계 473
Vol.11, No.11 pp.473~478 https://doi.org/10.3745/KTSDE.2022.11.11.473
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

Learning Source Code Context with Feature-Wise Linear Modulation to Support Online Judge System

Kyeong-Seok Hyun† · Woosung Choi†† · Jaehwa Chung†††

ABSTRACT

Evaluation learning based on code testing is becoming a popular solution in programming education via Online judge(OJ). In the recent past, many papers have been published on how to detect plagiarism through source code similarity analysis to support OJ. However, deep learning-based research to support automated tutoring is insufficient. In this paper, we propose Input & Output side FiLM models to predict whether the input code will pass or fail. By applying Feature-wise Linear Modulation(FiLM) technique to GRU, our model can learn combined information of Java byte codes and problem information that it tries to solve. On experimental design, a balanced sampling technique was applied to evenly distribute the data due to the occurrence of asymmetry in data collected by OJ. Among the proposed models, the Input Side FiLM model showed the highest performance of 73.63%. Based on result, it has been shown that students can check whether their codes will pass or fail before receiving the OJ evaluation which could provide basic feedback for improvements.

Keywords : Deep Learing, GRU, FiLM, Online Judge

온라인 저지 시스템 지원을 위한 Feature-Wise Linear Modulation 기반 소스코드 문맥 학습 모델 설계

현 경 석† · 최 우 성†† · 정 재 화†††

요 약

온라인 저지 시스템 지원하기 위한 표절 검사, 소스코드 분석 및 자동화된 튜터링 기법이 연구되고 있다. 최근 딥러닝 기술 기반의 소스코드 유사도 분석을 통한 표절 감지 기술들이 제안되었으나, 자동화된 튜터링을 지원하기 위한 딥러닝 기반의 연구는 미흡한 실정이다. 따라서 본 논문에서는 자바 바이트코드와 문제정보를 결합하여 학습하고, 학습자가 온라인 저지 시스템에 코드를 제출하기 전에 pass/fail 여부를 예측할 수 있는 GRU 기반의 Input / Output side FiLM 모델을 제안한다. 또한 온라인 저지에 수집되는 데이터의 특성상 비대칭이 발생하기 때문에 밸런스 샘플링 기법을 적용하여 데이터를 균등하게 분포시켜 두 상황을 제안한 모델로 학습하였다. 실험 결과 Input side FiLM 모델이 가장 높은 73.63%의 성능을 보였다. 이를 기반으로 학습자들이 온라인 저지의 평가를 받기 전에 pass/faill 여부를 확인하여 소스코드 개선에 대한 피드백 기능에 적용 가능할 것으로 예상된다.

키워드 : 딥러닝, GRU, FiLM, 온라인 저지

1. 서 론

온라인 저지(Online judge)는 학습자가 작성한 프로그램을 실행하여 검증 및 평가하는 시스템으로 프로그래밍 대회를 목적으로 개발되었다. 최근에는 학습자의 프로그래밍 수준에 따른 다양한 문제를 제공하고 평가하는 목적으로 많이 사용되며, 기업에서도 프로그래머 채용을 위해 온라인 저지 목적으로 leetcode.com[1], Baekjoon[2]과 같은 서비스가 사용되고 있다.

그러나 현재 대다수의 온라인 저지는 학습자들이 작성한 코드를 입력받아 컴파일 및 실행 후 사전에 지정된 테스트 케이스에 대해서 코드의 정확성을 검증한다는 점에서 결과 지향적인 목적에 초점이 맞추어져 있다. 따라서 학습자들이 온라인 저지에 제출하기 전 코딩 결과물을 만드는 과정에서 자신의 코드가 의미적 오류(semantic error) 없이 정상 작동할 것인지 피드백을 받을 수 없는 한계가 있다. 학습자가 프로그램을 제출하기 전에 유의미한 피드백 받을 수 있다면 흥미 유

※ 이 논문은 2019년도 한국방송통신대학교 학술연구비 지원을 받아 작성된 것임.
† 비 회 원 : 고려대학교 컴퓨터학과 석・박사통합과정
†† 준 회 원 : Sony Group Corporation 연구원
††† 종신회원 : 한국방송통신대학교 컴퓨터과학과 교수
Manuscript Received : July 5, 2022
First Revision : July 26, 2022
Accepted : August 9, 2022
* Corresponding Author : Jaehwa Chung(jaehwachung@knou.ac.kr)

발과 학습 효과를 높일 수 있으며 온라인 저지를 보다 프로그래밍 능력 향상에 적합하게 설계할 수 있다[3].

온라인 저지의 효율성을 향상하고자 코드 유사도 분석 기반 표절 검사 연구[4-6]가 이루어졌으며, 최근에는 딥러닝 기술 기반의 소스코드 분석을 통한 프로그램의 오류를 찾는 연구[7-13]들이 제시되고 있다. 평가자의 개입 없이 학습자의 코드를 분석하여 피드백을 제공하는 튜터링 자동화 기법으로는 코드에서 사용된 자료형에 대한 통계 기반의 피드백 제공을 하는 연구[3]가 있으나 딥러닝 기술을 적용한 자동 튜터링과 관련된 연구는 미흡한 상황이다.

따라서 온라인 저지의 자동 튜터링 지원 한계를 극복하기 위해 본 논문에서는 GRU 기반의 자바 바이트코드와 각 소스코드가 해결하고자 하는 문제정보를 Feature-wise Linear Modulation(FiLM)[14]을 통해 결합 학습하는 모델을 제안한다. 제안 모델을 통해 학습자가 온라인 저지에 코드를 제출하기 전 pass/fail 여부를 판단하는 일차적인 검증 도구로 피드백을 제공받을 수 있다. 또한, 다수의 학습자가 온라인 저지를 사용하는 환경에서 각 학습자가 소스코드를 제출할 때마다 온라인 저지가 평가하는데 소요되는 시간적, 물리적 비용을 절감할 수 있다. 본 논문의 공헌은 다음과 같이 요약된다.

- GRU 기반 소스코드 pass/fail 예측 모델을 제안함
- FiLM 기법을 통한 정보 결합 학습 모델을 제안함
- 실험을 통해 GRU 기반 베이스라인 모델 대비 FiLM 기법 및 밸런스 샘플링을 적용한 모델의 정확도가 3.07% 향상됨을 입증함

본 논문의 구성은 다음과 같다. 2장에서는 튜터링 자동화를 지원하기 위해 소스코드 유사도 분석 및 딥러닝 기반 구문 오류 검출 연구를 서술한다. 3장에서는 GRU 기반 소스코드의 pass/fail을 예측하는 모델을 제안한다. 4장에서는 본 논문에서 제안한 모델의 성능 평가를 진행한다. 5장에서는 연구 결과 정리 및 향후 연구 방향에 관해 서술한다.

2. 관련 연구

온라인 저지를 위해 머신러닝 기법을 활용한 소스코드 간의 유사도를 분석하는 연구들이 있었으며, 최근 딥러닝 기반의 언어모델을 활용한 소스코드의 에러검출 및 수정을 하는 연구가 제안되고 있다. 이번 장에서는 머신러닝 및 딥러닝 기반 모델에 대하여 서술한다.

2.1 머신러닝 기반 소스코드 유사도 분석

소스코드의 유사성을 분석하기 위하여 패턴 분석, 구문 트리 구조 분석을 중점으로 다양한 연구들이 제시되고 있다. 자바 바이트코드를 선형화한 후 지역정렬 알고리즘을 기반으로 두 바이트코드 사이의 유사구간 및 유사도를 계산하는 연구[4]가 있으며, ANTLR LL(k) 파서 생성기를 수정한 연구[5]는 파스 트리(V_t)를 생성하고 파스 트리로부터 생성할 수 있는 모든 서브 트리를 벡터화하여 두 프로그램 사이의 유사도를 계산한다. JPlag[6]의 경우 토큰 시퀀스를 이용하는 표절 검사 시스템으로 Greedy String Tiling 알고리즘을 이용하여 두 토큰 시퀀스에서 연속적으로 일치하는 구간을 찾아 유사도를 계산한다.

2.2 언어모델 기반 소스코드 오류 분석

DeepFix[7]는 프로그램의 구문 오류를 수정하는 연구로 GRU 기반의 어텐션(attention)을 적용하여 93개의 문제에 대하여 에러가 포함된 6971개의 C 소스코드 중 27%의 소스코드에 대하여 에러를 수정하였고 19%의 소스코드에 대하여 에러 수정이 부분적으로 이루어졌다. TRACER[8]의 경우 주어진 소스코드에 대해 새로운 토큰 시퀀스를 예측하는 방식으로 DeepFix와 동일한 데이터에 대하여 완전 수정율 44%, 부분 수정율 27%을 보였다. Neuro-Symbolic program corrector[9]의 경우 구문 오류가 없는 소스코드를 GRU로 학습한 후 에러가 있는 소스코드로부터 토큰 시퀀스를 예측하고 SynFix 알고리즘으로 구문 오류를 수정한다.

Bugram[10]은 소스 파일의 토큰으로부터 N-gram 모델을 사용하여 토큰 시퀀스들의 확률 분포를 학습하여 버그를 감지한다. Seq2seq 모델 기반의 연구 sk_p[11]는 에러가 포함된 소스코드를 수정하기 위한 확률 분포를 생성하여 9078개의 문제에 대해 제출된 파이썬 소스코드 중 오류가 있는 프로그램의 29%를 수정하였다. 그 외에 LSTM 기반 어텐션을 활용한 연구[12]가 변수 오용 버그를 해결하고자 제안되었다.

최근 의미적 오류를 감지하고자 추상 구문 트리를 학습시키기 위해 CNN을 변형한 연구[13]가 제안되었다. 이 연구에서는 약 29개의 문제에 대해 수집된 270,000개의 소스코드에 대하여 할당문, 조건문, 출력값 형식과 메모리 할당 관련 오류들을 지역화(localization)하는 방식으로 72%의 pass/fail 예측 정확도를 보였다.

그러나 지금까지의 연구는 주어진 소스코드만 분석하는 것에 그치며 해당 소스코드가 어떤 문제를 해결하는지에 대한 정보가 결여된 상태로 학습이 진행된다. 따라서 본 논문에서는 이러한 한계점에 착안하여 소스코드와 문제정보를 결합 학습하여 문제정보에 따른 문맥(context) 학습이 가능한 GRU 기반의 모델을 제안한다.

3. 제안 모델

이번 장에서는 자바 소스코드의 pass/fail 여부를 예측할 수 있는 GRU 기반 네트워크 모델을 제안한다. 또한 제안하는 모델의 예측 정확도를 향상하기 위해 소스코드와 각 소스코드가 해결하고자 하는 문제정보를 결합하는 FiLM 적용 모델을 제안한다.

3.1 베이스라인 모델

온라인 저지 시스템은 학습자가 작성한 소스코드를 수집하게 되며 제출된 소스코드에는 다양한 구문 오류와 의미적 오류가 존재한다. 구문 오류의 경우 컴파일 단계에서 감지할 수 있으나 의미적 오류의 유무는 온라인 저지가 사전에 정한 테스트 케이스에 대하여 검증 작업을 통해 소스코드의 pass/fail 여부를 판단할 수 있다. 본 연구에서는 제출된 자바 소스코드에 컴파일이 완료되어 구문 오류가 검증된 자바 바이트코드(bytecode)를 사용하여 의미적 오류에 따른 소스코드의 pass/fail을 예측하는 환경을 가정한다.

본 논문에서 제안하는 GRU 기반 베이스라인 모델은 Fig. 1과 같다. GRU는 update, reset gate를 사용하는 모델로 긴 시퀀스 데이터를 학습할 때 과거의 상태의 정보를 얼마나 반영할지 결정하여 기존 RNN의 short-term memory로 인한 기울기 소실(vanishing gradient) 문제를 해결하는 장점이 있다.

그림에서 베이스라인 모델은 여러 토큰으로 구성되는 자바 바이트코드 ($t_1, t_2, \ldots, t_n$) 입력 시, 각 1바이트의 토큰을 0 ~ 255 사이의 정수로 사상한다. 사상된 값은 임베딩 계층(embedding layer)을 통하여 128차원의 벡터($e_1, e_2, \ldots, e_n$)로 변환 후 GRU로 학습시켜 전체적인 바이트코드의 문맥 벡터(context vector)를 학습한다. 이후 문맥 벡터를 통하여 주어진 바이트코드가 온라인 저지의 검증에 통과하는지 예측한다.

제안 모델이 소스코드만을 학습하게 되는 경우, 특정 문제를 해결하고자 하는 소스코드의 문맥을 학습하는데 한계가 있다. 따라서 소스코드와 문제정보를 결합 학습하는 모델이 요구되며, 본 논문에서는 최근 이미지와 자연어를 결합 학습하는 연구 중 성능이 우수하고 다양한 문제에 적용할 수 있어 활용도가 높은 FiLM 기법을 사용한다.

3.2 FiLM 적용 모델

FiLM[14]은 신경망의 연산 과정에 조건 혹은 정보를 추가하기 위해 feature-wise 아핀변환을 하기 위한 계층을 추가하는 기법이다. 이미지와 그에 상응하는 질문을 각각 CNN과 LSTM으로 학습하여 특징 결합(feature concatenation)했을 때 52.3% 정답률을 보였으며, FiLM을 적용했을 때 질문에 따른 정답과 연관이 있는 이미지의 특징을 집중적으로 활용하여 예측한 결과 97.7% 정답률을 보였다. 이는 이미지와 자연어를 단순 결합하는 방법과 달리 FiLM 계층을 통하여 문제정보를 더욱 잘 반영할 수 있는 가중치 값을 학습하는 과정이 예측 결과 향상에 영향을 끼쳤을 것으로 분석된다. 따라서 본 논문에서도 단순 결합 혹은 아다마르 곱 연산을 통한 자바 바이트코드와 문제정보를 직접 결합하지 않고 FiLM을 적용하여 각 소스코드가 해결하고자 하는 문제의 문맥을 반영하여 pass/fail을 예측하는 모델을 설계한다. 본 논문에서 결합에 사용되는 문제정보는 최소한의 정보만으로 문제를 식별할 수 있는 양의 정수값인 문제의 번호를 사용한다. 문제정보 P_{id}는 $\{p_i \mid p_1, p_2, \ldots, p_n\}$로 정의한다.

신경망 상에 FiLM 적용 위치에 따른 성능 차이가 존재할 수 있어, GRU 학습 전 단계에 각 자바 바이트코드의 토큰에 FiLM을 적용하는 방식의 Input side FiLM 모델과, 자바 바이트코드를 GRU로 학습한 문맥 벡터에 FiLM을 적용하는 Output side FiLM 모델을 제시한다. Input side FiLM 모델의 경우 각 토큰에 대하여 문제정보를 결합하기 때문에, 동일한 토큰이 해결하고자 하는 문제가 상이할 때 다른 문맥을 담을 수 있도록 설계했다. Output side FiLM 모델의 경우 GRU를 통하여 전체적인 문맥을 학습한 후 문제정보를 결합하기 때문에 보다 코드의 문맥을 범용적으로 수용할 수 있다.

1) Input side FiLM 모델

Input side FiLM 모델은 문제의 정보를 GRU로 학습하기 전 단계에서 선행 결합하는 모델로 동일한 토큰이 문제가 다르면 다른 pi와 결합되어 학습되기 때문에 문맥 정보를 강하게 반영할 수 있다. Fig. 2와 같이 입력된 자바 바이트코드 토큰을 각각 128차원의 벡터로 임베딩한 후 pi와 결합한다. 결합 시 pi 또한 128차원으로 임베딩 하여 FiLM 가중치 값 (α, β)을 생성한다. 그 후 GRU의 학습 전 단계에서 각각의

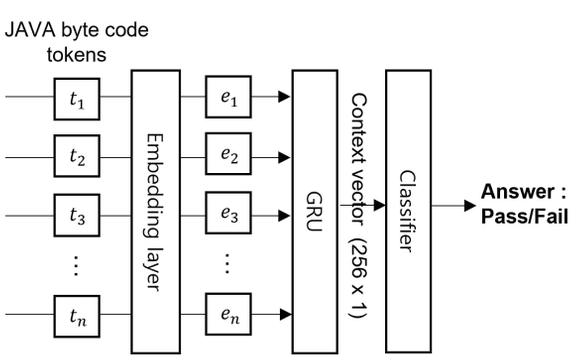

Fig. 1. Baseline Model

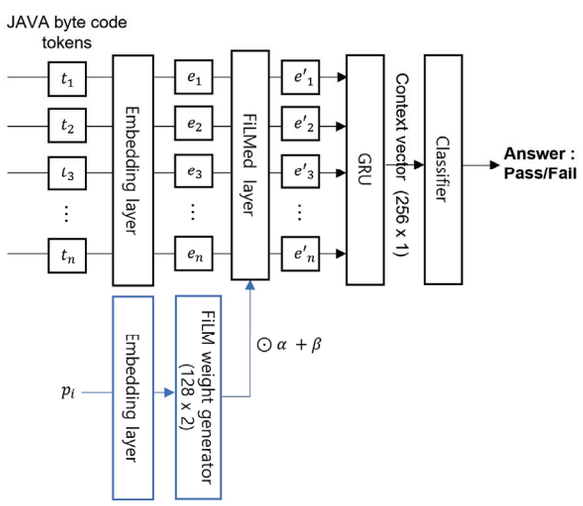

Fig. 2. Input Side FiLM Model

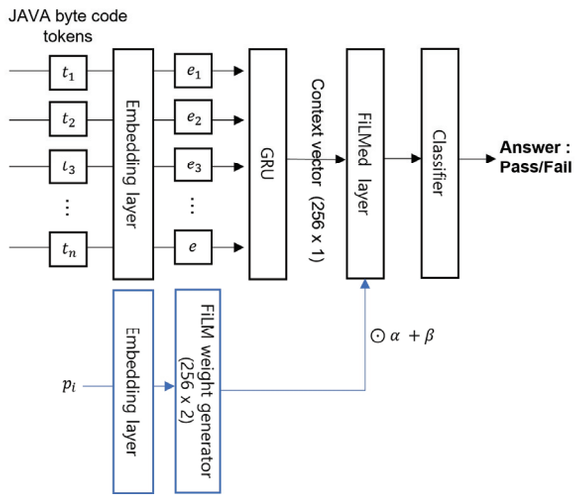

Fig. 3. Output Side FiLM Model

임베딩 토큰에 특징 변환(feature transformation) 연산 $\odot\alpha + \beta$을 적용한다.

이처럼 문제의 정보가 먼저 결합되어 GRU로 학습되는 경우 FiLM 계층이 pi로부터 생성한 가중치 값을 각 임베딩 토큰에 적용시켜 문맥 벡터가 공통된 pi를 갖는 토큰 셋에 의미가 강하게 반영된다. 하지만 해당 모델은 제출된 소스코드가 적을 경우 특정 문제에 대해 각 토큰의 연관성을 문맥 벡터에 반영하기 어려워지며 비대칭적인 데이터에 학습 능력은 다소 떨어질 것으로 예상된다.

2) Output side FiLM 모델

Output side FiLM 모델은 문제의 정보를 후속 결합하는 모델로 Fig. 3과 같이 자바 바이트코드의 토큰을 임베딩하여 GRU로 학습한 후 얻은 문맥 벡터와 문제의 p_i로부터 생성된 FiLM 가중치 값을 결합하여 최종적으로 해당 문제에 대하여 pass/fail을 예측하는 모델이다.

임베딩된 자바 바이트코드 토큰들을 GRU로 학습한 결과의 문맥 벡터에 FiLM 가중치 생성기(weight generator)가 생성한 α, β로 특징 변환 연산 $\odot\alpha + \beta$을 적용한다. 이 때 α, β는 연산을 위해 각각 문맥 벡터와 동일한 256 차원의 벡터로 생성한다. 문제의 정보가 부재한 상황에서 제출된 코드들을 GRU로 학습하여 범용적인 코드를 이해도가 높은 문맥 벡터를 얻을 수 있을 것이라 예상하며, 특정 문제에 대한 학습 데이터가 적어도 코드에 대한 범용적 이해도가 높아 Input side FiLM 모델보다 성능이 우수할 것으로 예상된다.

4. 실 험

본 논문에서는 3장에서 제안한 세 모델의 유효성을 증명하기 위해 실데이터를 기반으로 실험한다. 사용된 데이터는 프로그래밍 교육과정에서 사용된 47개의 문제에 대하여 제출된 13,827개의 자바 코드로 모델을 학습시켰다.

4.1 학습 데이터 및 시스템

모델 학습에 사용된 데이터는 Table 1과 같다. 데이터의 특성상 문제의 난이도와 시간 누적에 따른 제출 빈도 차이로 인하여 47개의 문제 중 가장 많은 코드가 제출된 문제는 932개, 가장 적은 코드가 제출된 문제는 153개로 데이터 비대칭이 존재한다. 또한 전체 문제에 대하여 정답률은 41% 정도이지만 가장 오답률이 높은 문제의 경우 92.72%, 가장 낮은 문제의 경우 22.7%로 문제별 오답률의 차이가 큰 특성이 있다.

온라인 저지에서 수집되는 데이터는 문제의 난이도가 낮은 문제는 제출 빈도가 높고, 난이도가 증가함에 따라 제출 빈도 및 정답률이 낮아 데이터가 비대칭적인 특성을 갖는다. 또한, 문제가 등록된 후 시간 경과에 따른 제출로 누적되는 코드로 인하여 새로 등록된 문제와 과거에 등록된 문제 간의 데이터 비대칭도 발생한다.

분류 문제에서의 데이터 비대칭은 소스코드가 적은 문제에 대하여 모델이 충분한 학습이 이루어지지 않으면 데이터 표현에 있어서 상대적으로 소스코드가 많은 문제에 대한 편향적 학습이 이루어지고, 이는 모델의 예측성능 저하로 이어질 수 있다. 따라서 본 논문에서는 이를 해결하기 위해 밸런스 샘플링(balanced sampling) 기법을 적용하였다. 총 n개의 문제 p_1, p_2, ..., p_n에 대해 문제 p_i에 속한 소스코드가 샘플링될 확률을 $(1/n) \times (1/|p_i|)$로, 각 문제 집단에 속한 소스코드가 샘플링될 확률을 동일하게 설정하였다. 밸런스 샘플링 기법을 적용한 결과, 소스코드가 가장 많이 제출된 문제의 경우 255 ~ 348개의 문제를 학습하도록 언더샘플링(undersampling)이 이루어졌다. 반면 제출된 소크코드가 가장 적은 문제의 경우 중복을 허용하여 253 ~ 336개의 문제를 학습하도록 오버샘플링(oversampling)이 이루어졌다.

본 논문에서는 데이터의 비대칭적인 상황과 분포를 균등화시킨 두 상황에 대하여 Input side FiLM 모델과 Output side FiLM 모델이 설계의 의도에 맞게 학습이 되었고 어떠한 성능을 내는지 비교분석을 하였다.

실험에 사용된 시스템은 Intel(r) Xeon(R) CPU E5-2630 v4 @ 2.20GHz 40 core, GeForce RTX 2080 Ti 126GB X 4로 구성된다. 학습에 사용된 옵티마이저는 adam이며, 학습율(learning rate)의 경우 0.00001부터 0.0001 구간에 대해서 실험을 진행하면서 optuna를 활용하여 하이퍼 파라미터를 조정하였다.

Table 1. Experimental data

	pass	fail	min items	max items	min fail(%)	max fail(%)	total items
Train data	5,693	8,124	153	932	22.7%	92.72%	13,827
Validation data	723	1005	16	109	14.81%	89.7%	1,728
Test data	710	1019	13	113	15.38%	93.75%	1,729

4.2 성능 평가

트레이닝 셋에 대하여 각 모델을 학습시켰을 때 Table 2와 같이 Base, Input side FiLM, Output side FiLM 모델 모두 밸런스 샘플링 기법을 사용한 경우 비대칭 데이터를 학습한 것보다 손실(loss)이 낮고 정확도(accuracy)이 높게 측정이 되었다. 트레이닝 셋에 대한 학습 결과로는 베이스라인 모델의 손실이 0.1665로 가장 낮게, 정확도가 93.35%로 가장 높게 측정되었다. Input side FiLM 모델과 Output side FiLM 모델을 비교했을 경우 트레이닝 셋에 대해서 밸런스 샘플링 기법을 적용한 데이터를 학습시킨 Output side FiLM 모델이 더 낮은 손실인 0.2384와 더 높은 정확도 88.81%를 보였다.

반면 테스트 셋의 경우 Input side FiLM 모델의 경우 비대칭 데이터를 학습했을 때의 손실이 0.609로 균형된 데이터를 학습했을 때의 손실인 0.7385보다 낮게 측정되었다. 정확도의 경우 비대칭 데이터를 학습했을 때와 균형된 데이터를 학습했을 때 73.63%로 동일하게 측정되었다.

Output side FiLM 모델의 경우 테스트 셋에 대하여 균형된 데이터를 학습했을 때의 손실이 0.7774로 비대칭 데이터를 학습했을 때의 손실인 0.8036보다 낮게 측정되었다. 하지만 비대칭 데이터를 학습했을 때 정확도가 72.18%로 균형적인 데이터를 학습할 때의 정확도 71.78%보다 높은 성능을 보였다.

최종적으로 테스트 셋에 대하여 Input side FiLM 모델이 비대칭 데이터를 학습했을 때 손실이 0.609로 제일 낮고, 밸런스 샘플링 기법을 적용한 데이터와 적용하지 않은 데이터에 대해 동일한 정확도 73.63%를 보였다.

실험 설계 시 학습 과정에서 30 steps 동안 기존의 정확도보다 높아지지 않는다면 조기 종료되도록 설정하였으며 Fig. 4와 같이 Output side FiLM 모델은 baltrue, balfalse (balanced sampling 사용 여부)에 상관없이 학습 속도가 Input side FiLM보다 빠르지만, 학습이 조기 종료되는 것을 볼 수 있다. 이는 문제정보가 GRU로 학습된 문맥 벡터와 결합되기 때문에 Input side FiLM 모델과 같이 각 토큰이 문제정보와 결합되는 구조보다 FiLM 가중치 생성기의 영향을 직접적으로 받기 때문에 학습이 빠르나 포화상태가 발생하는 것으로 추정된다.

Table 2. Experimental results

	balanced sampling	train/loss	train/acc	test/loss	test/acc
Base model (GRU only)	False	0.2748	86.16%	0.7231	71.02%
	True	0.1665	93.35%	0.7783	70.56%
Input side FiLM	False	0.2643	87.81%	0.609	73.63%
	True	0.2433	88.36%	0.7385	73.63%
Output side FiLM	False	0.2529	87.92%	0.8036	72.18%
	True	0.2384	88.81%	0.7774	71.78%

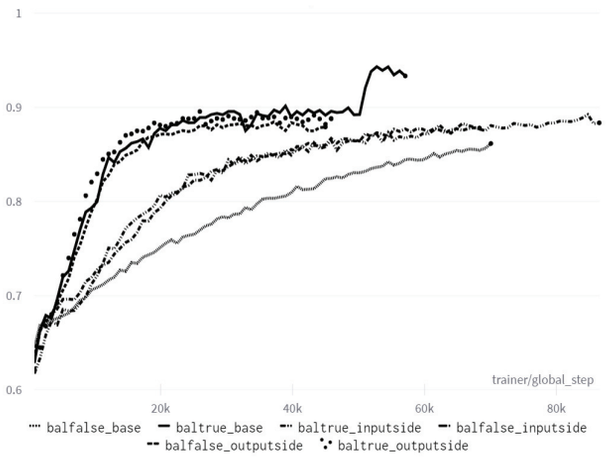

Fig. 4. Training Accuracy

5. 결 론

본 논문에서는 학습자가 온라인 저지에 코드를 제출하여 평가를 받는 전 단계에서 작성된 코드의 통과 여부를 예측하여 피드백을 제공하는 GRU 기반의 모델을 제안하였다. 제안한 모델은 소스코드 뿐만 아니라 소스코드가 해결하고자 하는 문제의 정보를 결합하여 context aware 한 pass/fail 예측이 가능하다. 정보 결합을 위해 FiLM 기법을 GRU가 학습하려는 토큰에 적용하는 방식과, GRU가 학습한 context vector에 적용하는 방식에 따라 Input side FiLM 모델과 Output side FiLM 모델을 제안했다. 제안한 모델의 검증을 위하여 본 논문에서는 온라인 저지를 통해 수집된 47개의 문제에 대하여 제출된 13,827개의 자바 코드를 학습하였고 Input side FiLM 모델의 정보를 결합하는 방법이 데이터의 비대칭과 상관없이 가장 높은 73.63% pass/fail 예측 정확도를 보였다.

본 논문에서는 GRU가 데이터 학습 과정에서 추가적인 문제정보를 FiLM 계층을 통하여 결합하는 것이 특정 문제 해결을 위해 작성된 코드가 pass/fail 하는지 예측하는 데 도움을 준다는 것을 확인하였다. 연구를 확장하여 문제 정보의 자연어 처리기반 결합 학습을 수행할 경우 더욱 높은 pass/fail 예측 정확도를 달성할 것으로 보인다.

References

[1] Leetcode - The World's Leading Online Programming Learning Platform [Internet], https://leetcode.com/.

[2] Baekjoon Online Judge [Internet], https://www.acmicpc.net/.

[3] H. S. Mun, S. H. Kim, J. H. Kim, and Y. S. Lee, "A source-code similarity-based automatic tutoring method for online coding test service," *Journal of KIISE,* Vol.48, No.9, pp.1044-1051, 2021.

[4] J. H. Ji, G. Woo, and H. G. Cho, "A plagiarism detection technique for Java program using bytecode analysis," *Journal of KIISE Software and Applications,* Vol.35, No.7, pp.442-451, 2008.

[5] J. W. Son, S. B. Park, and S. Y. Park, "Program plagiarism detection using parse tree kernels," *Pacific Rim International Conference on Artificial Intelligence,* Springer, Berlin, Heidelberg, pp.1000-1004, 2006.

[6] L. Prechelt, G. Malpohl, and M. Philippsen. "Finding plagiarisms among a set of programs with JPlag," *Journal of Universal Computer Science*, Vol.8, No.11, pp.1016-1038, 2002.

[7] R. Gupta, S. Pal, A. Kanade, and S. Shevade, "Deepfix: Fixing common c language errors by deep learning," *Thirty-First AAAI Conference on Artificial Intelligence,* Vol.31, No.1, 2017.

[8] U. Z. Ahmed, P. Kumar, A. Karkare, and S. Gulwani, "Compilation error repair: For the student programs, from the student programs," *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training*, pp.78-87, 2018.

[9] S. Bhatia, P. Kohli, R. Singh, P. Kohli, and R. Singh, "Neuro-symbolic program corrector for introductory programming assignments," *2018 IEEE/ACM 40th International Conference on Software Engineering,* pp.60-70, 2018.

[10] S. Wang, D. Chollak, and D. Movshovitz-Attias, "Bugram: Bug detection with n-gram language models," *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering,* pp.708-719, 2016.

[11] Y. Pu, K. Narasumhan, A. Solar-lezama, and R. Barzilay "sk_p: A neural program corrector for MOOCs," *Companion Proceedings of the 2016 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity*, pp.39-40, 2016.

[12] M. Vasic, A. Kanade, P. Maniatis, D. Bieber, and R. Singh, "Neural program repair by jointly learning to localize and repair," *arXiv preprint arXiv:1904.01720*, 2019.

[13] R. Gupta, A. Kanade, and S. Shevade, "Neural attribution for semantic bug-localization in student programs," *Advances in Neural Information Processing Systems,* 2019.

[14] E. Perez, F, Strub, H. DE Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol.32, No.1, 2018.

현 경 석

https://orcid.org/0000-0001-7626-8597
e-mail : ks_hyun@korea.ac.kr
2013년 고려대학교 컴퓨터교육과(학사)
2013년 ~ 현 재 고려대학교 컴퓨터학과 석・박사통합과정
관심분야 : Query Processing, Music Source Separation, Computer Science Education

최 우 성

https://orcid.org/0000-0003-2638-2097
e-mail : ws_choi@korea.ac.kr
2013년 고려대학교 컴퓨터교육과(학사)
2021년 고려대학교 컴퓨터학과(박사)
2021년 ~ 2022년 Centre for Digital Music at Queen Mary University of London(박사후과정)
2022년 ~ 현 재 Sony Group Corporation 연구원
관심분야 : Deep Learning, Signal Processing, Music Source Separation, Text-Guided Audio Manipulation

정 재 화

https://orcid.org/0000-0002-1904-7189
e-mail : jaehwachung@knou.ac.kr
1999년 고려대학교 컴퓨터교육과(학사)
2011년 고려대학교 컴퓨터교육과(석・박사)
2012년 ~ 현 재 한국방송통신대학교 컴퓨터과학과 교수
관심분야 : Audio Source Separation, Spatial Sound Scene Synthesis, Spatio-Temporal Data Management