테스트 출력 예측을 위한 LLM 친화적 의사코드

김재진, 황승원
서울대학교

jaejin.kim@snu.ac.kr, seungwonh@snu.ac.kr

LLM-native Pseudocode for Test Output Prediction

Jaejin Kim, Seung-won Hwang
Interdisciplinary Program in Artificial Intelligence, Seoul National University

요 약

본 논문은 대규모 언어 모델(Large Language Models, LLMs)을 활용한 코드 생성(Code Generation) 과정에서, 테스트 출력 예측(Test Output Prediction) 성능을 향상시키기 위한 새로운 접근을 제안한다. 기존 연구들은 사람이 설계한 자연어 또는 의사코드(pseudocode) 형식을 활용하여 모델의 추론 과정을 유도했으나, 이러한 형식은 LLM 의 사전학습 분포와의 불일치로 인해 추론 성능을 저해할 수 있다. 이에 본 연구에서는 LLM 이 사전학습 중 익숙하게 접했을 가능성이 높은 표현 구조, 즉 LLM-native pseudocode 를 reasoning trace 로 활용하는 방법을 제안한다. 우리는 zero-shot 방식으로 LLM 에 자연스러운 pseudocode 형식을 탐색하고, 이를 few-shot 예시로 제시하는 간단한 프롬프트 구성만으로도 모델의 추론 안정성과 출력 정확도를 높일 수 있음을 실험적으로 확인하였다. LiveCodeBench 벤치마크를 기반으로 한 실험 결과, 제안된 방법은 기존 방식 대비 일관된 성능 향상을 보였으며, 특히 어려운 샘플에 대해서도 경쟁력 있는 결과를 달성하였다. 본 연구는 LLM-native 표현 방식이 코드 관련 추론 과제에서 효과적인 중간 표현이 될 수 있음을 시사하며, 향후 다양한 모델과 태스크로의 확장을 기대할 수 있다.

Ⅰ. 서 론

대규모 언어 모델(Large Language Models, LLMs)을 활용한 코드 생성(Code Generation) 연구가 활발히 진행되면서, 생성된 코드의 품질을 평가하거나 보완하기 위한 다양한 보조 과제들도 주목받고 있다. [4,5] 그 중 테스트케이스 생성(Testcase Generation)은 모델이 작성한 코드의 정확성과 일반화 능력을 검증하는 데 핵심적인 역할을 한다. [6]

테스트케이스 생성은 일반적으로 두 가지 하위 과제로 나눌 수 있다: 하나는 test input generation 이며, 다른 하나는 test output prediction 이다. 본 논문에서는 후자인 test output prediction 에 집중한다. 이 과제는 주어진 문제 설명(problem description)과 테스트 입력(test input)을 바탕으로, 그에 대응되는 출력(output)을 예측하는 태스크이다.

최근에는 이와 같은 추론 과정의 품질을 높이기 위해, Chain-of-Thought (CoT) [1] 방식을 활용하는 시도가 이루어지고 있다. CoT 는 모델이 최종 출력을 도출하기까지의 중간 추론 과정을 명시적으로 서술함으로써 정답률을 높이는 기법이다. 이 중간 reasoning chain 은 다양한 형식으로 표현될 수 있는데, 대표적으로 실제 코드(code) [2] 또는 의사코드(pseudocode) 형식이 사용된다. [3]

그러나 의사코드(pseudocode)는 실제 코드와 달리 명확하게 정해진 문법이나 구조가 없기 때문에, 모델이 중간에 생성한 의사코드의 표현 방식에 따라 성능이 민감하게 달라질 수 있다. 기존 연구에서는 사람이 설계한 형식을 few-shot 예시로 제공하거나, 일관된 스타일을 유도하기 위해 prompt 를 정교하게 구성하는 방식으로 이를 통제하려 했다. [3]

본 논문에서는 이러한 방식과는 다른 접근을 제안한다. 사람이 임의로 정의한 pseudocode 형식을 모델에게 강제하는 것보다, 사전학습(pretraining) 중 LLM 이 자주 접했을 법한 표현 방식, 즉 LLM-native 한 형식의 pseudocode 를 사용하는 것이 더욱 효과적이라는 것이다. 우리는 이러한 형식을 few-shot 예시로만 간단히 제시하더라도, 모델이 reasoning trace 를 더 자연스럽고 안정적으로 생성할 수 있음을 확인하였다.

즉, zero-shot 으로 LLM 에 친화적인 pseudocode 형식을 먼저 탐색한 뒤, 그 형식을 few-shot 예시 형태로 제시하는 것이, 전통적인 human-designed 형식에 기반한 prompting 보다 test output prediction 성능을 더 안정적으로 향상시킨다는 것이 본 연구의 핵심 주장이다.

Ⅱ. 과제 정의 및 접근 방식

Ⅱ-1. Test Output Prediction 과제 정의

Method	*Format*	*Pass@1*	*Easy-Pass@1*	*Medium-Pass@1*	*Hard-Pass@1*
Baseline	-	64.3	76.9	59.2	54.2
Constrained	Code	63.3	77.6	58.7	48.6
Constrained	T&E [3]	66.7	81.4	61.2	53.6
LPG	LLM-native	**67.6**	**81.6**	**62.8**	**54.2**

표 1. Test Output Prediction 과제에 대한 방법별 성능 비교
각 방법은 문제 설명과 테스트 입력을 바탕으로 정답 출력을 예측하는 과제를 수행하며, Pass@1, Easy-Pass@1, Medium-Pass@1, Hard-Pass@1 지표를 기준으로 평가되었다. Baseline 은 reasoning trace 없이 출력을 직접 예측하는 방식이며, Constrained 는 사람이 정의한 코드 또는 T&E(Trace-and-Execute) 형식의 pseudocode 를 활용하는 방식이다. LPG 는 본 논문에서 제안하는 LLM-native pseudocode 기반 접근법으로, 모든 난이도 구간에서 안정적이거나 우수한 성능을 보였다.

Test Output Prediction 은 문제 설명(problem description)과 특정 입력값(test input)이 주어졌을 때, 해당 입력에 대한 정답 출력을 직접 예측하는 과제이다. 이 과제는 실제 코드를 실행하지 않고도, 문제 해결 논리에 기반하여 정답 출력을 추론하는 것을 목표로 한다. 즉, 모델은 문제의 조건과 테스트 입력을 해석하고, 그에 따른 논리적 계산 및 흐름을 모델 내부에서 전개하여 출력값을 생성해야 한다.

본 과제는 기존의 코드 생성(Code Generation)이나 정답 선택(Answer Selection) 과제와는 달리, 출력값의 정확한 수치를 생성해야 하므로 reasoning 과정이 더 중요하게 작용한다. 이에 따라, 중간 reasoning trace 를 유도하는 Chain-of-Thought prompting 이 필수적으로 활용된다.

Ⅱ-2. LLM-native Pseudocode

기존 CoT 연구들은 중간 추론 과정을 자연어로 기술하거나, 사람이 설계한 pseudocode 형식을 삽입하는 방식으로 모델의 reasoning 을 유도하였다. 그러나 이러한 형식은 LLM 의 사전학습 분포와 괴리가 있는 경우, 오히려 모델의 혼란을 유발할 수 있다.

본 연구에서는 LLM 이 사전학습 중 익숙하게 접했을 가능성이 높은 표현 구조, 즉 LLM-native 한 pseudocode 형식을 사용하였다. 이를 위해 다양한 pseudocode 스타일을 탐색하고, 모델이 안정적으로 해석 및 생성할 수 있는 형식을 선택하였다. 이 형식은 특별한 규칙이나 키워드 없이, 자연스러운 파이썬 유사 문법과 논리적 구조만을 포함하며, 이를 few-shot 예시로 제공하였다.우리 접근은 다음의 특징을 가진다:

- 사람이 직접 설계한 형식이 아닌, 모델의 분포에 자연스럽게 호환되는 형식을 채택
- Zero-shot 으로 형식을 탐색한 뒤, 그 형식을 few-shot prompt 에 삽입하여 모델의 reasoning 을 유도
- 자연어 또는 실제 코드 대신 pseudocode 를 사용함으로써 추상화 수준을 조절하고 오류 가능성을 줄임

Ⅲ. 실험 및 결과

실험은 LiveCodeBench[7]의 test output prediction 벤치마크를 기반으로 수행되었으며, cutoff date 는 2023 년 5 월 1 일로 설정되었다. 생성 모델은 GPT-4-Turbo-2024-04-09 를 사용하였다.

표 1 은 self-consistency 없이 수행한 다양한 방법들의 성능을 비교한 결과이다. Baseline 은 reasoning trace 없이 직접 출력을 예측하는 방식이며, Constrained 는 사람이 설계한 pseudocode 형식을 사용하였다. LPG 는 본 논문에서 제안한 LLM-native pseudocode 방식이다. 우리 방식은 전 구간에서 기존 방식과 유사하거나 더 나은 성능을 보였으며, 특히 Hard-Pass@1 에서도 Baseline 을 상회하는 결과를 나타냈다.

Ⅳ. 결론

본 논문은 test output prediction 과제에서 reasoning trace 로 pseudocode 를 활용할 때, LLM 에 친화적인 형식(LLM-native)을 사용하는 것이 성능 향상에 유리하다는 점을 실증적으로 보여주었다. 사람이 설계한 형식을 따르기보다는, 모델의 사전학습 분포에 기반한 표현을 사용하는 것이 reasoning 안정성과 결과 정확도 모두에 긍정적인 영향을 줄 수 있다.

향후 연구에서는 다양한 모델과 과제에 대해 LLM-native 형식의 일반성을 검증하고, 해당 형식을 자동으로 탐색하거나 생성하는 방법론으로 확장할 수 있을 것이다.

.

ACKNOWLEDGMENT

이 논문은 2025 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 [NO.RS-2021-II211343,인공지능대학원지원(서울대학교)]

참 고 문 헌

[1] Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." Advances in neural information processing systems 35 (2022): 24824-24837.

[2] Li, Chengshu, et al. "Chain of code: Reasoning with a language model-augmented code emulator." arXiv preprint arXiv:2312.04474 (2023).

[3] Chae, Hyungjoo, et al. "Language models as compilers: Simulating pseudocode execution improves algorithmic reasoning in language models." arXiv preprint arXiv:2404.02575 (2024).

[4] Li, Kefan, and Yuan Yuan. "Large language models as test case generators: Performance evaluation and enhancement." arXiv preprint arXiv:2404.13340 (2024).

[5] Fakhoury, Sarah, et al. "Llm-based test-driven interactive code generation: User study and empirical evaluation." IEEE Transactions on Software Engineering (2024).

[6] Wang, Wenhan, et al. "Testeval: Benchmarking large language models for test case generation." arXiv preprint arXiv:2406.04531 (2024).

[7] Jain, Naman, et al. "Livecodebench: Holistic and contamination free evaluation of large language models for code." arXiv preprint arXiv:2403.07974 (2024).