학습자 수준을 기반으로 한 온라인 저지 시스템의 평가 기법 분류

한상곤, 류샤오, 우균

부산대학교 컴퓨터공학과
e-mail: {sangkon, liuxiao, woogyun}@pusan.ac.kr

Classifying of Evaluation Methods of Online Judge System Based on The Levels of Learners

Sangkon Han, Xiao Liu, Gyun Woo
Dept. of CSE, Pusan National University

요 약

프로그래밍 역량에 대한 요구가 높아지면서, 프로그래밍 교육에 온라인 저지(online judge)와 같은 학습 플랫폼에 대한 논의가 활발하게 진행되고 있다. 온라인을 기반으로 한 프로그래밍 학습 플랫폼인 온라인 저지(online judge)는 학습자가 제출한 프로그래밍 코드를 자동으로 평가한다. 본 논문은 온라인 저지에서 학습자의 수준을 기반으로 한 평가기법을 분류하였다. 온라인 저지에서 동적 평가 기법과 정적 평가 기법을 조합하여 활용함으로써 프로그래밍 역량을 향상시킬 수 있을 것으로 기대된다.

1. 서론

인공지능(artificial intelligence)이 다양한 곳에서 사용되기 시작하면서, 관련 연구가 활발히 진행되고 있다[1,2]. 프로그래밍 학습을 보조하고, COVID-19로 인해서 비대면 교육에 대한 요구가 많아지고 있다. 온라인에서 프로그래밍 학습을 보조하는 플랫폼인 온라인 저지(online judge)에 대한 관심이 높아지고 있다[3,4].

본 논문은 프로그래밍 학습에 사용되는 온라인 저지의 평가 종류와 기법을 분류하였다. 이를 바탕으로 학습자 수준을 기반으로 한 평가 기법을 제안한다.

2. 관련 연구

온라인을 기반으로 학습자가 프로그래밍을 배울 수 있는 플랫폼을 온라인 저지(online judge)라 한다. 온라인 저지는 사람이 개입하지 않고, 자동으로 채점하는 자동 채점 시스템(auto-judge system)의 일종이다. 이 용어는 2001년 Kurnia와 Lim, Cheange에 의해서 처음 소개되었다[1].

온라인 저지는 UI, 평가 프로그램, 문제, 검증 데이터로 구성된다. UI는 사용자에게 필요한 기능을 제공한다. 문제는 프로그래밍을 통해 해결해야 하는 과제를 뜻한다. 검증 데이터는 학습자의 코드가 정확하게 작동하는지 검증하기 위한 입력과 출력 데이터로 쌍(pair)으로 구성된다. 검증 데이터는 학습자에게 공개하지 않는 것을 원칙으로 한다. 평가 프로그램은 학습자가 작성한 코드를 컴파일하고, 검증 데이터를 사용하여 자동으로 검증과 평가를 진행한다. 표 1은 온라인 저지의 주요 구성 요소를 정리한 것이다.

표 1 온라인 저지 구성 요소

구성 요소	설명
UI	온라인 저지를 사용할 때 필요한 기능을 제공함
평가 프로그램	학습자의 코드를 컴파일하고 검증 데이터를 사용해서 자동으로 채점을 진행
문제	프로그래밍을 통해 해결해야 하는 과제
검증 데이터	과제를 검증하기 위한 입력과 출력 데이터 쌍(pair)

3. 온라인 저지의 평가 종류

프로그램을 검증(test)하는 방법은 정적(static)과 동적(dynamic)으로 분류할 수 있다. 정적 검증은 학습자의 코드를 컴파일 또는 실행하지 않고 형식(style)이나 문법 등을 검증한다. 동적 검증은 프로그램을 실행해서 입력과 출력을 사용해서 프로그램이 정확하게 작동하는지 검증한다[6].

온라인 저지 플랫폼의 경우 대부분이 동적 평가를 기반으로 학습자의 코드를 평가하고 있다.

3.1 동적 평가의 목적

동적 평가는 학습자가 제출한 코드를 검증 데이터를 사

용해서 평가한다. 검증 데이터는 문제에 제시된 제한 조건과 관련된 사항을 반영한다. 학습자가 제출한 코드가 문제의 제한 조건을 만족하면서, 정확하게 작동하는지 평가하는 것이 동적 평가의 중요한 목적이다.

3.2 정적 평가의 목적

정적 평가는 학습자가 제출한 코드의 가독성이나 복잡도를 평가하는 것이다. 예를 들어, Python으로 문제를 해결하는 과정에서 반복문(for, while) 대신에 리스트 함축(list comprehension)을 사용했는지와 같이 문제 해결에 사용되는 언어의 문법적 요소, 구성 등을 평가하는 것이다.

동적 평가가 학습자가 제출한 코드의 정확한 작동 여부에 평가 기준을 두고 있다면, 정적 평가는 학습자의 코드 작성 규칙이나 풀이 방법, 구성 등을 평가하는데 중점을 두고 있다.

4. 평가 기법

동적 평가와 정적 평가는 기준이 다르다. 기준이 다르기 때문에 평가 기법에 차이를 보인다. 온라인 저지는 동적 평가와 정적 평가를 함께 제공해야 하며, 이때 평가 종류에 따라 적절한 기법을 제공해야 한다.

4.1 동적 평가 기법

4.1.1 일반 평가

일반 평가는 입력 데이터를 기반으로 출력 내용을 평가한다. 출력 내용에 포함된 구분자 등을 사용해서 토큰(token)을 분리한다. 분리된 토큰과 개수를 검증에 사용한다. 온라인 저지에서 코드를 평가할 때 가장 널리 사용되는 기법 중 하나다. 일반 평가는 알고리즘의 결과를 검증하거나, 수식 등을 평가하는 데 적합하다.

4.1.2 완전 일치 평가

완전 일치는 구분자를 사용하지 않는다. 완전 일치는 공백을 포함하여 출력 형식까지 동일한지 비교하여 평가를 진행한다. 완전 일치의 경우 출력 내용이 XML, JSON 등과 같이 특정한 형식을 가져야 하는 문제에 적합하다.

4.1.3 줄 단위 평가

줄 단위 평가는 개행 문자를 구분자로 사용해서 토큰을 분리한다. 줄 단위 평가는 공백의 포함 여부를 반영할 수 있다. 줄 단위 평가는 문자열 기반의 서식과 관련된 문제에 적합하다.

4.1.4 정렬 평가

문제의 제한 조건이 동시성(concurrency)이나 병렬성(parallelism)을 기반으로 했을 때, 출력의 순서가 무작위인 경우가 발생한다. 이때, 결과를 정렬한 후 평가를 진행할 수 있다.

4.2 정적 평가 기법

4.2.1 적합성

학습자가 해결해야 하는 문제의 제한 조건을 충족하는지가 평가의 기준이 된다. 예를 들어, 문제의 제한 조건이 특정 자료구조를 사용하지 못하도록 했을 때 해당 조건을 평가하는 것이다.

4.2.2 가독성

코드의 가독성을 측정하는 것으로, 일반적으로 학습자가 제출한 코드의 길이나 단어의 분량 등이 기준이 된다.

코드 가독성의 경우 문장 난이도와 단어 난이도를 기반으로 한 가독성 지수인 ARI[7], 분류기 기반의 가독성을 평가하는 도구[8] 등을 활용해서 가독성을 평가할 수 있다.

4.2.3 복잡도

복잡도는 코드에 사용된 연산자, 제어문 등을 기준으로 평가한다. 연산자와 피연산자의 개수를 기반으로 복잡도를 평가하는 Halstead Complexity Volume[9]이나, 제어 흐름을 기반으로 한 McCabe’s Cyclomatic Complexity[10]등을 사용하여 복잡도를 평가할 수 있다.

4.2.4 형식

코드 형식이란, 코드를 작성하는 편집 규약이다. C++처럼 프로그래밍 언어 사용자들의 암묵적인 합의에 의해서 만들어진 코드 형식(style)[11]이 있는 반면, 프로그래밍 언어를 개발하는 단체나 설계자에 의해서 공식적으로 코드 형식을 강제하는 Python[12] 같은 사례도 있다.

코드 형식을 강제하는 언어인 Python의 경우엔 언어에서 제시하는 형식에 대한 평가가 필요하다. 반면, C/C++처럼 암묵적인 합의나 관례를 기반으로 한 언어는 제출된 코드가 일관된 형식으로 작성되어 있는지를 평가할 수 있다.

5. 평가 기법

온라인 저지에서 사용하는 평가 기법은 사용자의 학습 단계와 온라인 저지의 사용 목적도 함께 고려되어야 한다. 예를 들어, 온라인 저지가 대학 교육과 개발자 채용에 사용될 경우 온라인 저지의 사용 목적이 다르기 때문에 평가 기법을 동일하게 할 수 없다. 그리고 대학 교육에 사용될 때 학습자의 학습 단계를 고려해야 교육 효과를 높일 수 있다.

5.1 학습 단계

학습 단계를 고려해서 평가 기법을 분류하였다. 초급 단계의 간단한 알고리즘이나 수식, 문자열 출력 등을 주로 학습한다. 일반 평가와 줄 단위 평가를 기반으로 제한 조건을 충족하는지 평가하는 적합성 평가를 진행해야 한다. 중급 단계의 학습자는 코드의 가독성을 추가로 평가해야 한다. 고급 단계의 학습자는 완전 일치 평가와 정렬 평가를 적용해서 고급 수준의 프로그래밍 기법을 학습하고, 더 나아가서 복잡도 평가를 통해서 프로그래밍 역량을 높일 수 있도

록 평가를 적용한다. 표 2는 학습자 단계를 고려해서 평가 기법을 제안한 것이다.

표 2 학습자 수준에 따른 평가 기법 분류

학습 단계	평가 기법	
	동적	정적
초급	일반 평가 줄 단위 평가	적합성
중급	일반 평가 줄 단위 평가	적합성 가독성
고급	일반 평가 완전 일치 평가 정렬 평가	적합성 가독성 복잡도

5.2 사용 목적

수업에 사용할 경우, 일반 평가와 줄 단위 평가를 기반으로 제한 조건을 충족하는지 판단하는 적합성을 평가의 기준으로 제시해야 한다. 채용 과정에 사용된다면, 출력 형식을 고려한 완전 일치 평가와 코드의 가독성과 복잡도를 함께 평가해야 한다. 개발 관련 업무 교육의 경우, 동시성과 병렬성 등과 같은 고급 기술에 대한 학습을 진행하고, 사내에 적용된 코드 규칙을 준수해야 하기 때문에 형식성을 포함해서 평가를 진행해야 한다. 표 3은 사용 목적에 따른 평가 기법을 정리한 것이다.

표 3 사용 목적에 따른 평가 기법 분류

사용 목적	평가 기법	
	동적	정적
대학 수업	일반 평가 줄 단위 평가	적합성
코딩 인터뷰	일반 평가 줄 단위 평가 완전 일치 평가	적합성 가독성 복잡도
개발 관련 업무 교육	일반 평가 완전 일치 평가 줄 단위 평가 정렬 평가	적합성 가독성 복잡도 형식성

6. 결론

본 논문은 프로그래밍 학습자의 코드를 평가하는 동적 평가 기준과 정적 평가 기준을 제시하였다. 온라인 저지에서 동적 평가 기법과 정적 평가 기법을 활용해서 학습자의 프로그래밍 역량을 향상시킬 수 있다.

향후 연구를 통해서 본 논문에서 제시한 평가 기법의 효율성을 검증하고, 평가 기법에 대한 세부적인 기준을 연구하여 제시할 예정이다.

ACKNOWLEDGEMENT

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2014-3-00035, 매니코어 기반 초고성능 스케일러블 OS 기초연구 (차세대 OS 기초연구센터)). *교신 저자: 우균(부산대학교, woogyun@pusan.ac.kr).

참 고 문 헌

[1] 양희태 외, “인공지능 기술 전망과 혁신정책 방향–국가 인공지능 R&D 정책 개선방안을 중심으로,” 정책연구, pp.1-321, 2019.

[2] 과학기술정보통신부, “인공지능 국가전략,” 문서번호 11-1721000-000392-01, 2019.

[3] 김수환, 김성훈, 이민정, 김현철, “K-12 학생 및 교사를 위한 인공지능 교육에 대한 고찰,” 컴퓨터교육학회 논문지, vol.23(4), pp.1-11, 2020.

[4] 장원영, 김성식, “자동평가시스템을 활용한 프로그래밍 교육에서 블록형 언어와 텍스트형 언어 간 자기효능감의 차이,” 컴퓨터교육학회 논문지, vol.23(4), pp.23-33, 2020.

[5] Kurnia, Andy, Andrew Lim, and Brenda Cheang. “Online judge,” Computers & Education, vol.36.4, pp.299-315, 2001

[6] 이석수, 오원찬, 박선녀, 조은선, 백인성, “정적, 동적 분석방법을 결합하는 바이너리 코드 취약점 분석 프레임워크,“ 정보과학회논문지, vol.45(12), pp.1217-1226, 2018.

[7] Senter, R. J., Smith, E. A., “Automated readability index,“ Cincinnati university, 1967.

[8] Buse, R. P., Weimer, W. R., “Learning a metric for code readability,” IEEE Transactions on Software Engineering, vol.36(4), pp.546-558. 2009.

[9] Hariprasad, T., et al., “Software complexity analysis using halstead metrics,“ International Conference on Trends in Electronics and Informatics (ICEI). IEEE, 2017.

[10] Abran, Alain, Miguel Lopez, and Naji Habra, “An analysis of the McCabe Cyclomatic complexity number,“ Proceedings of the 14th International Workshop on Software Measurement (IWSM) IWSM-Metrikon. 2004.

[11] Google. (2013). Google C++ Style Guide [Online]. Available: https://google.github.io/styleguide/cppguide.html

[12] Van Rossum, Guido, Barry Warsaw, and Nick Coghlan. (2001). PEP 8: style guide for Python code [Online]. Available: https://www.python.org/dev/peps/pep-0008/

[13] Lee, Taek, Jung Been Lee, and Hoh Peter In, “A study of different coding styles affecting code readability,” International Journal of Software Engineering and Its Applications, vol.7(5), pp.413-422, 2013.