자동화된 피드백을 통한 코드 생성 LLM 성능 향상 전략

유미선 [1], 백윤흥 [1]
[1] 서울대학교 전기정보공학부, 서울대학교 반도체 공동연구소

msyu@sor.snu.ac.kr, ypaek@sor.snu.ac.kr

Strategies for Enhancing Code Generation LLM Performance through Automated Feedback

Miseon Yu[1], Yunheung Peak[1]
[1]Dept. of Electrical and Computer Engineering and Inter-University Semiconductor Research Center (ISRC), Seoul National University

요 약

최근 몇 년간 대규모 언어 모델(LLM)은 자연어 처리(NLP) 분야에서 인간의 언어를 이해하고 생성하는 능력으로 큰 주목을 받아왔다. 초기에는 텍스트 생성, 번역, 질의응답 시스템과 같은 작업에 주로 사용되었으나, 최근에는 코드 생성과 같은 복잡한 기술 작업에도 응용되고 있다. 그러나 LLM이 생성한 코드는 문법적 오류, 논리적 결함, 실행 불가능한 문제 등 다양한 문제점을 포함할 수 있다. 이러한 문제를 해결하기 위해 LLM이 스스로 코드를 검증하고 개선할 수 있는 자동화 피드백 시스템이 주목받고 있다. 본 논문에서는 코드 생성 LLM의 성능을 향상시키기 위한 주요 자동화 피드백 메커니즘들을 분석한다. 이러한 자동화 피드백 시스템은 소프트웨어 개발 과정을 자동화하고 최적화하는 데 중요한 역할을 할 수 있으며, 향후 연구에서는 이 시스템의 정교화 및 코드 생성 분야에서의 확장 가능성에 대한 탐구가 필요할 것이다.

1. 서론

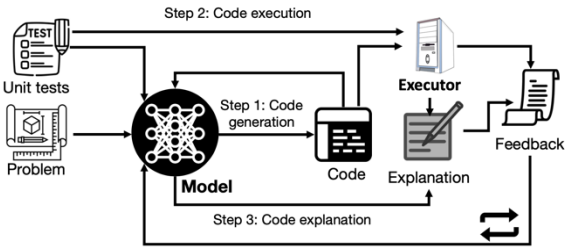
최근 몇 년간 자연어 처리(NLP) 분야에서는 대규모 언어 모델(Large Language Models, LLM)이 큰 관심을 받고 있다. LLM은 인간의 언어를 이해하고 생성하는 능력을 바탕으로 다양한 작업에서 놀라운 성능을 발휘하고 있다. 초기에는 텍스트 생성이나 번역, 질의응답 시스템에 주로 활용되었으나, 최근에는 이러한 기술이 더욱 확장되어 코드 생성과 같은 복잡한 기술 작업에도 응용되고 있다. 기존에는 사람이 직접 작성해야 했던 복잡한 코드나 반복적인 작업들을 LLM을 통해 자동화할 수 있기 때문이다. 실제로 GitHub는 코드 생성 LLM을 기본으로 한 Copilot Chat 서비스를 여러 소프트웨어 작업에 도입하였다. [1]에 따르면, Copilot은 백만명 이상의 개발자들과 5천 개 이상의 기업에서 널리 사용되고 있다. 하지만 LLM이 생성한 코드가 항상 완벽한 것은 아니다. 코드의 문법적 오류, 논리적 결함, 그리고 실행 불가능한 코드를 생성하는 문제는 여전히 빈번하게 발생한다. [2] 이러한 문제를 해결하기 위해 최근 연구에서는 LLM이 스스로 코드를 검증하고 수정하는 자동화된 피드백 시스템에 대한 관심이 높아지고 있다. [1] 자동화된 피드백을 통해 LLM이 생성한 코드의 문제를 감지하고, 이를 다시 개선하는 과정을 반복함으로써, 코드의 정확성과 실행 가능성을 높인다. 이러한 피드백 시스템은 사람이 일일이 코드를 검증하고 수정하는 과정을 대체하거나 보조하여, 시간과 비용을 절감하는 효과를 기대할 수 있다. 따라서 본 논문에서는 코드 생성 LLM의 성능을 향상시키기 위한 주요 자동화 피드백 전략들을 분석하고, 각각의 방법들이 코드 생성에서 어떤 역할을 하는지 살펴본다. 이를 통해 코드 생성 LLM이 앞으로 어떻게 발전할 수 있을지에 대한 방향성을 제시하고자 한다.

2. 코드 생성 LLM의 자동화 피드백 메커니즘

코드 생성 LLM의 성능을 향상시키기 위한 자동화 피드백 메커니즘은 여러 방식으로 구현되며, 그중 대표적인 방법으로 코드 검증 및 실행 기반 피드백이 있다. 코드 생성 후, 모델이 출력한 코드를 실제로 실

행하여 피드백을 제공하는 방식이 널리 사용되고 있다. 예를 들어, Self-Debug [3]와 ALGO [4] 시스템은 코드의 실행 결과를 분석해 피드백을 제공하는 대표적인 방식이다. LLM 이 생성한 코드에 대해 실행 후 피드백을 받아 수정하는 방법으로, 생성된 코드가 실행된 이후 피드백을 바탕으로 한 번 더 개선할 기회를 제공함으로써 더 정확한 결과를 얻을 수 있게 한다.

Self-Debug 는 LLM 이 생성한 코드의 실행 결과를 바탕으로 오류를 감지하고, 이를 설명하는 피드백을 LLM 에게 제공함으로써 코드를 개선하는 프레임워크를 제시한다. 코드 실행 과정에서 발생한 오류 메시지나 예외 처리 정보를 분석해, 해당 오류를 수정할 수 있도록 LLM 에게 자연어 피드백을 제공한다. 해당 전략은 그림 1 에서 볼 수 있듯, 3 가지 주요 단계로 이루어져 있다. 첫번째로, 주어진 문제를 바탕으로 모델이 코드를 생성한다. 다음으로, 생성된 코드에 대한 실행 결과 및 코드 설명 등의 정보를 추출한다. 마지막으로, 생성된 코드와 두번째 단계에서 생성된 정보들을 통합하여 피드백 메세지를 만들고 이를 다시 LLM 에 주어 코드를 개선하도록 한다. [3]



(그림 1) Self-Debug 개요도.[3]

ALGO 는 알고리즘 생성을 위한 두 가지 모듈인 검증기와 코드 생성기를 포함한다. 검증기는 시간 효율성과 상관없이 가능한 모든 경우를 탐색하여 오라클의 역할을 하며, 코드 생성기는 더 시간 효율적인 방법으로 해결책을 생성한다. 이후 생성된 코드의 출력은 오라클의 출력과 비교되어 주어진 테스트 입력에 대한 정확성을 평가한다. 검증 결과와 함께 관련 테스트 케이스가 코드 생성기에게 함께 제공되어 코드를 개선하도록 한다. 이 시스템은 단순히 코드가 잘못 실행되었는지를 넘어서, 코드가 의도한 대로 동작하지 않는 부분을 탐지하고, LLM 에게 그 부분에 대한 세부적인 피드백을 제공한다. 이를 통해 코드의 기능적 오류뿐 아니라 논리적 오류도 수정할 수 있도록 돕는다. [4]

CodeRL 은 사전 훈련된 언어 모델(Pretrained Language Model, LM)과 심층 강화 학습(Deep Reinforcement Learning, RL)을 결합하여 코드 생성 성능을 향상시키는 프레임워크를 제안했다. 이 방법론은 코드 생성 과정에서 중요한 피드백 신호인 유닛 테스트 결과를 학습 과정에서 적극적으로 활용하여 개선된 코드가 나오도록 LM 을 학습시킨다. 학습뿐만 아니라 추론 시에도 유닛 테스트 피드백을 활용하여 프로그램을 재생성 하거나, 코드 서브 시퀀스 중 올바른 부분을 선택하여 다시 생성하도록 유도하는 샘플링 전략을 제안한다. 코드 생성 모델은 Actor 네트워크로, 생성된 코드의 기능적 올바름을 평가하는 Critic 네트워크는 유닛 테스트 결과를 기반으로 평가를 수행한다. [5]

3. 결론

코드 생성 LLM 은 다양한 자동화 피드백 메커니즘을 통해 개선될 수 있다. 코드 실행 기반 검증, 이중 실행 합의, 사후 수정 전략은 각기 다른 방식으로 코드의 오류를 감지하고 수정할 수 있으며, 이를 통해 LLM 은 점차 더 정확하고 안정적인 코드를 생성할 수 있는 능력을 갖추게 된다. 코드 생성 LLM 은 소프트웨어 개발을 자동화하고 개선할 수 있는 중요한 도구로서 발전하고 있다. 자동화된 피드백 시스템은 코드의 정확성과 실행 가능성을 크게 향상시킬 수 있으며, 다양한 전략들이 이미 적용되고 있다. 향후 연구에서는 이러한 자동화된 피드백 시스템의 정교화와 다양한 코드 생성 분야에서의 확장 가능성에 대한 탐구가 필요할 것이다.

ACKNOWLEDGEMENT

이 논문은 2024 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (RS-2023-00277326). 이 논문은 2024 년도 BK21 FOUR 정보기술 미래인재 교육연구단에 의하여 지원되었음. 이 논문은 2024 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (IITP-2023-RS-2023-00256081). 본 연구는 반도체 공동연구소 지원의 결과물임을 밝힙니다. 이 논문은 2024 년도 정부(산업통상자원부)의 재원으로 한국산업기술기획평가원의 지원을 받아 수행된 연구임.(No. RS-2024-00406121, 자동차보안취약점기반 위협분석시스템개발(R&D)). 이 논문은 2024 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구 결과임(No.RS-2024-00438729, 익명화된 기밀실행을 이용한 전주기적 데이터 프라이버시 보호 기술 개발)

참고문헌

[1] Thomas Dohmke. 2023. GitHub Copilot X: the AI-powered Developer Experience. https://github.blog/2023-03-22-github-copilot-x-the-ai-powered-developer-experience/.

[2] Liangming Pan, et al. Automatically Correcting Large Language Models: Surveying the landscape of diverse self-correction strategies. arXiv. 2023.

[3] Xinyun Chen, et al. Teaching large language models to self-debug. ICLR. 2024.

[4] Kexun Zhang, et al. ALGO: Synthesizing Algorithmic Programs with LLM-Generated Oracle Verifiers. NeurIPS. 2023.

[5] Hung Le, et al. CodeRL: Mastering Code Generation through Pretrained Models and Deep Reinforcement Learning. NeurIPS. 2022.