효율적인 Tool Calling을 위한 APO 기반 Small Planner

육정훈[1], 조민경[1], 원인호[1], 유한결[2], 임경태[1*]

[1]한국과학기술원, [2]서울과학기술대학교

usually670@kaist.ac.kr, kveldsstjerne@kaist.ac.kr, kotmul2no@kaist.ac.kr, hgyoo@seoultech.ac.kr, ktlim@kaist.ac.kr

APO-Based Small Planner for Efficient Tool Calling

Yuk Jung Hun[1], Cho Min Kyung[1], Won In Ho[1], Yoo Han Gyeol[2], Lim Kyung Tae[1*]

[1]Korea Advanced Institute of Science and Technology (KAIST), [2]Seoul National University of Science and Technology

요 약

거대 언어 모델은 외부 도구 활용에서 높은 잠재력을 보였으나, 불필요한 탐색과 중복된 호출로 인해 추론 비용과 지연 시간이 증가하는 문제가 있다. 본 연구에서는 이러한 비효율성을 근본적으로 해결하기 위해 Small LLM Planner 학습을 위한 자동화된 선호도 데이터 구축 파이프라인과, 이를 통해 학습된 Planner를 제안한다. Berkeley Function Calling Leaderboard 벤치마크 실험 결과, 제안한 Planner를 적용한 Qwen-3 30B 모델은 베이스라인 대비 1.0%p~8.4%p의 성능 향상을 보였으며, 대부분의 세부 과제에서 평균 도구 호출 횟수를 감소시켜 비용 효율성이 향상됨을 확인하였다. 또한 본 연구에서 구축한 데이터셋과 파이프라인을 공개하여 효율적인 AI 에이전트 연구에 기여하고자 한다.

Ⅰ. 서 론

최근 거대 언어 모델은 외부 도구를 활용하여 복잡한 문제를 해결하는 Tool Calling능력에서 비약적인 발전을 이루었다 [1, 2]. 그러나 대규모 모델은 종종 불필요한 도구를 탐색하거나 과도하게 호출하는 경향을 보인다 [3]. 이를 해결하기 위해 대규모 모델 자체를 강화학습 등으로 학습하여 효율성을 높이는 연구가 진행되었으나 [4], 수천억 개의 파라미터를 가진 모델을 직접 학습시키는 것은 막대한 연산 자원과 비용을 초래한다는 한계가 있다.

이에 대한 대안으로, 비교적 가벼운 Small LLM을 Planner로 사용하여 대규모 모델의 도구 사용을 전략적으로 가이드하는 접근법이 주목받고 있다 [5, 6]. 이 방식은 앞단에서 효율적인 경로를 설계함으로써 시스템 전체의 추론 효율성을 높일 수 있으나, 대부분 대규모 모델의 출력을 단순 Distillation하여 비효율적인 도구 사용 패턴까지 그대로 답습할 수 있다.

본 연구에서는 이러한 문제를 해결하기 위해, Small LLM Planner를 위한 자동화된 Direct Preference Optimization(DPO) 데이터 구축 파이프라인을 제안한다. 우리는 대규모 모델이 생성한 '정답을 도출했으나 과정이 비효율적인 툴 사용 Trajector'에서 불필요한 단계를 제거 및 Synthesize하여, Efficient Plan과 Inefficient Plan의 Pair 데이터를 자동으로 생성한다. 이를 통해 별도의 사람 개입 없이도, Small LLM이 최소한의 도구 호출로 정확한 답을 도출하는 전략을 학습하도록 유도한다.

Ⅱ. 기존 연구

2-1. Tool-Augmented Language Models

LLM의 추론 능력을 확장하기 위해 외부 도구와 결합하려는 시도는 활발히 연구되어 왔다. 초기 연구인 ToolFormer [1]는 LLM이 스스로 API를 호출할 위치를 학습할 수 있음을 보였으며, ToolLLM [2]은 LLaMA 기반 모델이 실제 API를 사용하여 복잡한 지시를 수행하도록 Instruction Tuning을 수행했다. 최근에는 코드 인터프리터나 검색 API뿐만 아니라, 추론 과정 중간에 도구를 전략적으로 호출하는 에이전트 형태의 연구들이 주를 이룬다 [7, 8]. 그러나 이러한 접근법들은 도구 호출 횟수를 포함한 비용 효율적(Cost-effective) 도구 사용에 대한 직접적인 고려가 부족하여, 불필요한 API 호출로 인한 비용 문제를 야기한다.

2-2. Small LLM Guided Orchestration

대규모 모델 학습의 비효율성을 극복하기 위해, Small LLM을 활용한 Orchestration 연구가 대두되었다. LLM-Compiler [5]는 Small LLM이 병렬 실행 계획을 수립하게 하여 전체 실행 속도를 높였으며, Nvidia ToolOrchestra [6]는 8B 규모의 모델이 복잡한 툴 사용을 지휘하는 능력을 보여주었다. 하지만 기존 연구들은 Small LLM을 학습시키기 위해 사람의 개입이 필요한 데이터를 사용하거나, 거대 모델의 지식 증류 데이터를 그대로 학습하는 경우가 많다. 본 연구는 Large LLM의 비효율적인 경로를 Reverse-engineering하여 Small Planner를 Preference Data로 DPO 학습한다는 점에서 차이가 있다.

Ⅲ. 제안 방법

3-1. Synthesis from Inefficient Trajectories

우리는 강력하지만 비효율적인 Large LLM이 생성한 데이터의 특성을 활용한다. Large LLM은 정답을 맞히지만, 그 과정에서 불필요한 검색이나 중복된 호출 절차를 거치는 경우가 많다. 우리는 이 과정을 다음과 같이 정제한다.

Action Abstraction & Filtering 거대 모델이 만든 경로 데이터에서 실제 도구 호출(Tool Name)과 인자(Arguments) 정보 및 해당 Tool의 호출 근거를 추출한다.

Plan Synthesis LLM을 통해 최종 정답 도출에 기여하지 않은 불필요한 도구 호출을 제거하여 필수적인 행동만을 남긴 Tool 후보 List를 생성한다. 또한 해당 Tool 호출에 대한 근거를 추가로 출력하여, Small LLM이 이해하기 쉬운 형태의 데이터로 합성한다.

본 연구에서는 nvidia/Nemotron-Post-Training-Dataset-v1[9] 데이터셋의 Tool Calling Subset을 활용하여, 위의 두 과정을 통해 불필요한

Tool Calling을 제거한 데이터셋을 확보한다. Synthesize에 사용한 LLM은 Qwen-3 30B Instruction[10]모델이다.

3-2. Construction Preference Pairs for APO

효율적인 Planner 학습을 위해 우리는 APO 알고리즘을 적용한다. 이를 위해 앞서 생성한 데이터에서 Positive, Negative 쌍의 데이터를 설정하였다. Positive 데이터는 Play Synthesize단계에서 생성된 최적화 파이프라인을 거친 효율적인 Tool Call Plan 데이터로 설정하였다. Negative 데이터는 Action Abstraction&Filtering 단계에서 추출된 Large LLM의비효율적이고 중복이 포함된 Tool Call Plan이다. 이러한 쌍을 통해 Small LLM은 단순히 도구를 사용하는 법뿐만 아니라, 어떤 경로가 더 비용 효율적인가에 대한 선호도를 학습하게 된다. 이 파이프라인은 특정 모델에 종속되지 않으며, 어떤 Large LLM의 로그 데이터라도 Planner 학습용 데이터로 변환할 수 있는 범용성을 가진다.

3-3. Experiment Results and Analysis

제안하는 데이터셋으로 학습된 Small Planner의 유효성을 검증하기 위해, 최신 Tool Calling 성능 평가 벤치마크인 Berkeley Function Calling Leaderboard (BFCL)[11] 벤치마크를 활용하였다. 본 실험에서는 모델의 다양한 능력을 평가하기 위해 다음과 같은 세 가지 핵심 부분집합을 선정하였다.

Simple - Java 사용자 쿼리에 대해 단 한 개의 도구를 정확하게 선택하고 호출하는 능력을 평가한다. 특히 Java 언어 형식에 맞는 함수명과 인자(Argument), 문법 생성의 정확성(Syntax Accuracy)을 검증하는 데 초점을 둔다.

Multiple 하나의 쿼리를 해결하기 위해 두 개 이상의 도구를 호출해야 하는 시나리오이다. 주로 앞선 도구의 결과가 다음 도구의 입력으로 사용되는 등 Sequential Dependency이 있거나, 문맥을 유지하며 복합적인 추론이 필요한 능력을 평가한다.

Parallel 서로 의존성이 없어 동시에 실행 가능한(Independent and Simultaneous)다수의 도구 호출 능력을 평가한다. 모델이 여러 작업을 병렬적으로 처리하도록 계획할 수 있는지 검증하는 고난도 시나리오이다.

	Simple-JAVA	Multiple	Parallel Multiple
Qwen3 30b Inst	65.0(1.05)	92.5(1.02)	66.67(**2.21**)
w/ Planner	**66.0**(**1.04**)	**94.5**(**1.01**)	**75.00**(2.25)

표 1. BFCL 벤치마크의 세 가지 하위항목 평가 결과. 평균 도구 호출 횟수와 Accuracy를 의미한다.

Baseline으로는 Qwen-3 30B Instruction모델을 사용하였으며, Qwen-3 30B Inst w/ Planner은 본 연구에서 제안한 Planner가 생성한 최적화된 Tool List를 프롬프트에 함께 주입하여 추론하도록 설정하였다. 또한 학습 대상이 되는 Planner는 Qwen-3 4B Instruction 모델을 활용하였다.

Table 1은 BFCL 벤치마크에서의 성능 평가 결과를 보여준다. 실험 결과, Planner를 부착한 모델(Ours)이 모든 부분집합에서 Baseline 대비 우수한 성능을 보였다.

주목할 점은 Parallel Multiple에서의 결과이다. 해당 셋에서 제안하는 모델은 Baseline 대비 큰 성능 향상 폭을 보였으나, 평균 도구 호출 횟수는 증가하는 양상을 보였다. 이는 Baseline의 복잡한 병렬 쿼리에 대한 처리 성능을 Planner를 통해 향상시킬 수 있음을 의미한다. 즉, 호출 횟수의 증가는 기존 모델이 문제 해결에 필요한 도구를 호출하지 못하였으나, Planner를 통해 이를 보완하였음을 의미한다.

Ⅲ. 결론

본 연구에서는 강력하지만 비용 효율성이 떨어지는 Large LLM의 도구 사용 능력을 최적화하기 위해, Small LLM Planner 기반의 계층적 협업 프레임워크와 이를 위한 자동화된 데이터 합성 파이프라인을 제안하였다. 이를 통해 Small LLM은 단순히 상위 모델의 행동을 모방하는 것을 넘어, 최소한의 도구 호출로 정확한 답을 도출하는 최적 경로를 학습할 수 있었다.

우리는 향후 연구를 통해 본 파이프라인을 다양한 도메인의 API 환경으로 확장하고, Planner가 도구간의 의존성까지 고려 가능하도록 고도화할 계획이다. 본 연구에서 구축된 데이터셋과 코드를 커뮤니티에 공개함으로써 비용 효율적인 에이전트 연구의 저변 확대에 기여하고자 한다.

ACKNOWLEDGMENT

This work was supported in part by Korea Atomic Energy Research Institute R&D Program under Grant KAERI-524540-25.

참 고 문 헌

[1] Schick, Timo, et al. "Toolformer: Language models can teach themselves to use tools." Advances in Neural Information Processing Systems 36 (2023): 68539-68551.

[2] Qin, Yujia, et al. "Toolllm: Facilitating large language models to master 16000+ real-world apis." arXiv preprint arXiv:2307.16789 (2023).

[3] Zhang, Jieyu, et al. "Ecoassistant: Using llm assistant more affordably and accurately." arXiv preprint arXiv:2310.03046 (2023).

[4] Chen, Baian, et al. "Fireact: Toward language agent fine-tuning." arXiv preprint arXiv:2310.05915 (2023).

[5] Kim, Sehoon, et al. "An llm compiler for parallel function calling." Forty-first International Conference on Machine Learning. 2024.

[6] Su, Hongjin, et al. "ToolOrchestra: Elevating Intelligence via Efficient Model and Tool Orchestration." arXiv preprint arXiv:2511.21689 (2025).

[7] Gao, Luyu, et al. "Pal: Program-aided language models." International Conference on Machine Learning. PMLR, 2023.

[8] Qian, Chen, et al. "Chatdev: Communicative agents for software development." Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2024.

[9] Bercovich, Akhiad, et al. "Llama-nemotron: Efficient reasoning models." arXiv preprint arXiv:2505.00949 (2025).

[10] Yang, An, et al. "Qwen3 technical report." arXiv preprint arXiv:2505.09388 (2025).

[11] Patil, Shishir G., et al. "The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models." Forty-second International Conference on Machine Learning.