Introduction to Tool LLM

2026. 07. 17. 금

Data Mining & Quality Analytics Lab.

김수림

About Me

❖ 김수림 (Soorim Kim)

- 고려대학교 산업경영공학과 석사과정 (2025.09 ~ Present)
- Data Mining & Quality Analytics Lab. (김성범 교수님)

❖ Research Interest

- Large Language Models
- Federated Learning
- Time Series Analysis

❖ Contact

- srkim1@korea.ac.kr

Contents

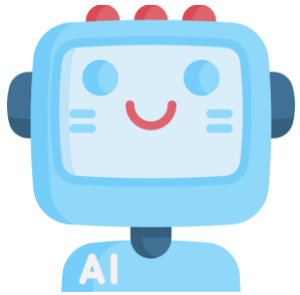
What powers today's LLM agents?

❖ **요즘 LLM 에이전트는 스스로 일을 처리한다**

- 최근 LLM은 단순 언어 생성을 넘어, 코드 작성·리서치·문서정리처럼 사람이 직접하던 작업을 자동으로 수행
- 하나의 모델이 아닌 여러 전문 에이전트가 협업하는 오케스트레이션 방식으로 확장되는 추세

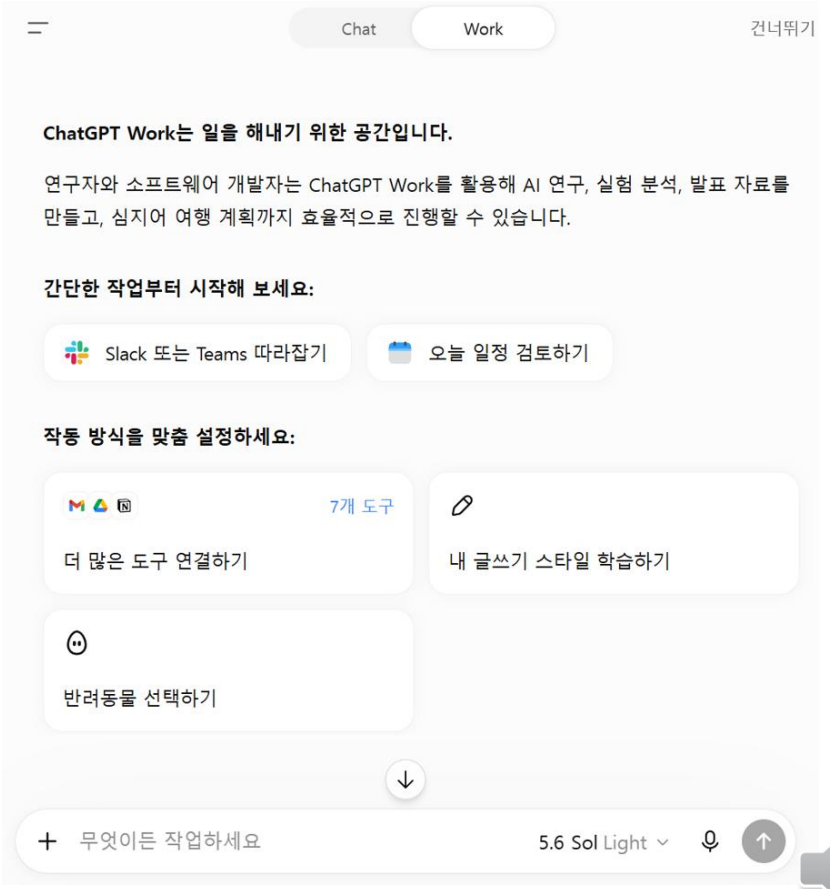
내일 프로젝트 회의 일정 잡고, 관련 문서 요약해서 이메일로 보내줘

다음 주 제주도 2박 3일 항공권과 호텔 예약해줘



이 코드 버그 수정하고 테스트까지 실행해줘

지난달 매출 데이터 분석해서 그래프로 그려줘



What powers today's LLM agents?

❖ **LLM이 똑똑해진 것이 아닌, 도구(Tool)를 쓰기 시작했기 때문**

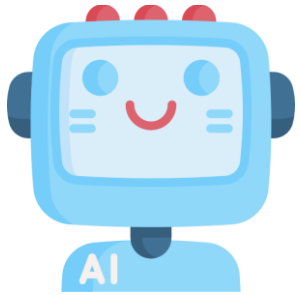
- 검색·계산·외부API 같은 도구를 호출하면서, 내부 지식에 갇혀있던 LLM이 현실 세계로 확장됨
- 지금의 도약은 더 큰 모델이 아니라 도구를 쓸 줄 아는 모델로의 변화에서 비롯됨

내일 프로젝트 회의 일정 잡고, 관련 문서 요약해서 이메일로 보내줘

- 캘린더 API: 캘린더 조회 → 회의 일정 등록
- 문서 검색: 관련 회의 문서 검색 → 핵심 내용 요약
- 메일 API: 요약본 첨부 → 참석자에게 발송

다음 주 제주도 2박 3일 항공권과 호텔 예약해줘

- 항공편 검색: 항공편 조회 → 시간/가격 비교
- 호텔 예약 API: 조건 맞는 숙소 검색 → 2박 예약
- 결제 API: 예약 확정 및 결제 처리



이 코드 버그 수정하고 테스트까지 실행해줘

- 코드 읽기: 파일 탐색 → 버그 위치/원인 파악
- 코드 수정: 해당 코드 수정 → 변경 사항 적용
- 테스트 실행: 유닛 테스트 실행 → 통과 여부 확인

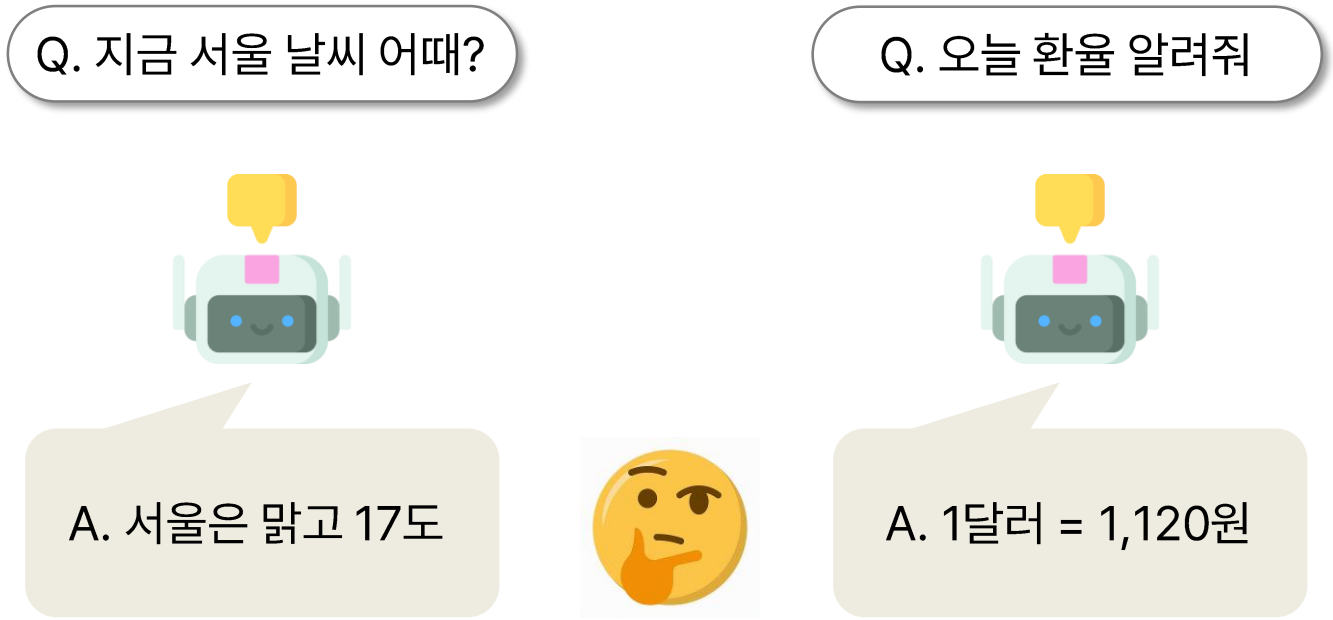
지난달 매출 데이터 분석해서 그래프로 그려줘

- DB 조회: 지난달 매출 데이터 쿼리 → 추출
- 코드 실행: 데이터 집계/분석 (ex. pandas)
- 그래프 생성: 분석 결과를 차트로 시각화

What powers today's LLM agents?

❖ **LLM의 한계를 도구로 해결**

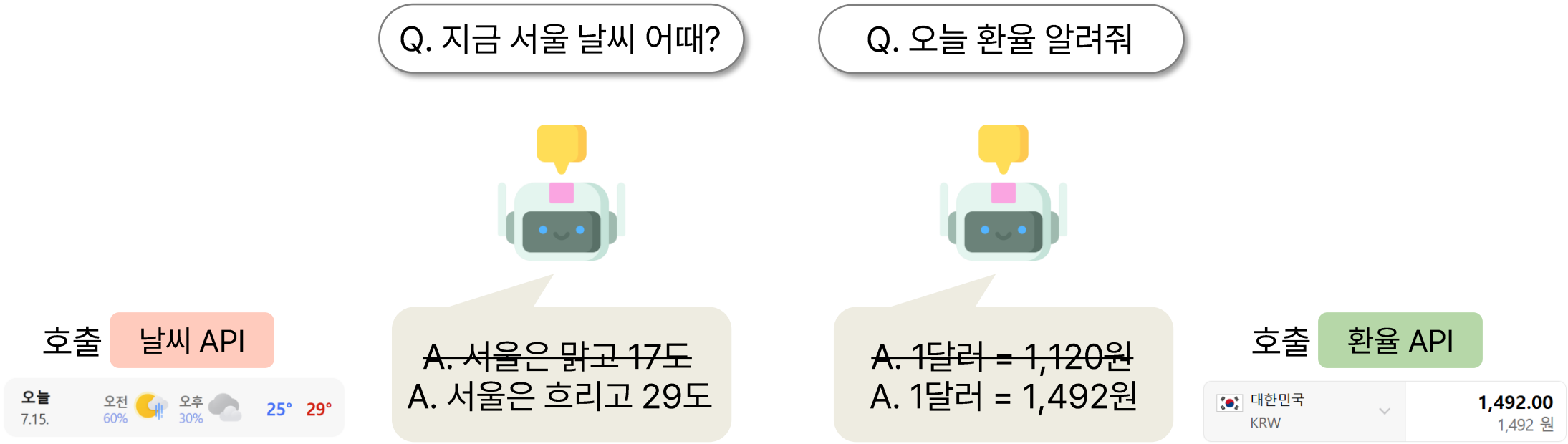
- 학습 시점에 접한 데이터에만 의존하여 최신 정보에 취약, 정밀 계산 약함, 외부 행동 불가
- 모른다고 답하는 대신 그럴듯하게 지어내는 환각(hallucination) 발생



What powers today's LLM agents?

❖ LLM의 한계를 도구로 해결

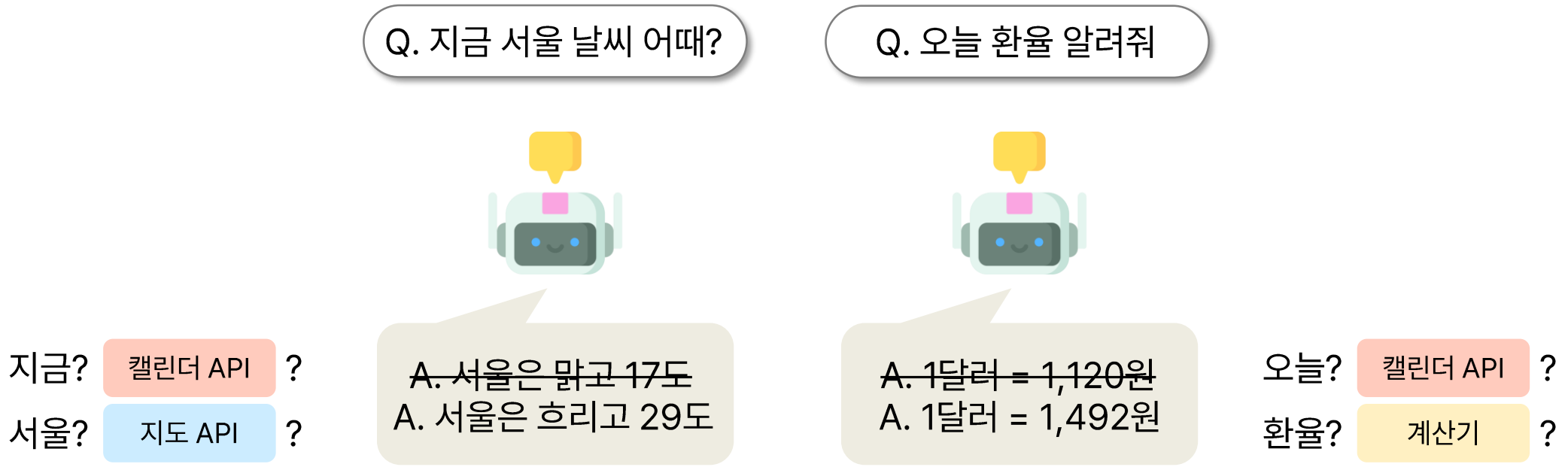
- 학습 시점에 접한 데이터에만 의존하여 최신 정보에 취약, 정밀 계산 약함, 외부 행동 불가
- 모른다고 답하는 대신 그럴듯하게 지어내는 환각(hallucination) 발생 → 부족한 정보와 기능을 도구로 조회·검증하며 보완



What powers today's LLM agents?

❖ **도구만 주어지면 모든 일을 잘 해낼까?**

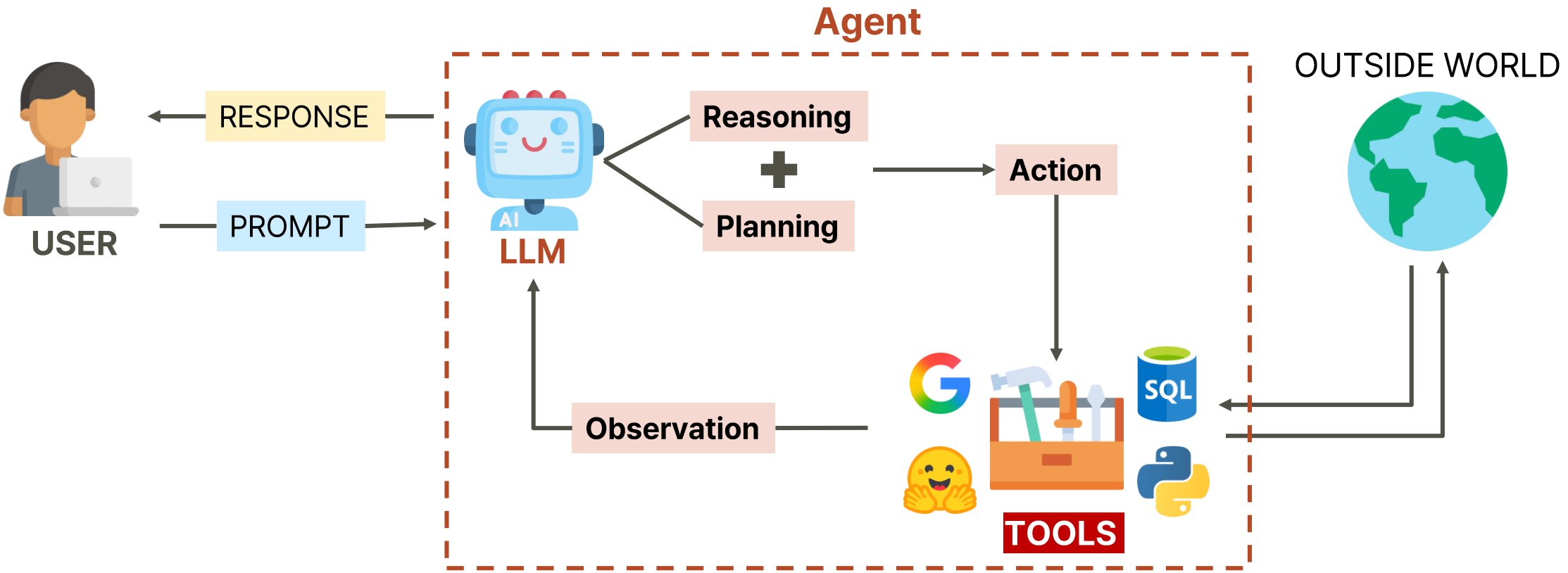
- 도구가 있어도 언제·왜·어떤 도구를 써야 할지 판단하지 못하면 무용지물
- LLM의 생각(reasoning)과 행동(acting)이 단절 → 도구의 사용 근거를 스스로 마련하지 못함



Is having tools enough?

❖ **도구를 능동적으로 다루며 스스로 문제를 해결하는 Tool LLM**

- 주어진 문제에 맞춰 필요한 도구를 직접 선택·호출, 결과를 다시 추론에 반영해 다음 행동 결정
- "사고(reasoning) → 행동(acting) → 관찰(observation)"을 반복하며 정적인 지식으로는 풀 수 없던 작업을 해결



Tool LLM

❖ 도구를 능동적으로 다루며 스스로 문제를 해결하는 Tool LLM

ReAct

추론과 행동을 결합하여 도구 사용의 근거 마련

REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

Shunyu Yao[*,1], Jeffrey Zhao[2], Dian Yu[2], Nan Du[2], Izhak Shafran[2], Karthik Narasimhan[1], Yuan Cao[2]

[1]Department of Computer Science, Princeton University
[2]Google Research, Brain team
[1]{shunyuy,karthikn}@princeton.edu
[2]{jeffreyzhao,dianyu,dunan,izhak,yuancao}@google.com

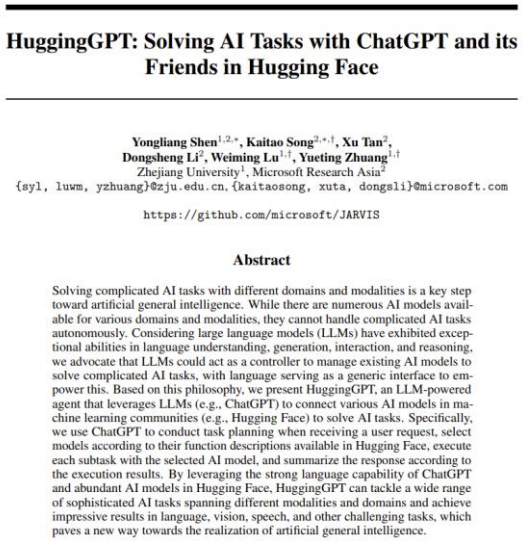
ABSTRACT

While large language models (LLMs) have demonstrated impressive performance across tasks in language understanding and interactive decision making, their abilities for reasoning (e.g. chain-of-thought prompting) and acting (e.g. action plan generation) have primarily been studied as separate topics. In this paper, we explore the use of LLMs to generate both reasoning traces and task-specific actions in an interleaved manner, allowing for greater synergy between the two: reasoning traces help the model induce, track, and update action plans as well as handle exceptions, while actions allow it to interface with and gather additional information from external sources such as knowledge bases or environments. We apply our approach, named ReAct, to a diverse set of language and decision making tasks and demonstrate its effectiveness over state-of-the-art baselines in addition to improved human interpretability and trustworthiness. Concretely, on question answering (HotpotQA) and fact verification (Fever), ReAct overcomes prevalent issues of hallucination and error propagation in chain-of-thought reasoning by interacting with a simple Wikipedia API, and generating human-like task-solving trajectories that are more interpretable than baselines without reasoning traces. Furthermore, on two interactive decision making benchmarks (ALFWorld and WebShop), ReAct outperforms imitation and reinforcement learning methods by an absolute success rate of 34% and 10% respectively, while being prompted with only one or two in-context examples.

2023 ICLR

HuggingGPT

LLM을 컨트롤러 삼아 여러 도구를 조율



HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face

Yongliang Shen[1,2,*], Kaitao Song[2,*,†], Xu Tan[2],
Dongsheng Li[2], Weiming Lu[1,†], Yueting Zhuang[1,†]
Zhejiang University[1], Microsoft Research Asia[2]
{syl, luwm, yzhuang}@zju.edu.cn, {kaitaosong, xuta, dongsli}@microsoft.com

https://github.com/microsoft/JARVIS

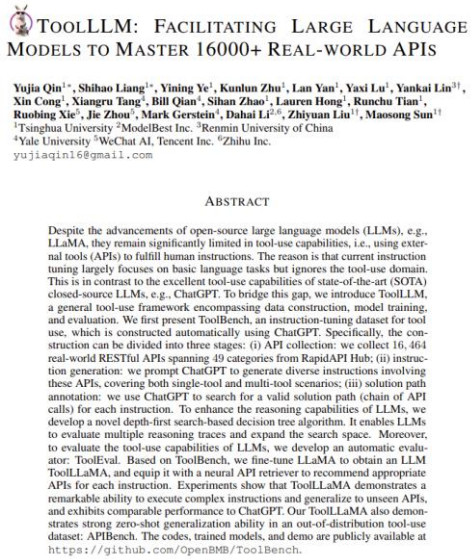
Abstract

Solving complicated AI tasks with different domains and modalities is a key step toward artificial general intelligence. While there are numerous AI models available for various domains and modalities, they cannot handle complicated AI tasks autonomously. Considering large language models (LLMs) have exhibited exceptional abilities in language understanding, generation, interaction, and reasoning, we advocate that LLMs could act as a controller to manage existing AI models to solve complicated AI tasks, with language serving as a generic interface to empower this. Based on this philosophy, we present HuggingGPT, an LLM-powered agent that leverages LLMs (e.g., ChatGPT) to connect various AI models in machine learning communities (e.g., Hugging Face) to solve AI tasks. Specifically, we use ChatGPT to conduct task planning when receiving a user request, select models according to their function descriptions available in Hugging Face, execute each subtask with the selected AI model, and summarize the response according to the execution results. By leveraging the strong language capability of ChatGPT and abundant AI models in Hugging Face, HuggingGPT can tackle a wide range of sophisticated AI tasks spanning different modalities and domains and achieve impressive results in language, vision, speech, and other challenging tasks, which paves a new way towards the realization of artificial general intelligence.

2023 NeurIPS

ToolLLM

실세계 API 확장 및 도구 사용 학습



TOOLLLM: FACILITATING LARGE LANGUAGE MODELS TO MASTER 16000+ REAL-WORLD APIS

Yujia Qin[1*], Shihao Liang[1*], Yining Ye[1], Kunlun Zhu[1], Lan Yan[1], Yaxi Lu[1], Yankai Lin[3†], Xin Cong[1], Xiangru Tang[4], Bill Qian[4], Sihan Zhao[1], Lauren Hong[1], Runchu Tian[1], Ruobing Xie[5], Jie Zhou[5], Mark Gerstein[4], Dahai Li[2,6], Zhiyuan Liu[1†], Maosong Sun[1†]
[1]Tsinghua University [2]ModelBest Inc. [3]Renmin University of China
[4]Yale University [5]WeChat AI, Tencent Inc. [6]Zhihu Inc.
yujiaqin16@gmail.com

ABSTRACT

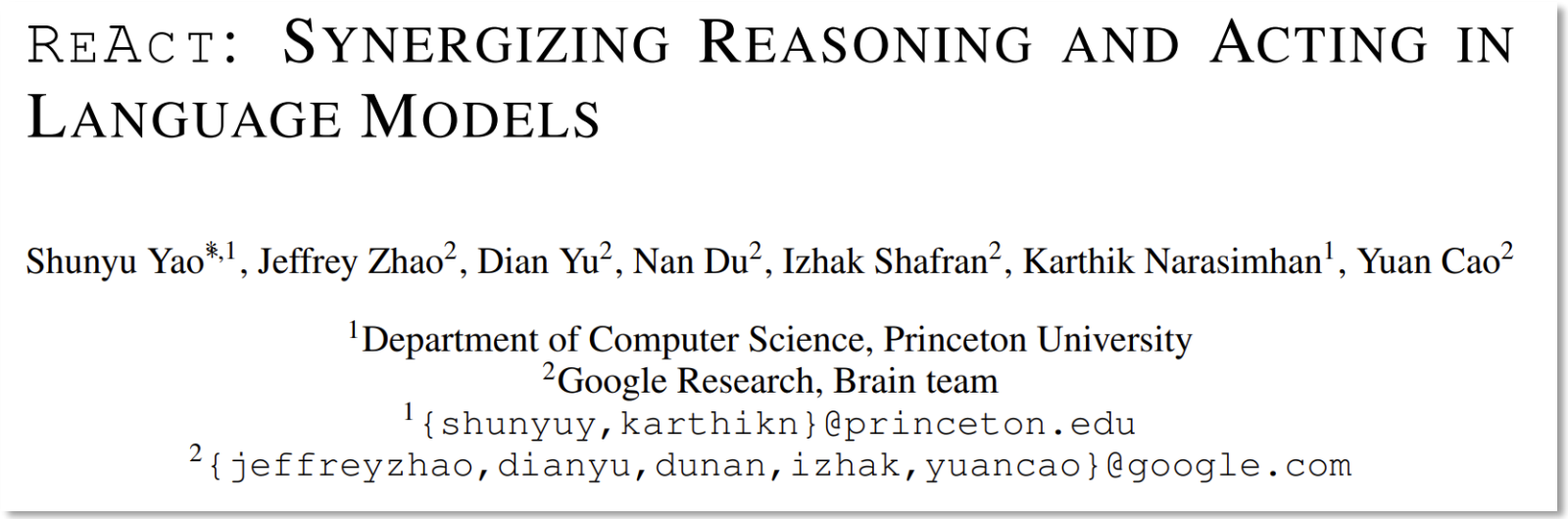
Despite the advancements of open-source large language models (LLMs), e.g., LLaMA, they remain significantly limited in tool-use capabilities, i.e., using external tools (APIs) to fulfill human instructions. The reason is that current instruction tuning largely focuses on basic language tasks but ignores the tool-use domain. This is in contrast to the excellent tool-use capabilities of state-of-the-art (SOTA) closed-source LLMs, e.g., ChatGPT. To bridge this gap, we introduce ToolLLM, a general tool-use framework encompassing data construction, model training, and evaluation. We first present ToolBench, an instruction-tuning dataset for tool use, which is constructed automatically using ChatGPT. Specifically, the construction can be divided into three stages: (i) API collection: we collect 16,464 real-world RESTful APIs spanning 49 categories from RapidAPI Hub; (ii) instruction generation: we prompt ChatGPT to generate diverse instructions involving these APIs, covering both single-tool and multi-tool scenarios; (iii) solution path annotation: we use ChatGPT to search for a valid solution path (chain of API calls) for each instruction. To enhance the reasoning capabilities of LLMs, we develop a novel depth-first search-based decision tree algorithm. It enables LLMs to evaluate multiple reasoning traces and expand the search space. Moreover, to evaluate the tool-use capabilities of LLMs, we develop an automatic evaluator: ToolEval. Based on ToolBench, we fine-tune LLaMA to obtain an LLM ToolLLaMA, and equip it with a neural API retriever to recommend appropriate APIs for each instruction. Experiments show that ToolLLaMA demonstrates a remarkable ability to execute complex instructions and generalize to unseen APIs, and exhibits comparable performance to ChatGPT. Our ToolLLaMA also demonstrates strong zero-shot generalization ability in an out-of-distribution tool-use dataset: APIBench. The codes, trained models, and demo are publicly available at https://github.com/OpenBMB/ToolBench.

2024 ICLR

ReAct

❖ **ReAct: Synergizing Reasoning and Acting in Language Models (2023 ICLR)**

- 2023년 ICLR, 인용수 12559회
- 생각(Reasoning)과 행동(Acting)을 반복하며 이를 서로 보완하는 tool-use 에이전트의 대표적 초기 논문



REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

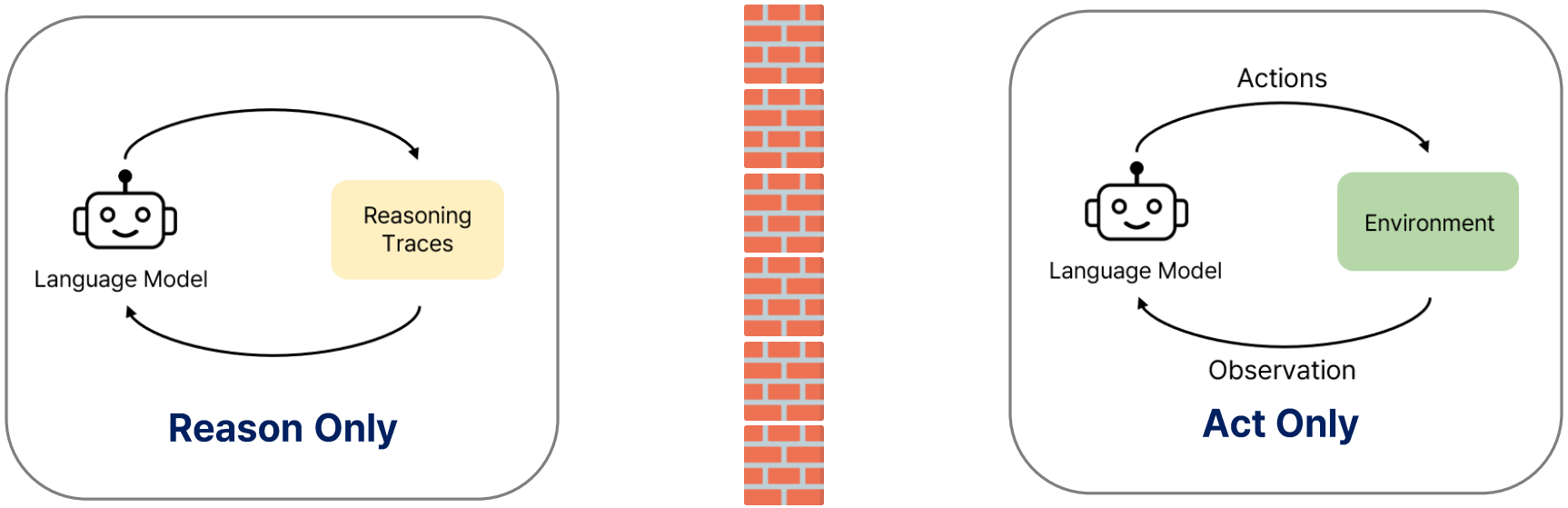
Shunyu Yao[*,1], Jeffrey Zhao[2], Dian Yu[2], Nan Du[2], Izhak Shafran[2], Karthik Narasimhan[1], Yuan Cao[2]

[1]Department of Computer Science, Princeton University
[2]Google Research, Brain team
[1]{shunyuy,karthikn}@princeton.edu
[2]{jeffreyzhao,dianyu,dunan,izhak,yuancao}@google.com

Background

❖ Reasoning과 Acting의 분리

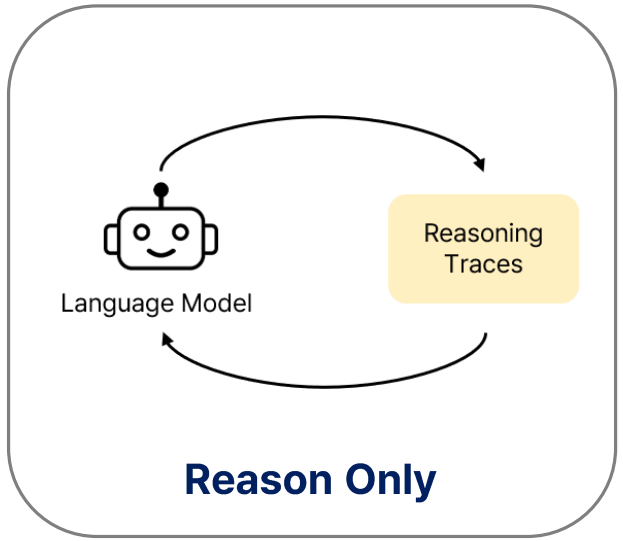
- 추론만 하는 LLM(CoT) – 외부와 단절된 채 생각만..
- 행동만 하는 LLM(planning, acting) – 왜 하는지 모른 채 행동만..

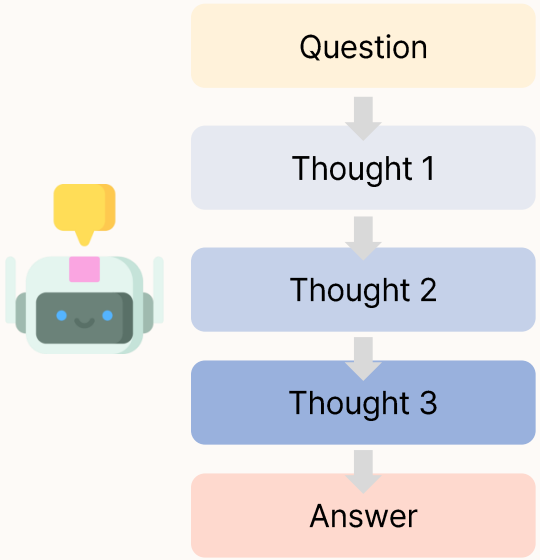


Background

❖ Reasoning과 Acting의 분리

- 추론만 하는 LLM(CoT) – 외부와 단절된 채 생각만..
- 행동만 하는 LLM(planning, acting) – 왜 하는지 모른 채 행동만..





Chain-of-Thought

모델 내부 기억만으로 추론

외부 세계와 소통 X

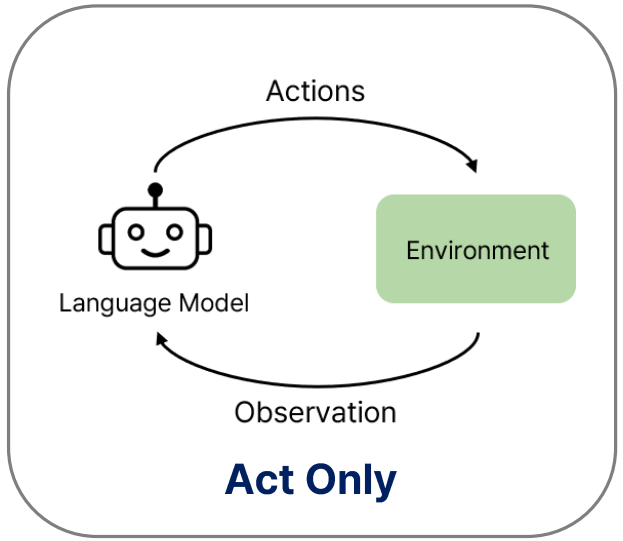
실시간 지식 업데이트 X

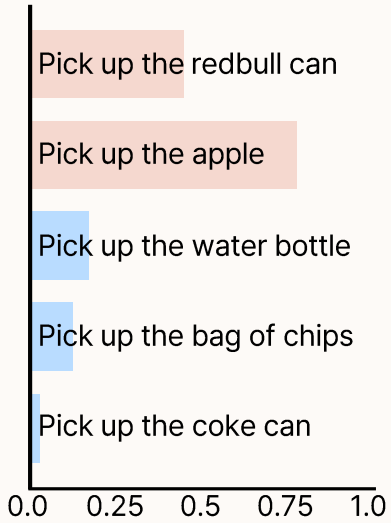
환각 (hallucination)
&
오류 전파 (error propagation)

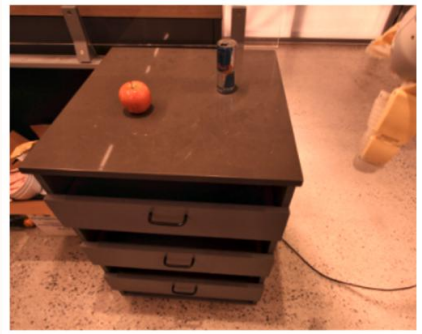
Background

❖ Reasoning과 Acting의 분리

- 추론만 하는 LLM(CoT) – 외부와 단절된 채 생각만..
- 행동만 하는 LLM(planning, acting) – 왜 하는지 모른 채 행동만..







언어 사전 지식에 의존

단순 다음 행동 예측

작업 기억 유지 X

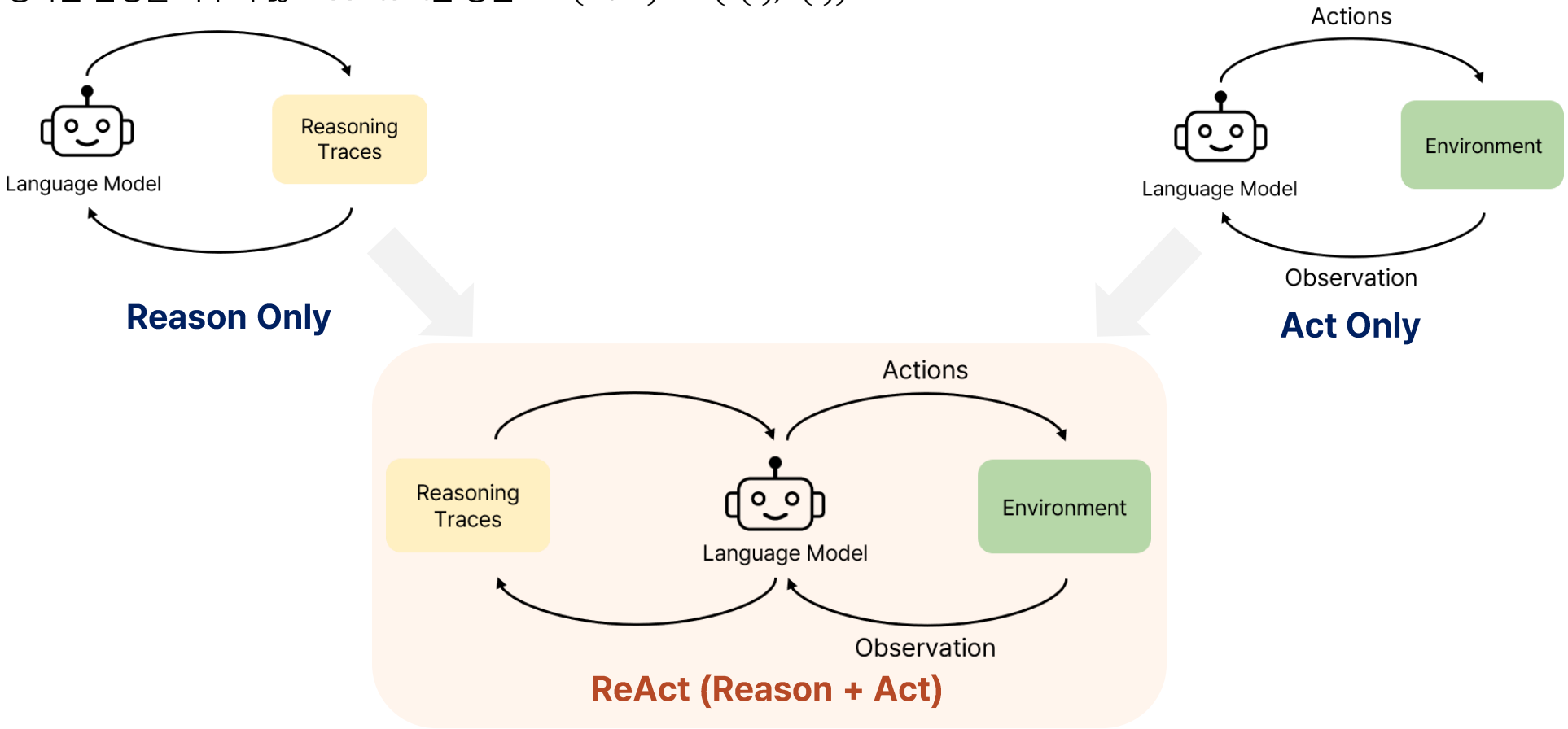
Planning & Acting

Ahn, Michael, et al. "Do as i can, not as i say: Grounding language in robotic affordances." *arXiv preprint arXiv:2204.01691* (2022).

Method

❖ **행동 공간을 언어까지 확장**

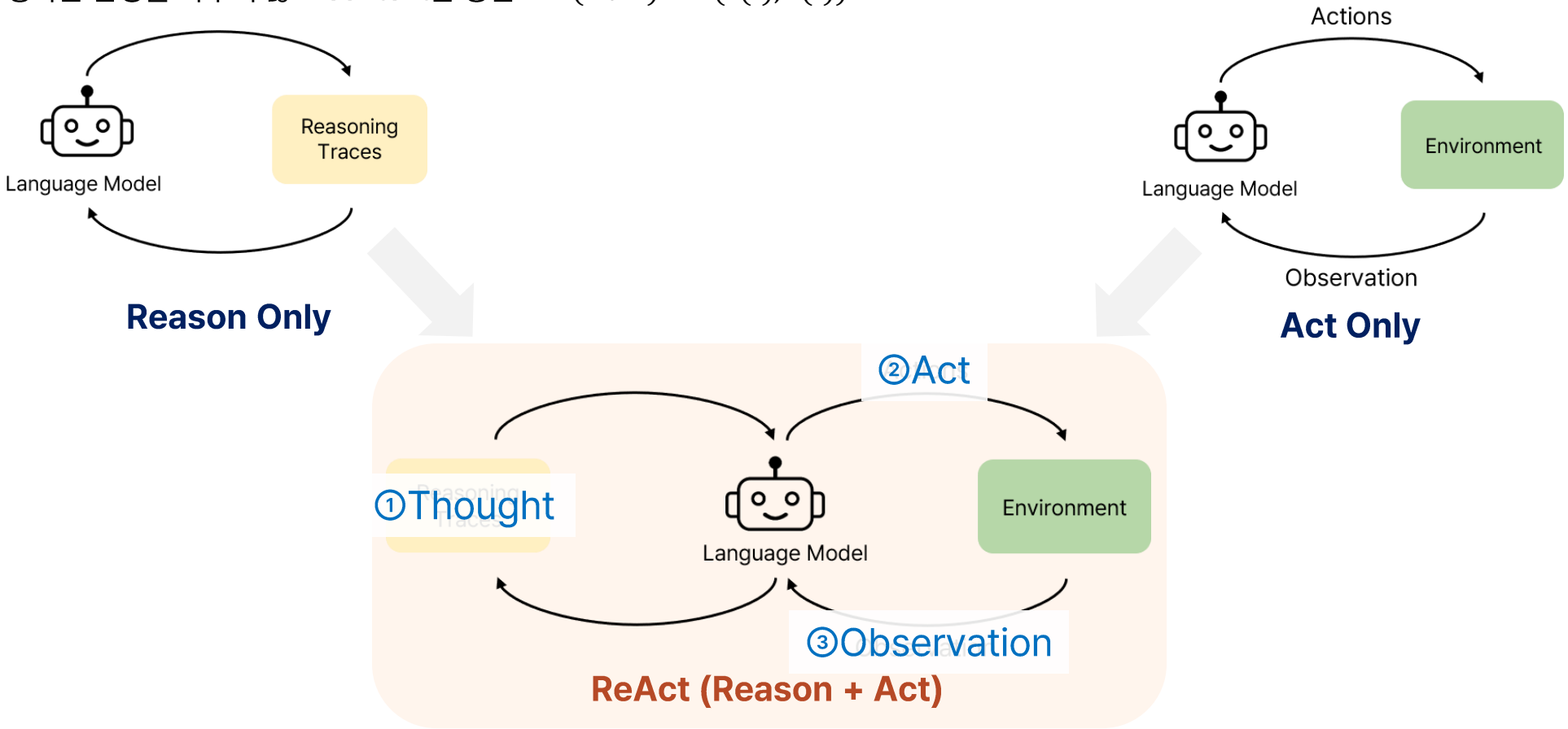
- $\hat{A} = A \cup L$: 기존 행동(A)에 언어(L)를 더해, "생각"도 하나의 행동으로 취급
- 생각은 환경을 바꾸지 않고 context만 갱신 → $c(t+1) = (c(t), \hat{a}(t))$



Method

❖ **행동 공간을 언어까지 확장**

- $\hat{A} = A \cup L$: 기존 행동(A)에 언어(L)를 더해, "생각"도 하나의 행동으로 취급
- 생각은 환경을 바꾸지 않고 context만 갱신 → $c(t+1) = (c(t), \hat{a}(t))$



Method

❖ 행동 공간을 언어까지 확장

- 별도 학습이나 복잡한 프롬프트 없이 예시(Thought → Action → Observation)만 전달 (in-context learning)

  → 직관적이며 쉽고 유연한 설계

<Input Prompt>

ReAct

Question: What is the elevation range for the area that the eastern sector of the Colorado orogeny extends into? Colorado orogeny의 동쪽 지역이 뻗어 있는 지역의 고도 범위는 얼마인가?

Thought 1: I need to search Colorado orogeny, find the area that the eastern sector of the Colorado orogeny extends into, then find the elevation range of the area. 문제 분해

Action 1: Search[Colorado orogeny] 1차 검색

Observation 1: The Colorado orogeny was an episode of mountain building (an orogeny) in Colorado and surrounding areas. 검색 결과 확인

Thought 2: It does not mention the eastern sector. So I need to look up eastern sector. 검색 전략 수정

Action 2: Lookup[eastern sector] 세부 정보 탐색

Observation 2: (Result 1 / 1) The eastern sector extends into the High Plains and is called the Central Plains orogeny. 중간 단서 발견

Thought 3: The eastern sector of Colorado orogeny extends into the High Plains. So I need to search High Plains and find its elevation range. 새로운 하위 문제 설정

Action 3: Search[High Plains] 후속 검색

Observation 3: High Plains refers to one of two distinct land regions 검색 결과의 모호성 확인

Thought 4: I need to instead search High Plains (United States). 검색어 구체화

Action 4: Search[High Plains (United States)] 정교화된 검색

⋮

Question

Experiments

❖ Settings

- 추론 중심과 행동 중심의 데이터셋으로 구분지어 실험
- 베이스라인: Standard(사고·행동·관측 전부 제거), CoT(행동·관측 제거), CoT-SC(CoT 다수결), Act(사고만 제거)
  - ALFWorld: BUTLER (과제 유형당 10만 개로 훈련한 IL)
  - WebShop: IL(Imitation Learning), IL+RL(Reinforce Learning), Human Expert

	데이터셋	과제	지표
추론 중심	HotpotQA	멀티홉 QA	EM
	FEVER	사실 검증	Acc
행동 중심	ALFWorld	텍스트 기반 가정 환경	Success Rate
	WebShop	실제 온라인 쇼핑 상품	Score/SR

Dataset

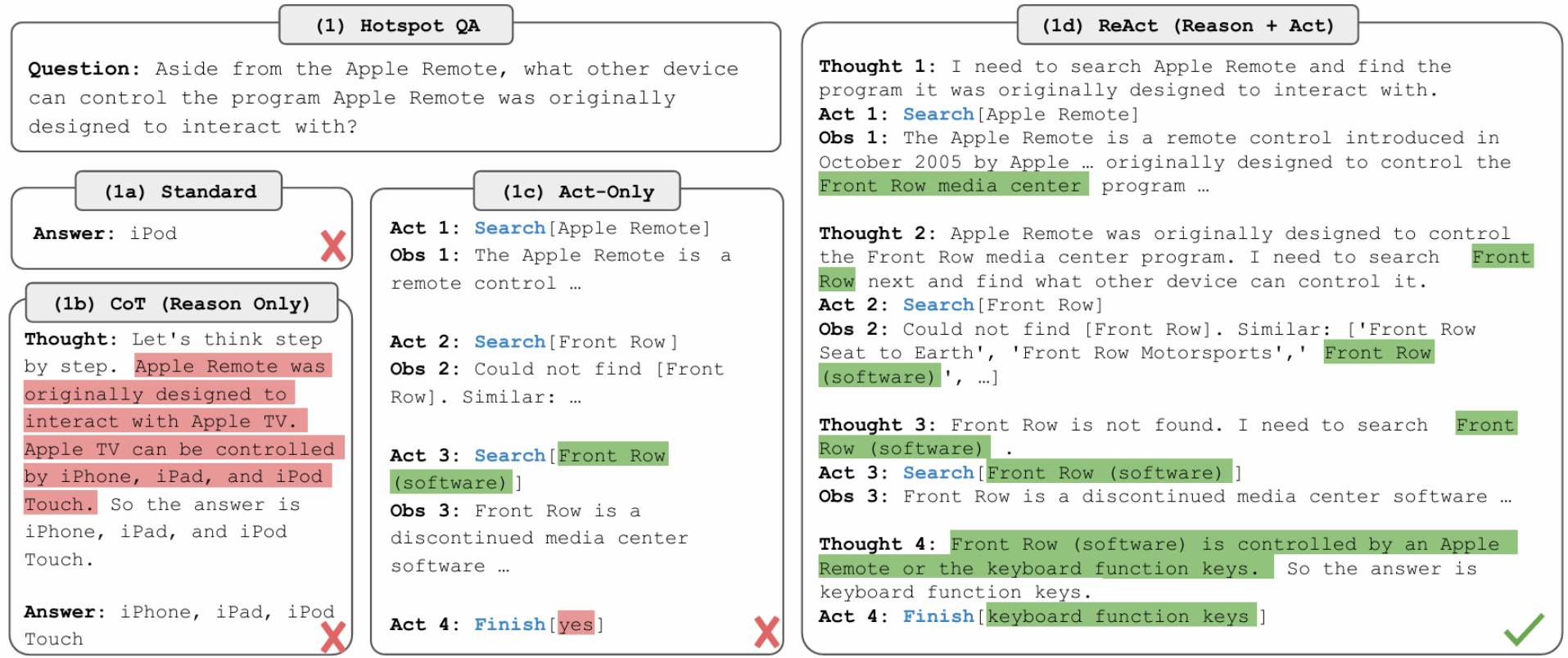
방법	Thought	Action	Observation
Standard	X	X	X
CoT	V	X	X
Act	X	V	V
ReAct	V	V	V

Baseline

Experiments

❖ Qualitative Results

- CoT는 내부 지식만으로 잘못 추론하고 Act-Only는 검색 결과를 해석하지 못함
- ReAct는 검색 결과를 바탕으로 추론과 행동을 반복 수정해 정답에 도달



Experiments

❖ Qualitative Results

- Act-Only는 관찰 결과를 고려하지 않고 실패한 행동을 반복
- ReAct는 목표를 단계별로 계획하고 관찰에 따라 다음 행동을 조정해 작업을 완료

(2) AlfWorld

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.
Your task is to: Put some pepper shaker on a drawer.

(2a) Act-Only

Act 1: Go to drawer 1
Obs 1: The drawer 1 is closed.

Act 2: Open drawer 1
Obs 2: You open the drawer 1. The drawer 1 is open. In it, you see a dishsponge 2, and a spoon 1.

Act 3: Go to sinkbasin 1
Obs 3: On the sinkbasin 1, you see a dishsponge 3, a spatula 1, and a spoon 2.

Act 4: Take peppershaker 1 from sinkbasin 1
Obs 4: Nothing happens.

Act 5: Take peppershaker 1 from sinkbasin 1
Obs 5: Nothing happens. ✗

(2b) ReAct (Reason + Act)

Act 1: Think[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ...]
Act 2: Go to cabinet 1
Obs 2: On the cabinet 1, you see a vase 2.

(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

Act 6: Go to countertop 3
Obs 6: On the countertop 3, you see a apple 1, a bread 1, ... a pepper shaker 1, and a vase 3.

Act 7: Take pepper shaker 1 from countertop 3
Obs 7: You pick up the pepper shaker 1 from the countertop 3.

Act 8: Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.]
Act 9: Go to drawer 1
Obs 9: Drawer 1 is closed.

Act 10: Open drawer 1
Obs 10: You open Drawer 1 …

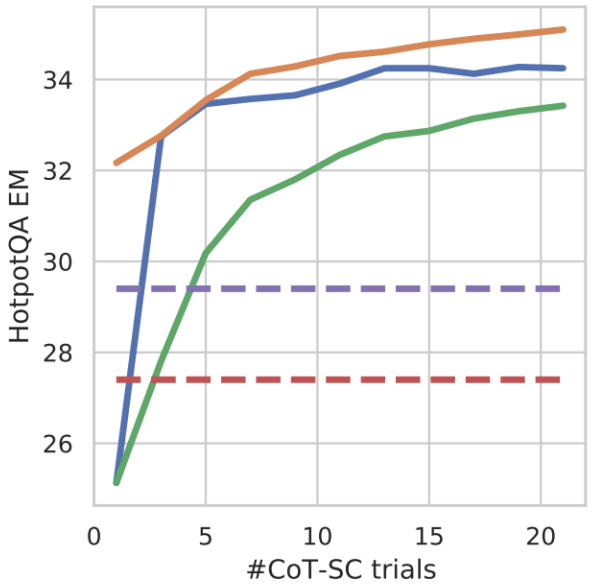
Act 11: Put pepper shaker 1 in/on drawer 1
Obs 11: You put pepper shaker 1 in/on the drawer 1. ✓

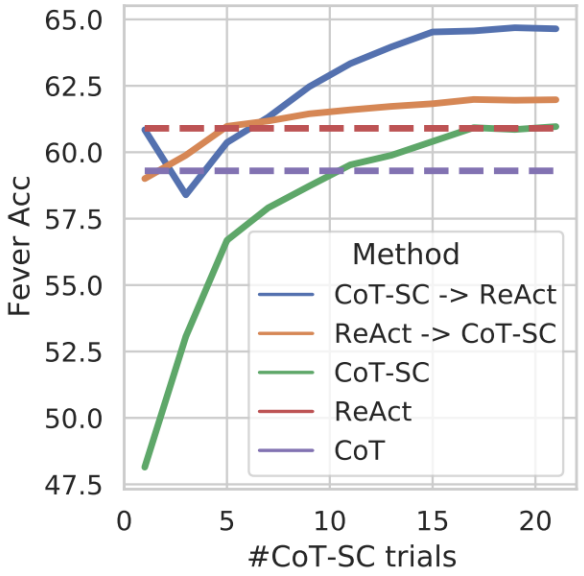
Experiments

❖ Quantitative Results

- ReAct vs Act: 두 과제 모두 ReAct가 Act보다 성능 우위를 보임
- ReAct만 사용할 때보다 CoT-SC(Self-Consistency)를 결합했을 때 가장 높은 성능을 보임
- 두 결합 방식이 CoT-SC가 21개의 샘플로 내는 성능을 단 3~5샘플로 달성 → 더 적은 비용으로 더 좋은 성능

Prompt Method[a]	HotpotQA (EM)	Fever (Acc)
Standard	28.7	57.1
CoT (Wei et al., 2022)	29.4	56.3
CoT-SC (Wang et al., 2022a)	33.4	60.4
Act	25.7	58.9
ReAct	27.4	60.9
CoT-SC → ReAct	34.2	**64.6**
ReAct → CoT-SC	**35.1**	62.0
Supervised SoTA[b]	67.5	89.5





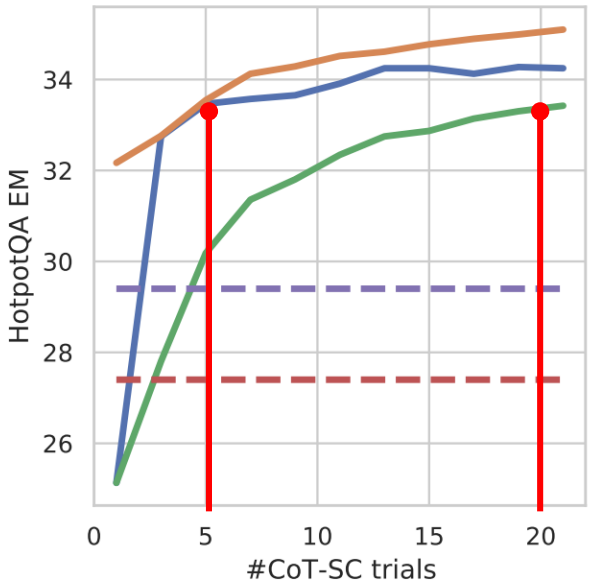
1) ReAct → CoT-SC: 먼저 ReAct로 풀어보고 정해진 단계 안에 답을 못내면 CoT-SC로 전환
2) CoT-SC → ReAct: 먼저 CoT-SC로 n번 풀었을 때 다수결 답이 n/2 미만이라면(답에 확신 없음) ReAct로 전환

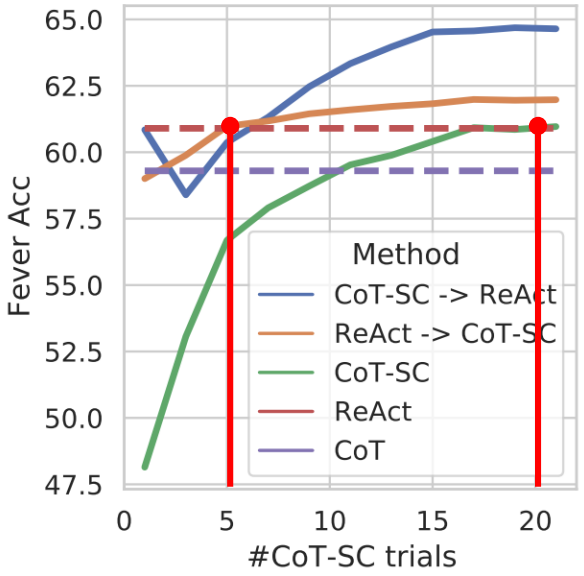
Experiments

❖ Quantitative Results

- ReAct vs Act: 두 과제 모두 ReAct가 Act보다 성능 우위를 보임
- ReAct만 사용할 때보다 CoT-SC를 결합했을 때 가장 높은 성능을 보임
- 두 결합 방식이 CoT-SC가 21개의 샘플로 내는 성능을 단 3~5샘플로 달성 → 더 적은 비용으로 더 좋은 성능

Prompt Method[a]	HotpotQA (EM)	Fever (Acc)
Standard	28.7	57.1
CoT (Wei et al., 2022)	29.4	56.3
CoT-SC (Wang et al., 2022a)	33.4	60.4
Act	25.7	58.9
ReAct	27.4	60.9
CoT-SC → ReAct	34.2	**64.6**
ReAct → CoT-SC	**35.1**	62.0
Supervised SoTA[b]	67.5	89.5





1) ReAct → CoT-SC: 먼저 ReAct로 풀어보고 정해진 단계 안에 답을 못내면 CoT-SC로 전환
2) CoT-SC → ReAct: 먼저 CoT-SC로 n번 풀었을 때 다수결 답이 n/2 미만이라면(답에 확신 없음) ReAct로 전환

Experiments

❖ Quantitative Results

- 행동 중심 과제는 과정이 길고 중간에 잘 하고 있는지 보상이 없기에 더 어려운 태스크
- IM(Inner Monologue): 내적 독백, ReAct 이전의 유사 연구, 에이전트가 "속으로 말하며" 행동하게 한 방식
- ReAct(71)가 Act(45)와, 학습된 BUTLER(37)의 평균 성능을 능가함 → 비교 방법론은 사고 능력이 없어 하위 단계를 구분하지 못함

Method	Pick	Clean	Heat	Cool	Look	Pick 2	All
Act (best of 6)	88	42	74	67	72	**41**	45
ReAct (avg)	65	39	83	76	55	24	57
ReAct (best of 6)	**92**	58	**96**	86	**78**	**41**	**71**
ReAct-IM (avg)	55	59	60	55	23	24	48
ReAct-IM (best of 6)	62	**68**	87	57	39	33	53
BUTLER$_g$ (best of 8)	33	26	70	76	17	12	22
BUTLER (best of 8)	46	39	74	**100**	22	24	37

HuggingGPT

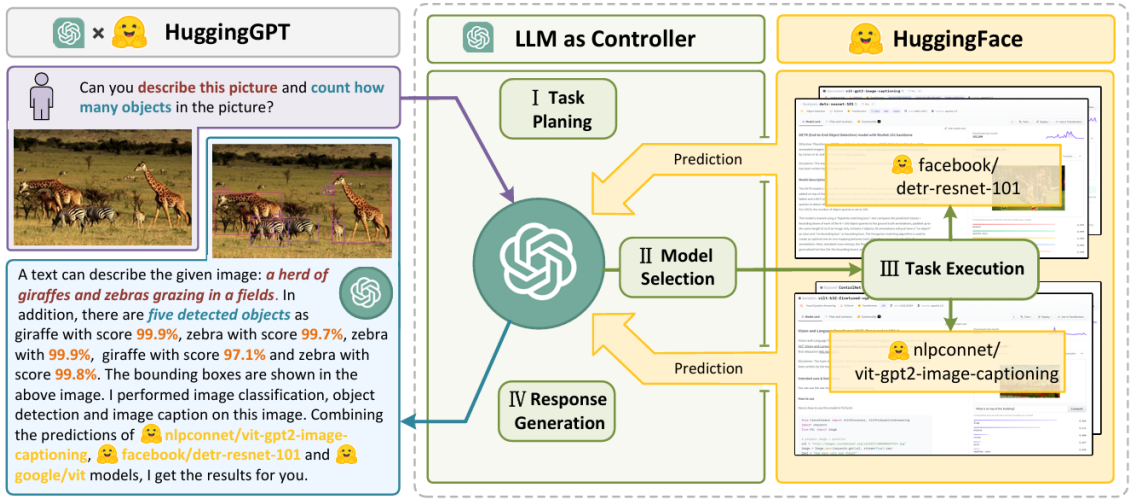
❖ **HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face (2023 NeurIPS)**

- 2023년 NeurIPS, 인용수 2303회
- LLM이 사용자 요청을 위해 태스크를 쪼개고, Hugging Face 모델을 태스크에 맞게 골라 실행하여 최종 결과를 종합

HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face

Yongliang Shen[1,2,*], **Kaitao Song**[2,*,†], **Xu Tan**[2],
Dongsheng Li[2], **Weiming Lu**[1,†], **Yueting Zhuang**[1,†]
Zhejiang University[1], Microsoft Research Asia[2]
{syl, luwm, yzhuang}@zju.edu.cn, {kaitaosong, xuta, dongsli}@microsoft.com

https://github.com/microsoft/JARVIS

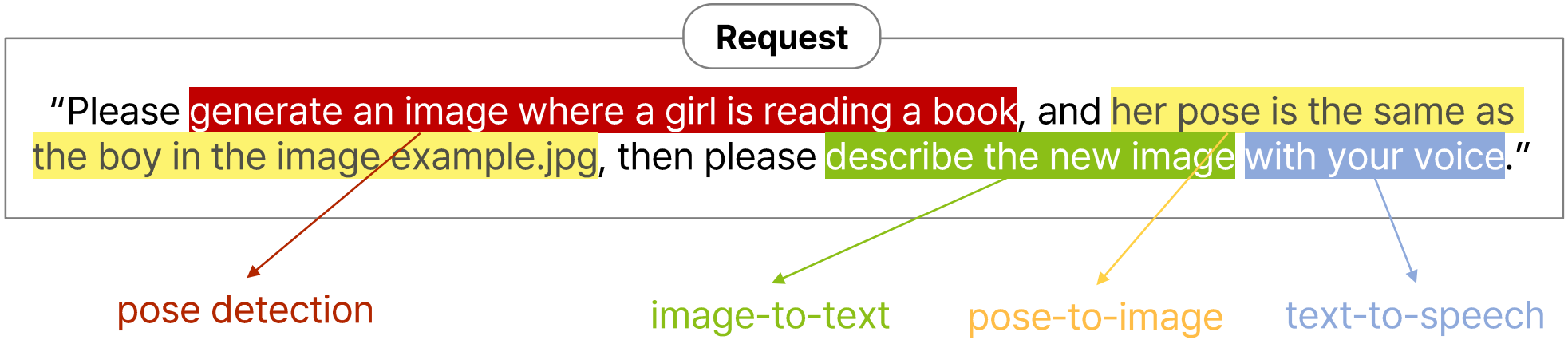


Background

❖ **현실의 복잡한 문제는 여러 하위 태스크로 이루어져 있다**

- 기존 단일 멀티모달 모델은 sub-task 조합으로 확장이 어려움
- 외부 도구 호출하는 LLM은 복잡한 요청을 하위 단위로 분해하고 계획적으로 오케스트레이션하는 능력 부재

→ HuggingFace의 모델 텍스트 설명을 활용하자

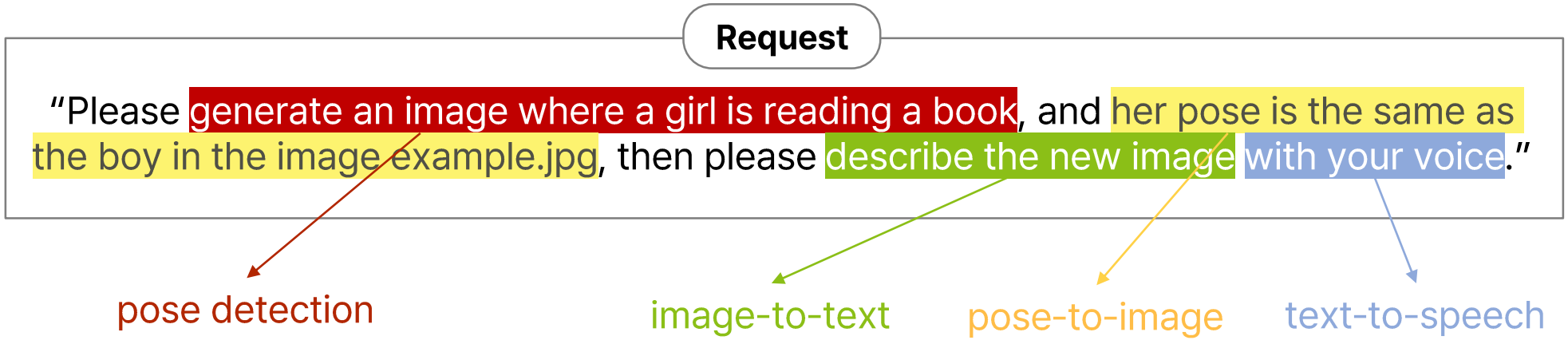


Background

❖ **현실의 복잡한 문제는 여러 하위 태스크로 이루어져 있다**

- 기존 단일 멀티모달 모델은 sub-task 조합으로 확장이 어려움
- 외부 도구 호출하는 LLM은 복잡한 요청을 하위 단위로 분해하고 계획적으로 오케스트레이션하는 능력 부재

→ HuggingFace의 모델 텍스트 설명을 활용하자

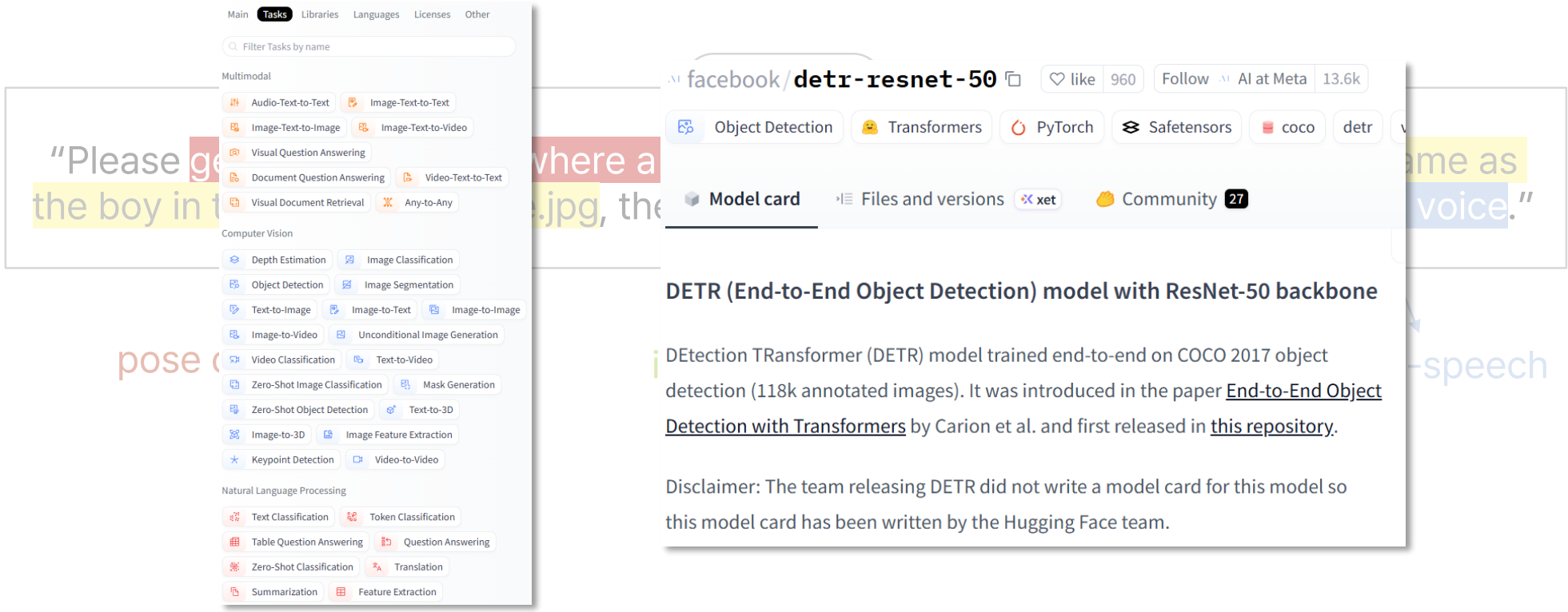


Background

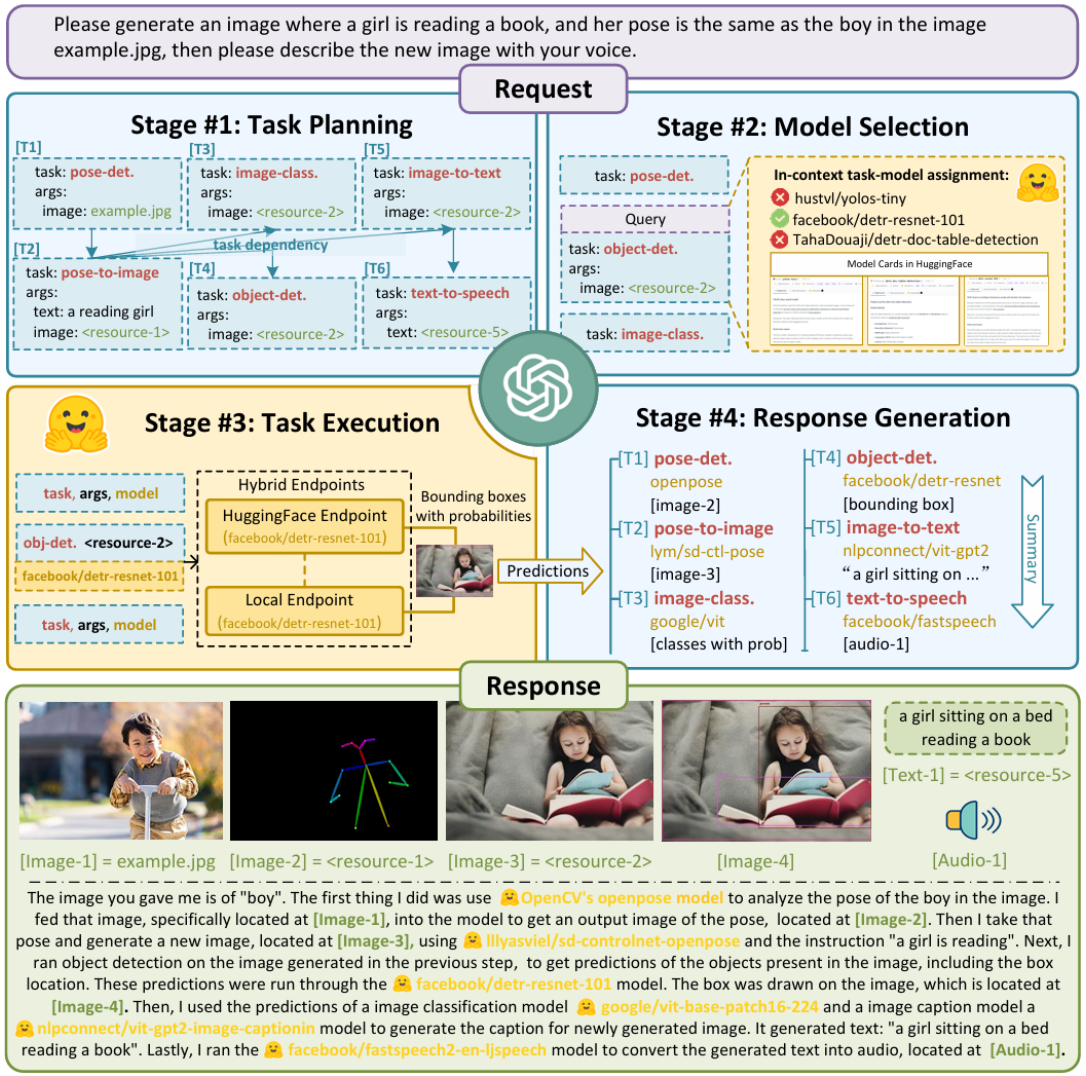
❖ **현실의 복잡한 문제는 여러 하위 태스크로 이루어져 있다**

- 기존 단일 멀티모달 모델은 sub-task 조합으로 확장이 어려움
- 외부 도구 호출하는 LLM은 복잡한 요청을 하위 단위로 분해하고 계획적으로 오케스트레이션하는 능력 부재

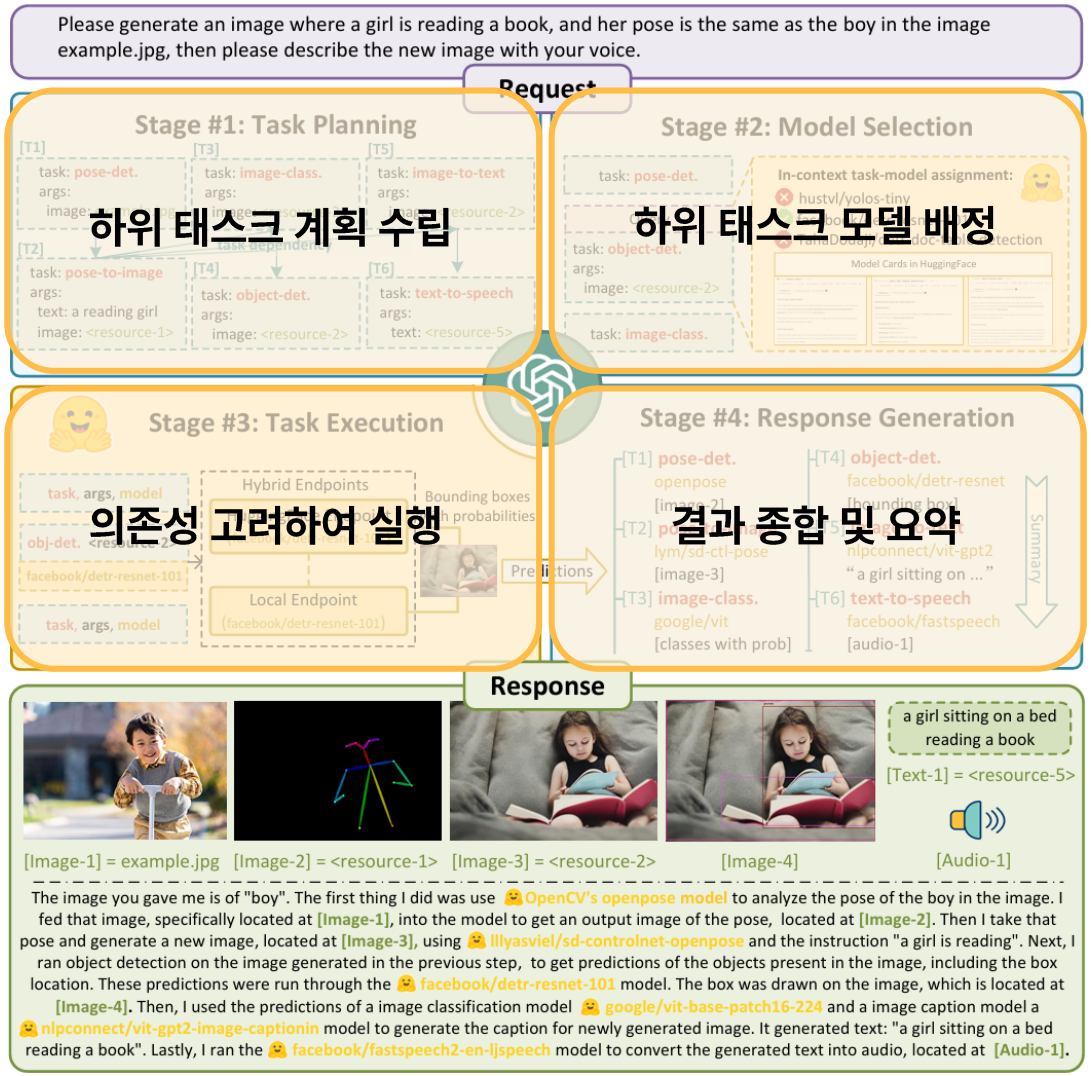
→ HuggingFace의 모델 텍스트 설명을 활용하자



Method


Please generate an image where a girl is reading a book, and her pose is the same as the boy in the image example.jpg, then please describe the new image with your voice.
Request
Stage #1: Task Planning
[T1] task: pose-det. args: image: example.jpg
[T3] task: image-class. args: image: <resource-2>
[T5] task: image-to-text args: image: <resource-2>
task dependency
[T2] task: pose-to-image args: text: a reading girl image: <resource-1>
[T4] task: object-det. args: image: <resource-2>
[T6] task: text-to-speech args: text: <resource-5>
Stage #2: Model Selection
task: pose-det.
Query
task: object-det. args: image: <resource-2>
task: image-class.
In-context task-model assignment:
hustvl/yolos-tiny
facebook/detr-resnet-101
TahaDouaji/detr-doc-table-detection
Model Cards in HuggingFace
Stage #3: Task Execution
task, args, model
obj-det. <resource-2>
facebook/detr-resnet-101
task, args, model
Hybrid Endpoints
HuggingFace Endpoint (facebook/detr-resnet-101)
Local Endpoint (facebook/detr-resnet-101)
Bounding boxes with probabilities
Predictions
Stage #4: Response Generation
[T1] pose-det. openpose [image-2]
[T2] pose-to-image lym/sd-ctl-pose [image-3]
[T3] image-class. google/vit [classes with prob]
[T4] object-det. facebook/detr-resnet [bounding box]
[T5] image-to-text nlpconnect/vit-gpt2 "a girl sitting on ..."
[T6] text-to-speech facebook/fastspeech [audio-1]
Summary
Response
a girl sitting on a bed reading a book
[Text-1] = <resource-5>
[Image-1] = example.jpg
[Image-2] = <resource-1>
[Image-3] = <resource-2>
[Image-4]
[Audio-1]
The image you gave me is of "boy". The first thing I did was use OpenCV's openpose model to analyze the pose of the boy in the image. I fed that image, specifically located at [Image-1], into the model to get an output image of the pose, located at [Image-2]. Then I take that pose and generate a new image, located at [Image-3], using lllyasviel/sd-controlnet-openpose and the instruction "a girl is reading". Next, I ran object detection on the image generated in the previous step, to get predictions of the objects present in the image, including the box location. These predictions were run through the facebook/detr-resnet-101 model. The box was drawn on the image, which is located at [Image-4]. Then, I used the predictions of a image classification model google/vit-base-patch16-224 and a image caption model a nlpconnect/vit-gpt2-image-captionin model to generate the caption for newly generated image. It generated text: "a girl sitting on a bed reading a book". Lastly, I ran the facebook/fastspeech2-en-ljspeech model to convert the generated text into audio, located at [Audio-1].

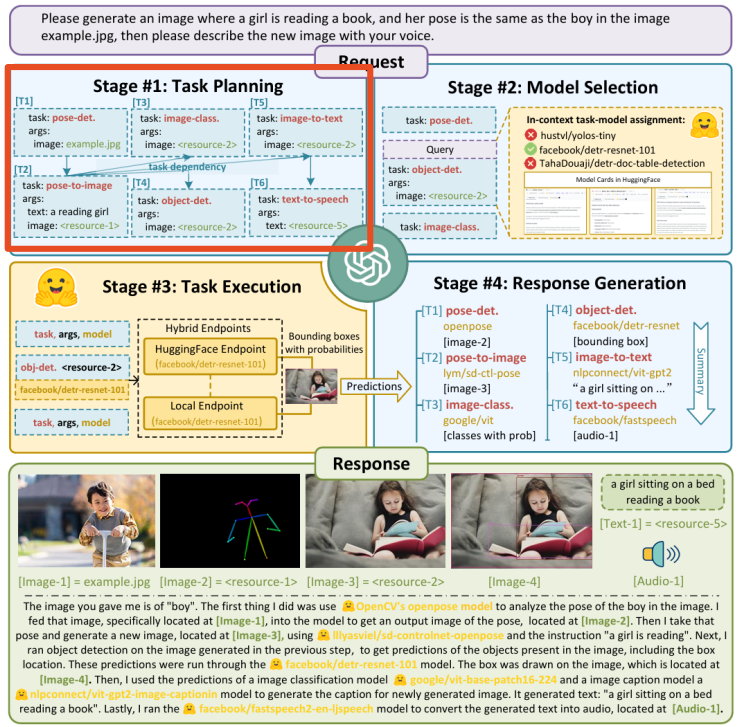
Method


Please generate an image where a girl is reading a book, and her pose is the same as the boy in the image example.jpg, then please describe the new image with your voice.
Request
Stage #1: Task Planning
하위 태스크 계획 수립
Stage #2: Model Selection
하위 태스크 모델 배정
Stage #3: Task Execution
의존성 고려하여 실행
Stage #4: Response Generation
결과 종합 및 요약
Response
a girl sitting on a bed reading a book
[Image-1] = example.jpg
[Image-2] = <resource-1>
[Image-3] = <resource-2>
[Image-4]
[Audio-1]
[Text-1] = <resource-5>
The image you gave me is of "boy". The first thing I did was use OpenCV's openpose model to analyze the pose of the boy in the image. I fed that image, specifically located at [Image-1], into the model to get an output image of the pose, located at [Image-2]. Then I take that pose and generate a new image, located at [Image-3], using lllyasviel/sd-controlnet-openpose and the instruction "a girl is reading". Next, I ran object detection on the image generated in the previous step, to get predictions of the objects present in the image, including the box location. These predictions were run through the facebook/detr-resnet-101 model. The box was drawn on the image, which is located at [Image-4]. Then, I used the predictions of a image classification model google/vit-base-patch16-224 and a image caption model a nlpconnect/vit-gpt2-image-captionin model to generate the caption for newly generated image. It generated text: "a girl sitting on a bed reading a book". Lastly, I ran the facebook/fastspeech2-en-ljspeech model to convert the generated text into audio, located at [Audio-1].

Method

❖ 1. Task Planning: 하위 태스크 계획 수립

- Specification-based Instruction: 사용자 요청을 하위 태스크로 분리
  - 각 태스크를 4개 슬롯의 JSON 템플릿으로 파싱
- Demonstration-based Parsing: 요청 & 파싱 결과 예시를 프롬프트로 입력



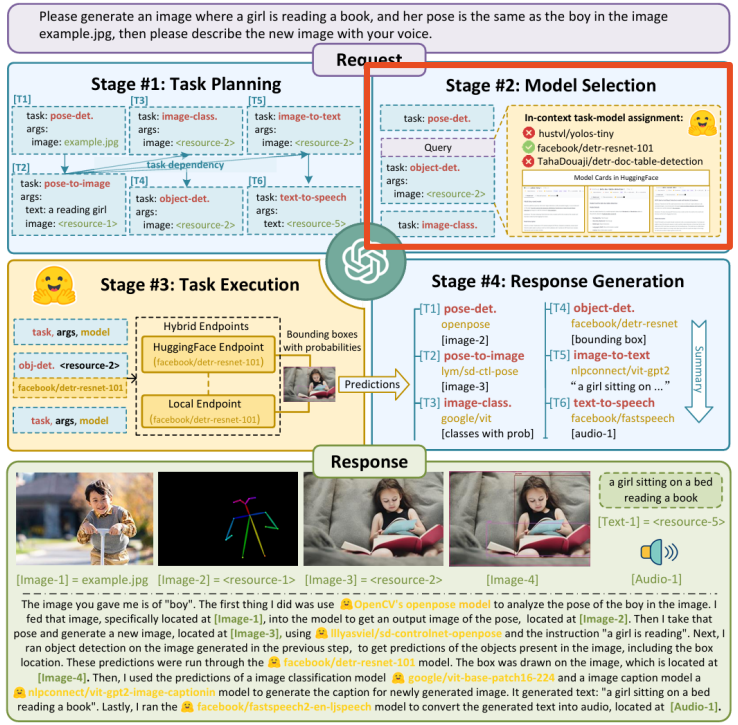
Task Planning	Prompt
	#1 Task Planning Stage - The AI assistant performs task parsing on user input, generating a list of tasks with the following format: [{"task": task, "id", task_id, "dep": dependency_task_ids, "args": {"text": text, "image": URL, "audio": URL, "video": URL}}]. The "dep" field denotes the id of the previous task which generates a new resource upon which the current task relies. The tag "<resource>-task_id" represents the generated text, image, audio, or video from the dependency task with the corresponding task_id. The task must be selected from the following options: {{ *Available Task List* }}. Please note that there exists a logical connections and order between the tasks. In case the user input cannot be parsed, an empty JSON response should be provided. Here are several cases for your reference: {{ *Demonstrations* }}. To assist with task planning, the chat history is available as {{ *Chat Logs* }}, where you can trace the user-mentioned resources and incorporate them into the task planning stage.
	Demonstrations
Can you tell me how many objects in e1.jpg?	[{"task": "object-detection", "id": 0, "dep": [-1], "args": {"image": "e1.jpg" }}]
In e2.jpg, what's the animal and what's it doing?	[{"task": "image-to-text", "id": 0, "dep":[-1], "args": {"image": "e2.jpg" }}, {"task":"image-cls", "id": 1, "dep": [-1], "args": {"image": "e2.jpg" }}, {"task":"object-detection", "id": 2, "dep": [-1], "args": {"image": "e2.jpg" }}, {"task": "visual-quesrion-answering", "id": 3, "dep":[-1], "args": {"text": "what's the animal doing?", "image": "e2.jpg" }}]
First generate a HED image of e3.jpg, then based on the HED image and a text "a girl reading a book", create a new image as a response.	[{"task": "pose-detection", "id": 0, "dep": [-1], "args": {"image": "e3.jpg" }}, {"task": "pose-text-to-image", "id": 1, "dep": [0], "args": {"text": "a girl reading a book", "image": "<resource>-0" }}]

Method

❖ 2. Model Selection: 하위 테스크 모델 배정

- HuggingFace에 작성되어 있는 모델 설명을 기준으로 배정
- 모든 모델의 설명을 입력할 수 없으므로 태스크 타입으로 필터링 → 다운로드 수 기준 Top-k만 후보로 제시

  → 모델 설명만 제공하면 되므로 확장 용이



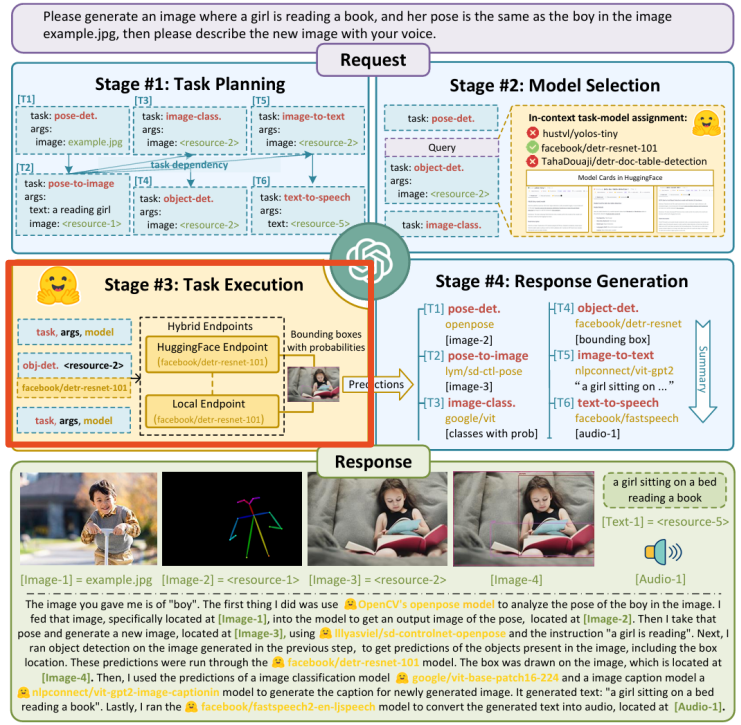
Model Selection	Prompt
	#2 Model Selection Stage - Given the user request and the call command, the AI assistant helps the user to select a suitable model from a list of models to process the user request. The AI assistant merely outputs the model id of the most appropriate model. The output must be in a strict JSON format: {"id": "id", "reason": "your detail reason for the choice"}. We have a list of models for you to choose from {{ *Candidate Models* }}. Please select one model from the list.
	Candidate Models
	{"model_id": model id #1, "metadata": meta-info #1, "description": description of model #1}
	{"model_id": model id #2, "metadata": meta-info #2, "description": description of model #2}
	. . . .
	{"model_id": model id #K, "metadata": meta-info #K, "description": description of model #K}

Method

❖ 3. Task Execution: 의존성 고려하여 실행

- 선행 태스크의 출력이 동적으로 생성 → 실행 직전에 의존 리소스를 지정해야 함
- **<resource>-task_id**: Planning 단계에서 dependency 자리에 심볼 삽입

  → 실행 단계에서 선행 태스크의 실제 결과로 동적 치환



1. Task Planning

```
{"task": "pose-detection", "id": 0, "dep": [-1],
 "args": {"image": "example.jpg"}}
```

```
{"task": "pose-to-image", "id": 1, "dep": [0],
 "args": {"text": "a reading girl", "image": "<resource>-0"}}
```

3. Task Execution

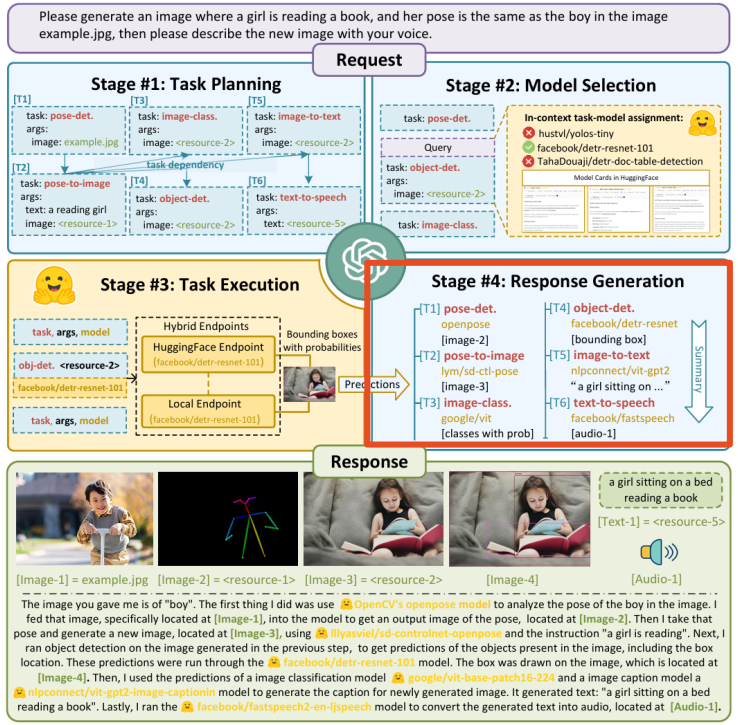
```
{"task": "pose-detection", "id": 0, "dep": [-1],
 "args": {"image": "example.jpg"}}
```

```
{"task": "pose-to-image", "id": 1, "dep": [0],
 "args": {"text": "a reading girl", "image": "/image/pose_2b3f.jpg"}}
```

Method

❖ 4. Response Generation: 최종 응답 생성 및 요약

- 사용자 요청 + 하위 태스크 + 모델 배정 + 추론 결과 ➔ 최종 응답 생성

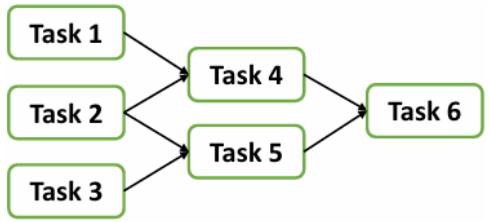


Response Generation	Prompt
	#4 Response Generation Stage - With the input and the inference results, the AI assistant needs to describe the process and results. The previous stages can be formed as - User Input: {{ *User Input* }}, Task Planning: {{ *Tasks* }}, Model Selection: {{ *Model Assignment* }}, Task Execution: {{ *Predictions* }}. You must first answer the user's request in a straightforward manner. Then describe the task process and show your analysis and model inference results to the user in the first person. If inference results contain a file path, must tell the user the complete file path. If there is nothing in the results, please tell me you can't make it.

Experiments

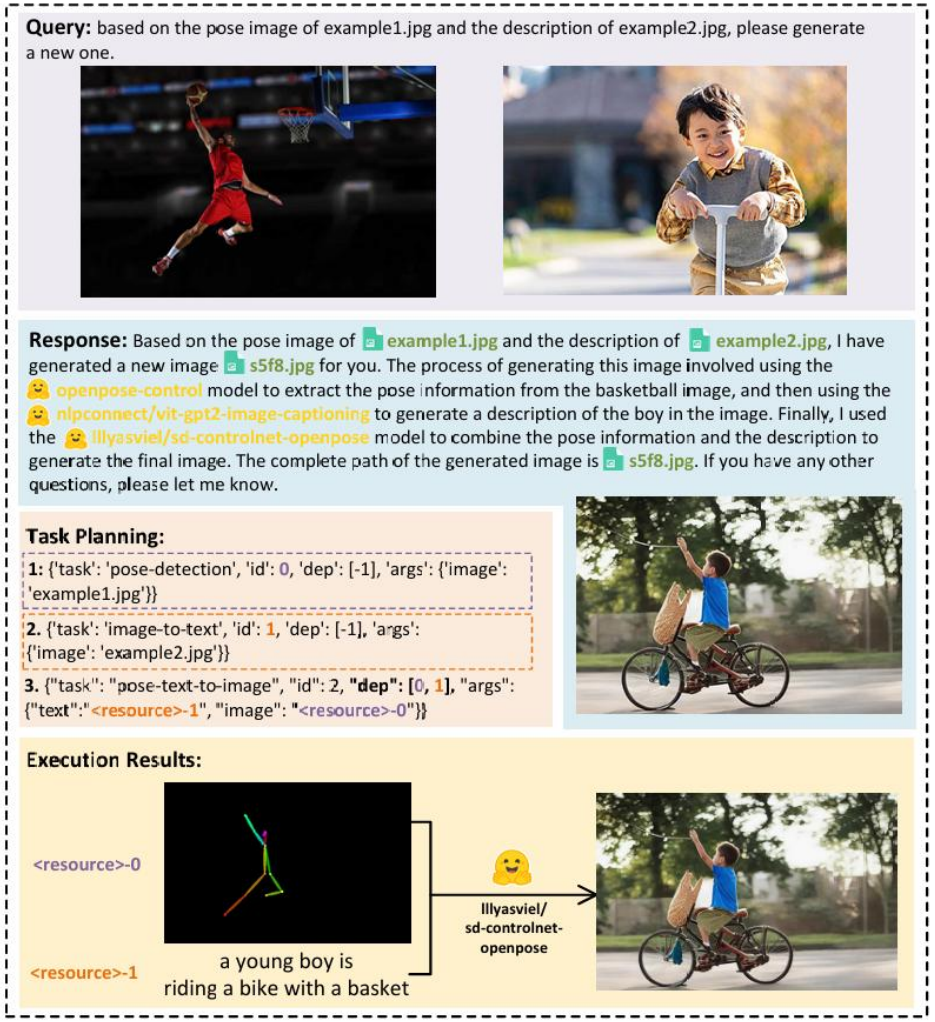
❖ Settings

- Task planning 능력만을 평가 - 전체 파이프라인에서 어떤 태스크가 실행될지 결정하는 가장 중요한 단계
- **Single Task**: 태스크 한 개 요청 → F1, Accuracy
- **Sequential Task**: 여러 서브 태스크가 순차적으로 분해되는 요청 → F1, ED(한 시퀀스를 다른 시퀀스로 바꾸는 데 필요한 최소 편집 횟수)
- **Graph Task**: 방향성 비순환 그래프로 분해되는 요청 → GPT4를 평가자로 사용
- Dataset: GPT-4로 생성한 Pseudo-label + human-annotated dataset

Task Type	Diagram	Example	Metrics
Single Task		Show me a funny image of a cat	Precision, Recall, F1, Accuracy
Sequential Task		Replace the cat with a dog in example.jpg	Precision, Recall, F1 Edit Distance
Graph Task		Given a collection of image A: a.jpg, B: b.jpg, C: c.jpg, please tell me which image is more like image B in terms of semantic, A or C?	Precision, Recall, F1 GPT-4 Score

Experiments

❖ Qualitative Results



Experiments

❖ Quantitative Evaluation

- 전반적으로 GPT-3.5가 오픈소스 LLM을 모든 태스크 유형에서 크게 앞섬
- 복잡한 태스크(Graph)일수록 격차가 더 벌어짐

LLM	Acc ↑	Pre ↑	Recall ↑	F1 ↑
Alpaca-7b	6.48	35.60	6.64	4.88
Vicuna-7b	23.86	45.51	26.51	29.44
GPT-3.5	52.62	62.12	52.62	54.45

Single Task

LLM	ED ↓	Pre ↑	Recall ↑	F1 ↑
Alpaca-7b	0.83	22.27	23.35	22.80
Vicuna-7b	0.80	19.15	28.45	22.89
GPT-3.5	0.54	61.09	45.15	51.92

Sequential Task

LLM	GPT-4 Score ↑	Pre ↑	Recall ↑	F1 ↑
Alpaca-7b	13.14	16.18	28.33	20.59
Vicuna-7b	19.17	13.97	28.08	18.66
GPT-3.5	50.48	54.90	49.23	51.91

Graph Task

Experiments

❖ Quantitative Evaluation

- 사람이 직접 라벨링한 고품질, 즉 더 어려운 데이터로 평가
- 모델이 좋을수록 성능 향상
- 하지만, GPT-4도 아직 완벽하지 않음 → 아직까지 인간의 답과 Gap 존재

LLM	Sequential Task		Graph Task	
	Acc ↑	ED ↓	Acc ↑	F1 ↑
Alpaca-7b	0	0.96	4.17	4.17
Vicuna-7b	7.45	0.89	10.12	7.84
GPT-3.5	18.18	0.76	20.83	16.45
GPT-4	41.36	0.61	58.33	49.28

Human-annotated dataset

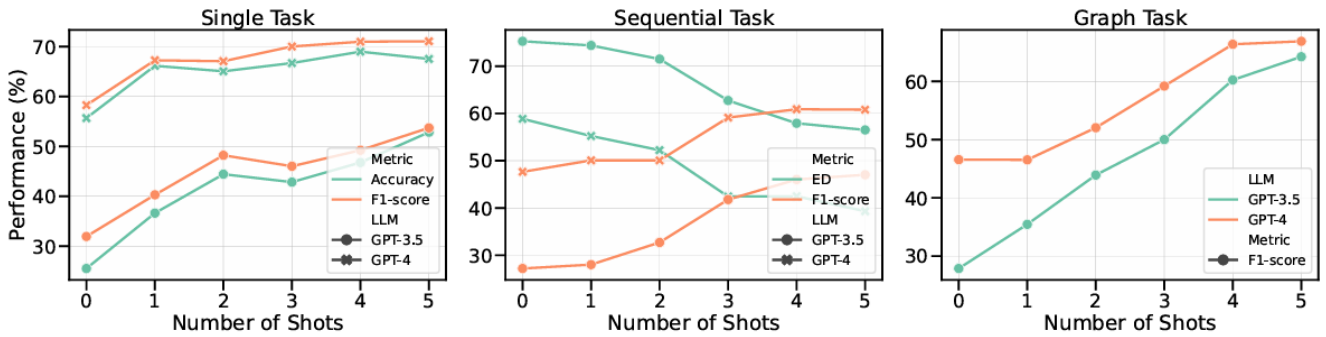
Experiments

❖ Ablation study

- Demonstration(예시) 다양성: 서로 다른 태스크 타입의 종류 수를 늘릴수록 성능 향상
- Demonstration(예시) 개수: 4개에서 성능 포화
- 개수보다 얼마나 다양한 유형을 모델에게 보여줄 건지가 더 중요

Demo Variety (# task types)	LLM	Single Task		Sequencial Task		Graph Task
		Acc ↑	F1 ↑	ED (%) ↓	F1 ↑	F1 ↑
2	GPT-3.5	43.31	48.29	71.27	32.15	43.42
	GPT-4	65.59	67.08	47.17	55.13	53.96
6	GPT-3.5	51.31	51.81	60.81	43.19	58.51
	GPT-4	66.83	68.14	42.20	58.18	64.34
10	GPT-3.5	52.83	53.70	56.52	47.03	64.24
	GPT-4	67.52	71.05	39.32	60.80	66.90

Variety of demonstrations

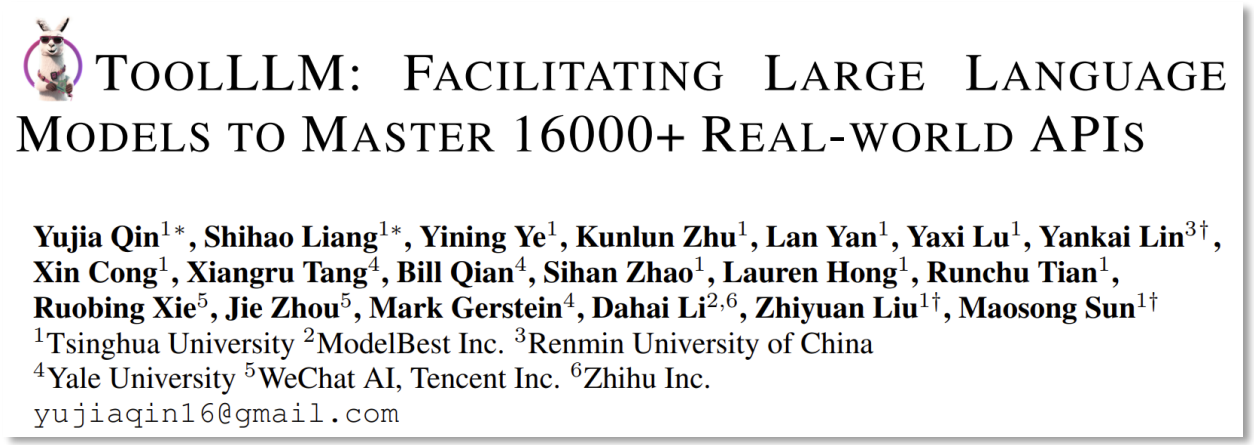


Number of demonstrations

ToolLLM

❖ **ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs (2024 ICLR, spotlight)**

- 2024년 ICLR, 인용수 2005회
- 오픈소스 LLM(LLaMA)가 실제 API를 다룰 수 있도록 데이터셋 구축 및 트리 탐색 기반 추론(DFSDT)으로 ChatGPT급 tool-use 능력 학습



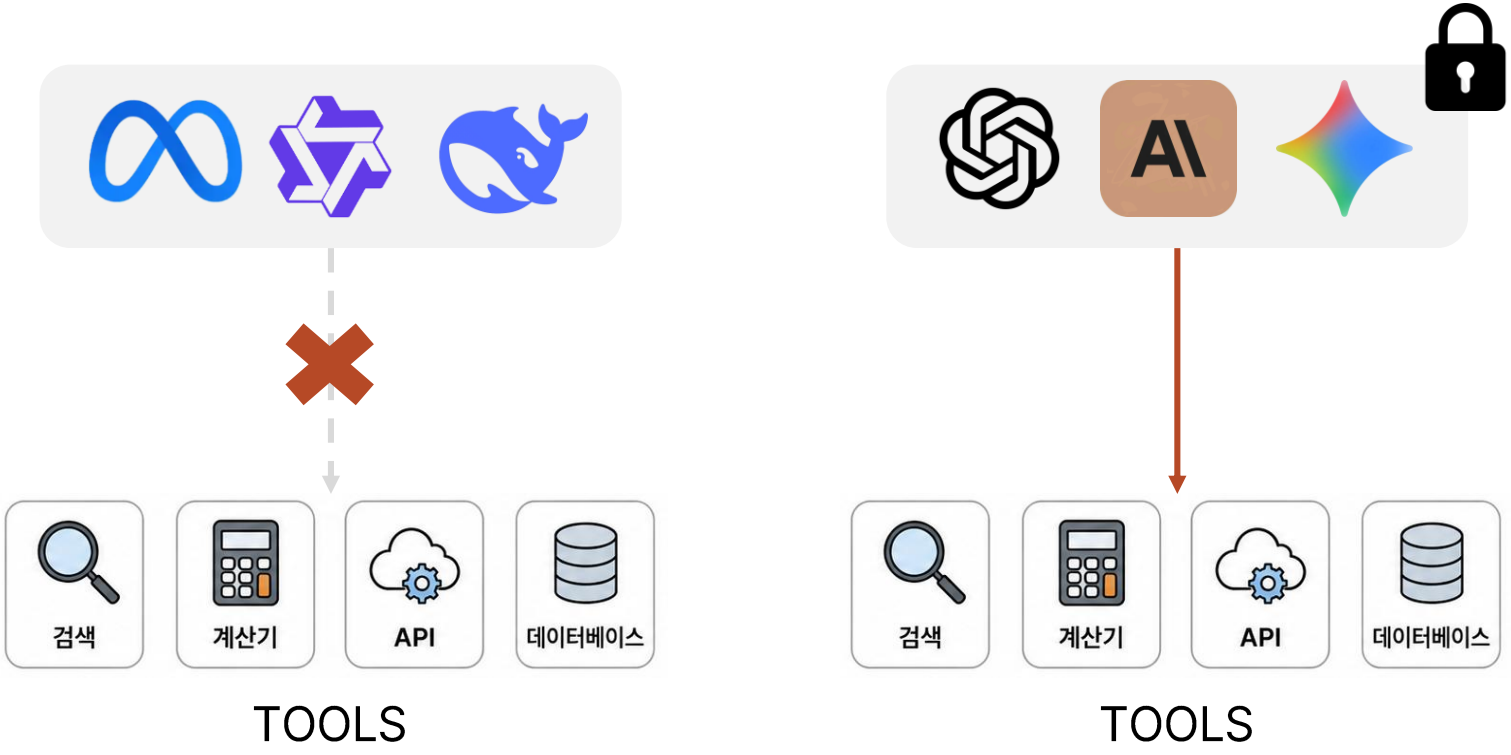
TOOLLLM: FACILITATING LARGE LANGUAGE MODELS TO MASTER 16000+ REAL-WORLD APIS

Yujia Qin[1*], Shihao Liang[1*], Yining Ye[1], Kunlun Zhu[1], Lan Yan[1], Yaxi Lu[1], Yankai Lin[3†], Xin Cong[1], Xiangru Tang[4], Bill Qian[4], Sihan Zhao[1], Lauren Hong[1], Runchu Tian[1], Ruobing Xie[5], Jie Zhou[5], Mark Gerstein[4], Dahai Li[2,6], Zhiyuan Liu[1†], Maosong Sun[1†]
[1]Tsinghua University [2]ModelBest Inc. [3]Renmin University of China
[4]Yale University [5]WeChat AI, Tencent Inc. [6]Zhihu Inc.
yujiaqin16@gmail.com

Background

❖ 오픈소스 LLM은 tool 사용 능력이 부족하다

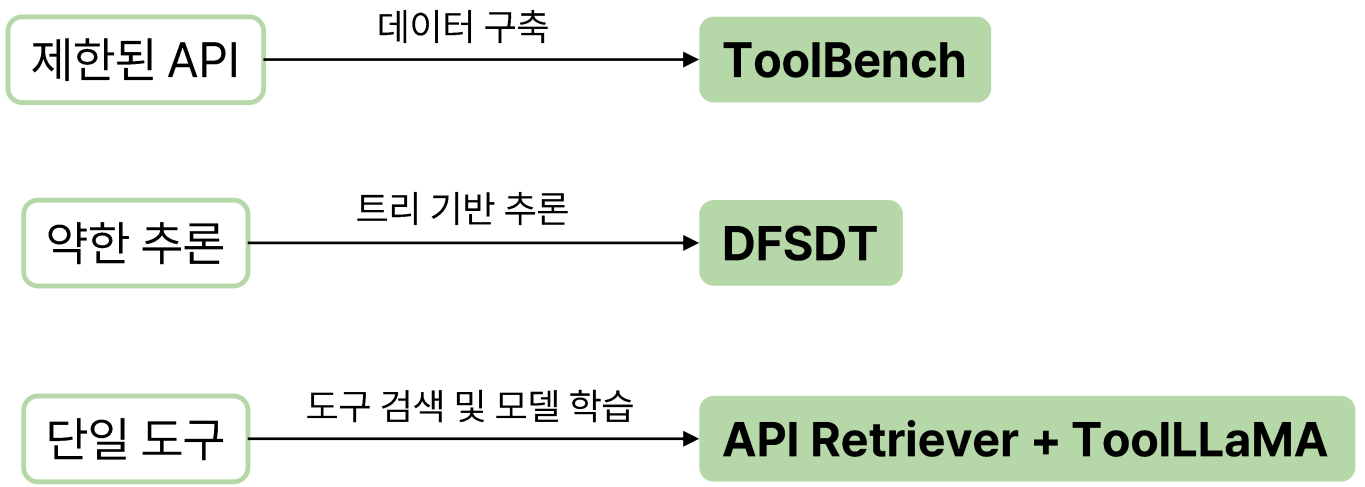
- LLaMA 같은 오픈소스 LLM은 instruction tuning에 집중되어 있어, 도구를 사용하는 복잡한 태스크 해결에는 한계 존재
- ChatGPT 같은 모델은 도구 사용에 뛰어나다고 알려져 있지만 내부 메커니즘 공개 X → 오픈소스 모델을 발전시키기 어려움



Background

❖ 기존 연구의 한계점

- 제한된 API: 실제 API를 적용하지 않고 시뮬레이션에 그치거나, 적용하더라도 다양성 부족 → 폭넓은 도구 사용 X
- 약한 추론: CoT 혹은 ReAct 방식은 한 방향으로 추론하여 에러가 전파되고 복구하지 못함
- 단일 도구·비현실적 가정: 여러 도구를 사용하는 multi-round 태스크를 다루지 못하며, 사용자가 직접 지정해주는 방식은 현실적이지 않음



Method

❖ ToolBench

- 학습에 필요한 대규모 데이터셋을 세 단계(API 수집 → 명령 생성 → 풀이 경로 주석)로 구축
- 전 과정을 ChatGPT로 자동화하여 사람 개입 X

API Collection

Travel

Tool: Flight Search
- API: search_flights (params: from, to, date)

Tool: Hotel Booking
- API: search_hotels (params: city, checkin, checkout)

Weather

Tool: Weather API
- API: get_forecast (params: location, date)

Finance

Tool: Currency Exchange
- API: convert_currency (params: from_cur, to_cur, amount)

RapidAPI 크롤링

Instruction Generation

sampling tools

search_hotels, get_forecast, convert_currency

"이 도구들로 풀 수 있는 명령 생성해"

명령1: "다음 주 파리 날씨 확인하고, 묵을 호텔 찾아줘."
→ API: get_forecast, search_hotels
명령2: "100만원이 유로로 얼마인지 환전 계산하고, 그 예산에 맞는 호텔 찾아줘."
→ API: convert_currency, search_hotels

[명령-APIs] 데이터셋 생성

$\{[\text{Inst}_1, \text{APIs}_1], [\text{Inst}_2, \text{APIs}_2], [\text{Inst}_3, \text{APIs}_3], \ldots\}$

Solution Path Annotation

messages: system 프롬프트 + 명령1
functions: {get_forecast, search_hotels} + Finish 2종

"명령1을 실제로 풀어봐"

[Step 1]
Thought: 먼저 파리 날씨를 확인하자
Act: get_forecast(location="Paris", date="2026-07-16"
Observation: {temp: 24, condition: "sunny"}
[Step 2]
Thought: ...

[명령-대화로그] 데이터셋 생성

$\{[\text{Inst}_1, \text{Path}_1], [\text{Inst}_2, \text{Path}_2], [\text{Inst}_3, \text{Path}_3], \ldots\}$

Method

❖ ToolBench

- 학습에 필요한 대규모 데이터셋을 세 단계(API 수집 → 명령 생성 → 풀이 경로 주석)로 구축
- 전 과정을 ChatGPT로 자동화하여 사람 개입 X

API Collection

Travel

Tool: Flight Search
- API: search_flights (params: from, to, date)

Tool: Hotel Booking
- API: search_hotels (params: city, checkin, checkout)

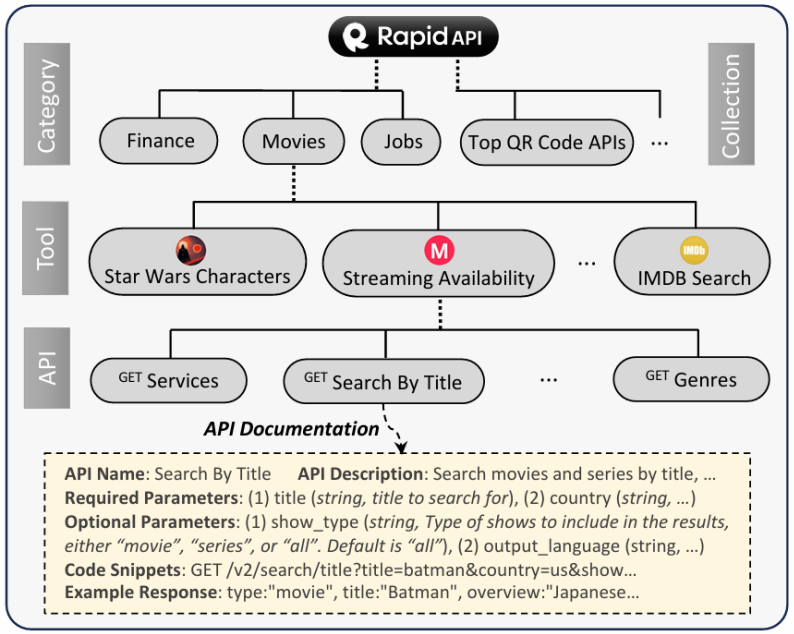
Weather

Tool: Weather API
- API: get_forecast (params: location, date)

Finance

Tool: Currency Exchange
- API: convert_currency (params: from_cur, to_cur, amount)

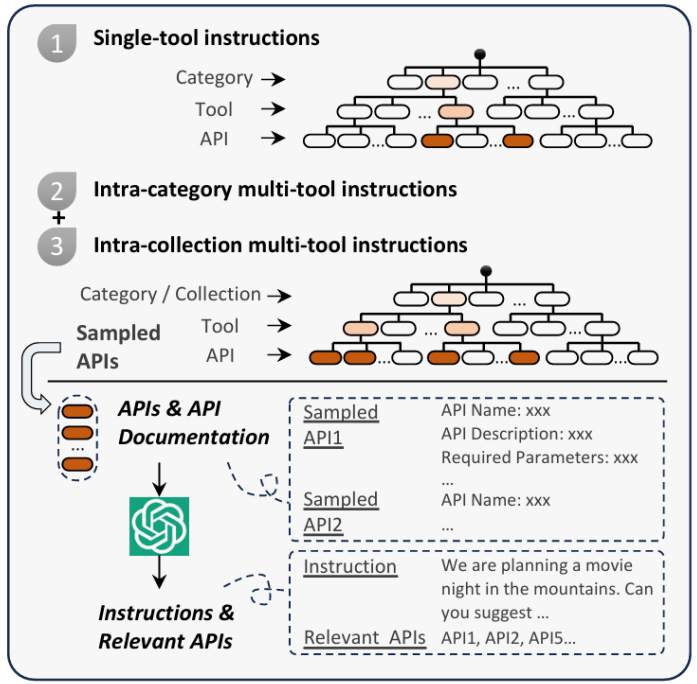
RapidAPI 크롤링



Method

❖ ToolBench

- 학습에 필요한 대규모 데이터셋을 세 단계(API 수집 → 명령 생성 → 풀이 경로 주석)로 구축
- 전 과정을 ChatGPT로 자동화하여 사람 개입 X



Instruction Generation

sampling tools

search_hotels, get_forecast, convert_currency

"이 도구들로 풀 수 있는 명령 생성해"

명령1: "다음 주 파리 날씨 확인하고, 묵을 호텔 찾아줘."
→ API: get_forecast, search_hotels
명령2: "100만원이 유로로 얼마인지 환전 계산하고, 그 예산에 맞는 호텔 찾아줘."
→ API: convert_currency, search_hotels

[명령-APIs] 데이터셋 생성

$\{[\text{Inst}_1, \text{APIs}_1], [\text{Inst}_2, \text{APIs}_2], [\text{Inst}_3, \text{APIs}_3], \dots\}$

Solution Path Annotation

messages: system 프롬프트 + 명령1
functions: {get_forecast, search_hotels} + Finish 2종

"명령1을 실제로 풀어봐"

[Step 1]
Thought: 먼저 파리 날씨를 확인하자
Act: get_forecast(location="Paris", date="2026-07-16"
Observation: {temp: 24, condition: "sunny"}
[Step 2]
Thought: ...

[명령-대화로그] 데이터셋 생성

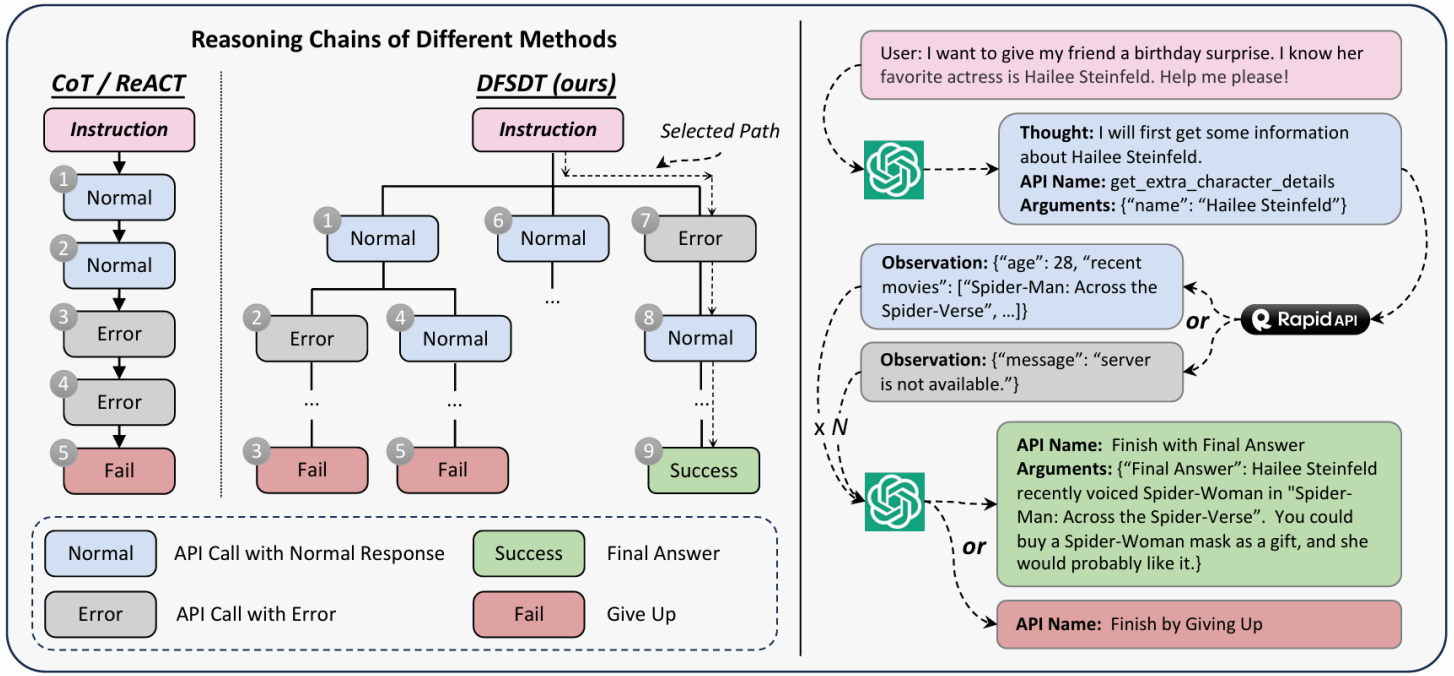
$\{[\text{Inst}_1, \text{Path}_1], [\text{Inst}_2, \text{Path}_2], [\text{Inst}_3, \text{Path}_3], \dots\}$

Method

❖ DFSDT (Depth-First Search-based Decision Tree)

- ReAct는 Thought → Action → Observation 한 방향으로만 사고하기 때문에 에러 계속 전파
- 하나의 경로만 탐색하므로 전체 행동 공간 탐색 좁음

추론을 트리 탐색으로 확장
풀이는 하나만 찾으면 되므로 BFS 대신 DFS → API 호출 비용 절약



Solution Path Annotation

messages: system 프롬프트 + 명령1
functions: {get_forecast, search_hotels} + Finish 2종

"명령1을 실제로 풀어봐"

[Step 1]
Thought: 먼저 파리 날씨를 확인하자
Act: get_forecast(location="Paris", date="2026-07-16"
Observation: {temp: 24, condition: "sunny"}
[Step 2]
Thought: ...

[명령-대화로그] 데이터셋 생성

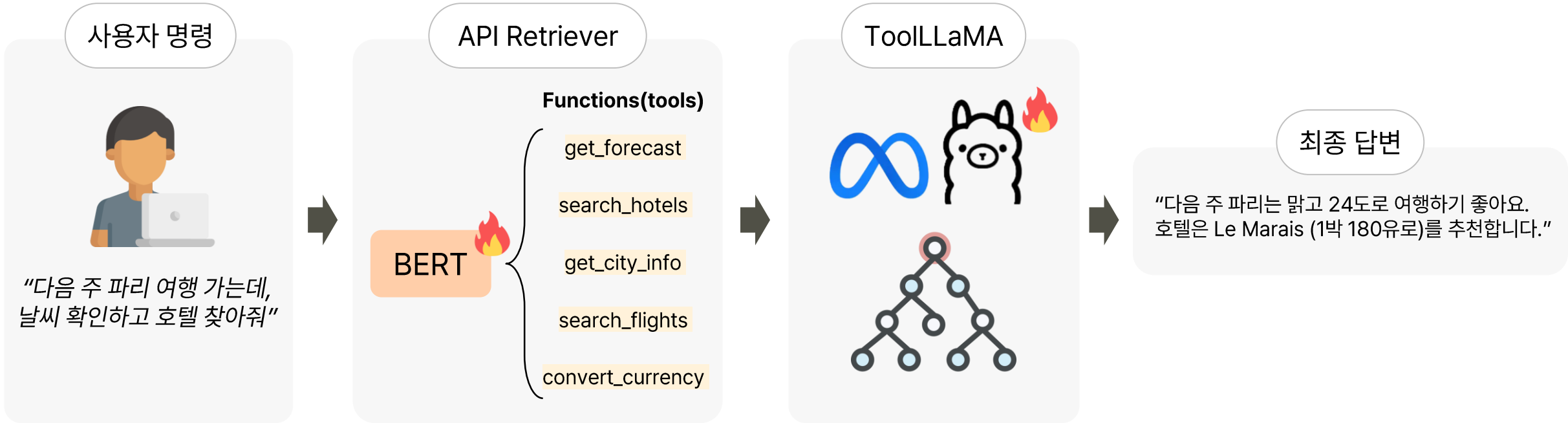
$$\{[\mathrm{Inst}_1, \mathrm{Path}_1], [\mathrm{Inst}_2, \mathrm{Path}_2], [\mathrm{Inst}_3, \mathrm{Path}_3], \ldots\}$$

Method

❖ ToolLLaMA + API Retriever

- DFSDT로 만든 데이터로 LLaMA-2 7B를 fine-tuning → 폐쇄형에 의존하지 않고도 도구를 다룰 수 있는 오픈소스 모델 획득
- 현실에서는 사람이 수만개의 API 중 필요한 것을 직접 지정할 수 없음

  → 명령 만으로 관련 API를 자동 추천하는 Sentence-BERT 기반 retriever 학습

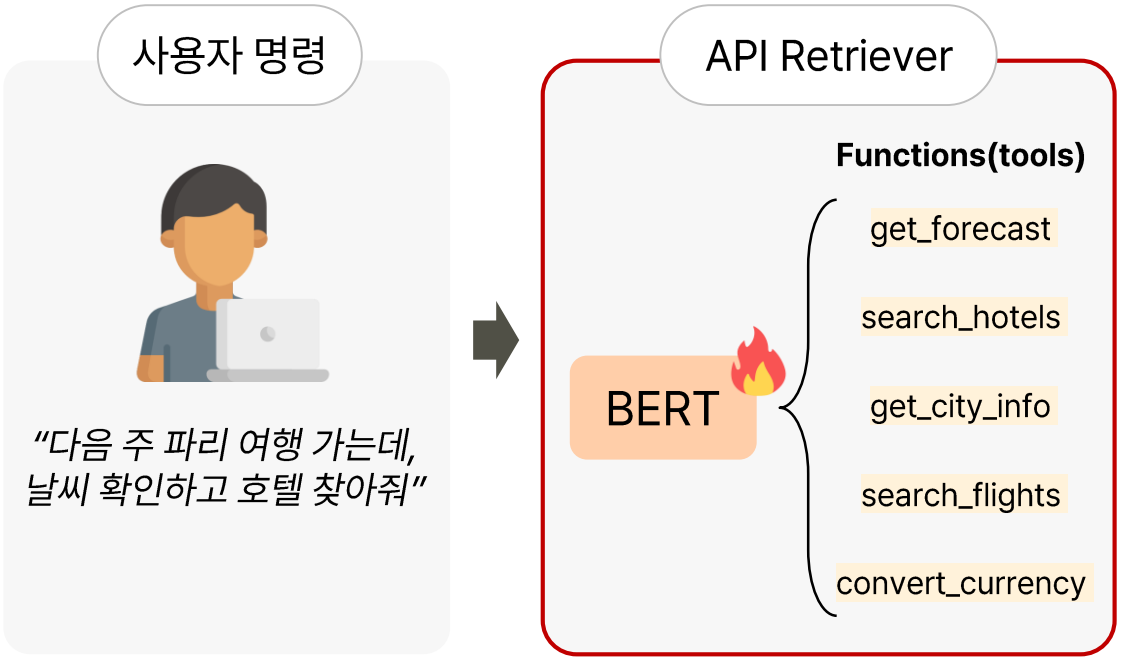


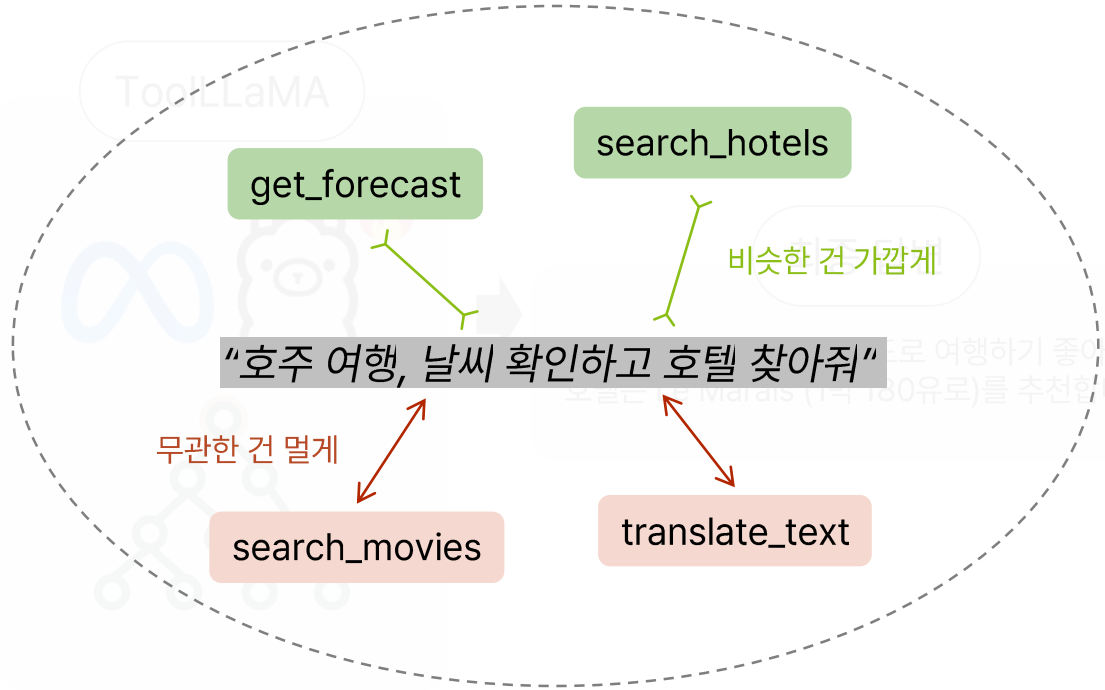
Method

❖ ToolLLaMA + API Retriever

- DFSDT로 만든 데이터로 LLaMA-2 7B를 fine-tuning → 폐쇄형에 의존하지 않고도 도구를 다룰 수 있는 오픈소스 모델 획득
- 현실에서는 사람이 수만개의 API 중 필요한 것을 직접 지정할 수 없음

  → 명령 만으로 관련 API를 자동 추천하는 Sentence-BERT 기반 retriever 학습





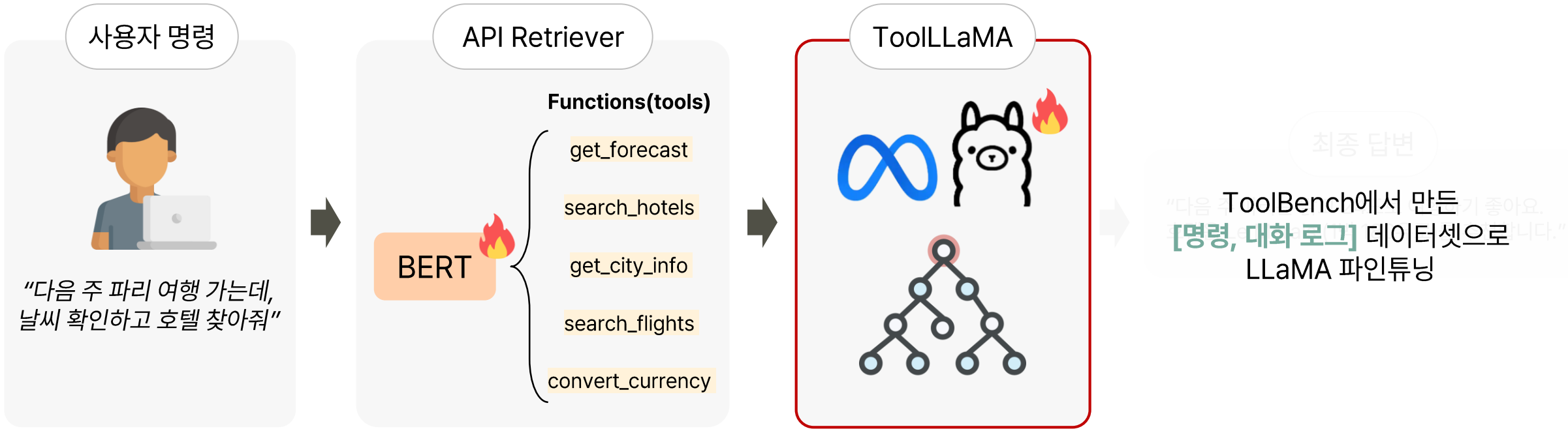
ToolBench에서 만든 **[명령-APIs]** 데이터셋으로 대조학습

Method

❖ ToolLLaMA + API Retriever

- DFSDT로 만든 데이터로 LLaMA-2 7B를 fine-tuning → 폐쇄형에 의존하지 않고도 도구를 다룰 수 있는 오픈소스 모델 획득
- 현실에서는 사람이 수만개의 API 중 필요한 것을 직접 지정할 수 없음

  → 명령 만으로 관련 API를 자동 추천하는 Sentence-BERT 기반 retriever 학습



Experiments

❖ ToolEval

- RapidAPI는 자주 변하며, 정답 풀이를 하나로 고정할 수 없고, 사람의 평가는 비효율적 → ChatGPT를 평가자로 활용
  - Pass Rate: 제한된 자원 내 명령 완수한 비율 / Win Rate: 'ChatGPT-ReACT'와 비교해 더 나은 풀이로 평가된 비율
- ToolBench의 명령 유형에 따라 학습 때 보지 못한 명령·tool·카테고리에 따른 일반화 성능 측정
- 오픈 소스 Vicuna, Alpaca와 폐쇄형 ChatGPT(gpt-3.5-turbo), GPT-4, Claude-2, Text-Davinci-003에 ReACT와 DFSDT를 적용

레벨	정의	난이도
Inst.	학습에 쓴 tool은 같지만, 못 본 명령	하
Tool	같은 카테고리 안의 못 본 tool	중
Cat.	아예 다른(못 본) 카테고리의 tool	상

Experiments

❖ Quantitative Evaluation

- 오픈 소스 Vicuna, Alpaca는 pass rate 0.0% → instruction tuning으로는 도구 사용 능력을 다루지 못함
- 모든 LLM에서 DFSDT가 ReACT 성능을 능가, ChatGPT+DFSDT(64.8) > GPT-4+ReACT(57.2)
  → 약한 모델 + 강한 전략 > 강한 모델 + 약한 전략
- ToolLLaMA(66.7)가 Claude-2(22.6)와 Text-Davinci-003(43.1)을 크게 앞서며, 학습 데이터를 만들어준 ChatGPT(64.8)과 대등
  → 학습에서 보지 못한 명령, 도구, 카테고리에도 잘 작동하여 일반화 성능 입증

Model	Method	I1-Inst.		I1-Tool		I1-Cat.		I2-Inst.		I2-Cat.		I3-Inst.		Average	
		Pass	Win	Pass	Win	Pass	Win	Pass	Win	Pass	Win	Pass	Win	Pass	Win
ChatGPT	ReACT	41.5	-	44.0	-	44.5	-	42.5	-	46.5	-	22.0	-	40.2	-
	DFSDT	54.5	60.5	65.0	62.0	60.5	57.3	75.0	72.0	71.5	**64.8**	62.0	69.0	64.8	64.3
Claude-2	ReACT	5.5	31.0	3.5	27.8	5.5	33.8	6.0	35.0	6.0	31.5	14.0	47.5	6.8	34.4
	DFSDT	20.5	38.0	31.0	44.3	18.5	43.3	17.0	36.8	20.5	33.5	28.0	65.0	22.6	43.5
Text-Davinci-003	ReACT	12.0	28.5	20.0	35.3	20.0	31.0	8.5	29.8	14.5	29.8	24.0	45.0	16.5	33.2
	DFSDT	43.5	40.3	44.0	43.8	46.0	46.8	37.0	40.5	42.0	43.3	46.0	63.0	43.1	46.3
GPT4	ReACT	53.5	60.0	50.0	58.8	53.5	63.5	67.0	65.8	72.0	60.3	47.0	78.0	57.2	64.4
	DFSDT	60.0	**67.5**	**71.5**	**67.8**	**67.0**	**66.5**	79.5	**73.3**	**77.5**	63.3	**71.0**	**84.0**	**71.1**	**70.4**
Vicuna	ReACT & DFSDT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Alpaca	ReACT & DFSDT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ToolLLaMA	ReACT	25.0	45.0	29.0	42.0	33.0	47.5	30.5	50.8	31.5	41.8	25.0	55.0	29.0	47.0
	DFSDT	57.0	55.0	61.0	55.3	62.0	54.5	77.0	68.5	77.0	58.0	66.0	69.0	66.7	60.0
	DFSDT-Retriever	**64.0**	62.3	64.0	59.0	60.5	55.0	**81.5**	68.5	68.5	60.8	65.0	73.0	67.3	63.1

Experiments

❖ Quantitative Evaluation

- 오픈 소스 Vicuna, Alpaca는 pass rate 0.0% → instruction tuning으로는 도구 사용 능력을 다루지 못함
- 모든 LLM에서 DFSDT가 ReACT 성능을 능가, ChatGPT+DFSDT(64.8) > GPT-4+ReACT(57.2)
  → 약한 모델 + 강한 전략 > 강한 모델 + 약한 전략
- ToolLLaMA(66.7)가 Claude-2(22.6)와 Text-Davinci-003(43.1)을 크게 앞서며, 학습 데이터를 만들어준 ChatGPT(64.8)과 대등
  → 학습에서 보지 못한 명령, 도구, 카테고리에도 잘 작동하여 일반화 성능 입증

Model	Method	I1-Inst.		I1-Tool		I1-Cat.		I2-Inst.		I2-Cat.		I3-Inst.		Average	
		Pass	Win	Pass	Win	Pass	Win	Pass	Win	Pass	Win	Pass	Win	Pass	Win
ChatGPT	ReACT	41.5	-	44.0	-	44.5	-	42.5	-	46.5	-	22.0	-	40.2	-
	DFSDT	54.5	60.5	65.0	62.0	60.5	57.3	75.0	72.0	71.5	**64.8**	62.0	69.0	64.8	64.3
Claude-2	ReACT	5.5	31.0	3.5	27.8	5.5	33.8	6.0	35.0	6.0	31.5	14.0	47.5	6.8	34.4
	DFSDT	20.5	38.0	31.0	44.3	18.5	43.3	17.0	36.8	20.5	33.5	28.0	65.0	22.6	43.5
Text-Davinci-003	ReACT	12.0	28.5	20.0	35.3	20.0	31.0	8.5	29.8	14.5	29.8	24.0	45.0	16.5	33.2
	DFSDT	43.5	40.3	44.0	43.8	46.0	46.8	37.0	40.5	42.0	43.3	46.0	63.0	43.1	46.3
GPT4	ReACT	53.5	60.0	50.0	58.8	53.5	63.5	67.0	65.8	72.0	60.3	47.0	78.0	57.2	64.4
	DFSDT	60.0	**67.5**	**71.5**	**67.8**	**67.0**	**66.5**	79.5	**73.3**	**77.5**	63.3	**71.0**	**84.0**	**71.1**	**70.4**
Vicuna	ReACT & DFSDT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Alpaca	ReACT & DFSDT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ToolLLaMA	ReACT	25.0	45.0	29.0	42.0	33.0	47.5	30.5	50.8	31.5	41.8	25.0	55.0	29.0	47.0
	DFSDT	57.0	55.0	61.0	55.3	62.0	54.5	77.0	68.5	77.0	58.0	66.0	69.0	66.7	60.0
	DFSDT-Retriever	**64.0**	62.3	64.0	59.0	60.5	55.0	**81.5**	68.5	68.5	60.8	65.0	73.0	67.3	63.1

Experiments

❖ Quantitative Evaluation

- 오픈 소스 Vicuna, Alpaca는 pass rate 0.0% → instruction tuning으로는 도구 사용 능력을 다루지 못함
- 모든 LLM에서 DFSDT가 ReACT 성능을 능가, ChatGPT+DFSDT(64.8) > GPT-4+ReACT(57.2)
  → 약한 모델 + 강한 전략 > 강한 모델 + 약한 전략
- ToolLLaMA(66.7)가 Claude-2(22.6)와 Text-Davinci-003(43.1)을 크게 앞서며, 학습 데이터를 만들어준 ChatGPT(64.8)과 대등
  → 학습에서 보지 못한 명령, 도구, 카테고리에도 잘 작동하여 일반화 성능 입증

Model	Method	I1-Inst.		I1-Tool		I1-Cat.		I2-Inst.		I2-Cat.		I3-Inst.		Average	
		Pass	Win	Pass	Win	Pass	Win	Pass	Win	Pass	Win	Pass	Win	Pass	Win
ChatGPT	ReACT	41.5	-	44.0	-	44.5	-	42.5	-	46.5	-	22.0	-	40.2	-
	DFSDT	54.5	60.5	65.0	62.0	60.5	57.3	75.0	72.0	71.5	**64.8**	62.0	69.0	64.8	64.3
Claude-2	ReACT	5.5	31.0	3.5	27.8	5.5	33.8	6.0	35.0	6.0	31.5	14.0	47.5	6.8	34.4
	DFSDT	20.5	38.0	31.0	44.3	18.5	43.3	17.0	36.8	20.5	33.5	28.0	65.0	22.6	43.5
Text-Davinci-003	ReACT	12.0	28.5	20.0	35.3	20.0	31.0	8.5	29.8	14.5	29.8	24.0	45.0	16.5	33.2
	DFSDT	43.5	40.3	44.0	43.8	46.0	46.8	37.0	40.5	42.0	43.3	46.0	63.0	43.1	46.3
GPT4	ReACT	53.5	60.0	50.0	58.8	53.5	63.5	67.0	65.8	72.0	60.3	47.0	78.0	57.2	64.4
	DFSDT	60.0	**67.5**	**71.5**	**67.8**	**67.0**	**66.5**	79.5	**73.3**	**77.5**	63.3	**71.0**	**84.0**	**71.1**	**70.4**
Vicuna	ReACT & DFSDT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Alpaca	ReACT & DFSDT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ToolLLaMA	ReACT	25.0	45.0	29.0	42.0	33.0	47.5	30.5	50.8	31.5	41.8	25.0	55.0	29.0	47.0
	DFSDT	57.0	55.0	61.0	55.3	62.0	54.5	77.0	68.5	77.0	58.0	66.0	69.0	66.7	60.0
	DFSDT-Retriever	**64.0**	62.3	64.0	59.0	60.5	55.0	**81.5**	68.5	68.5	60.8	65.0	73.0	67.3	63.1

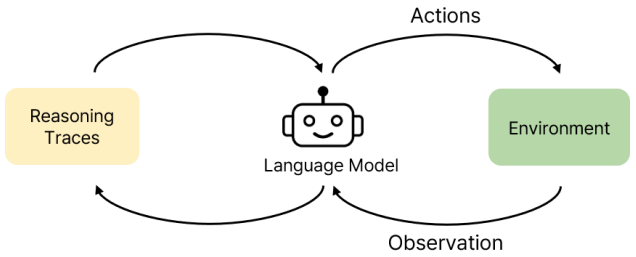
Conclusion

❖ What is Tool LLM?

- 복잡한 문제 해결을 위해 LLM이 외부 도구를 능동적으로 사용하며 그 결과를 바탕으로 추론을 이어가는 것

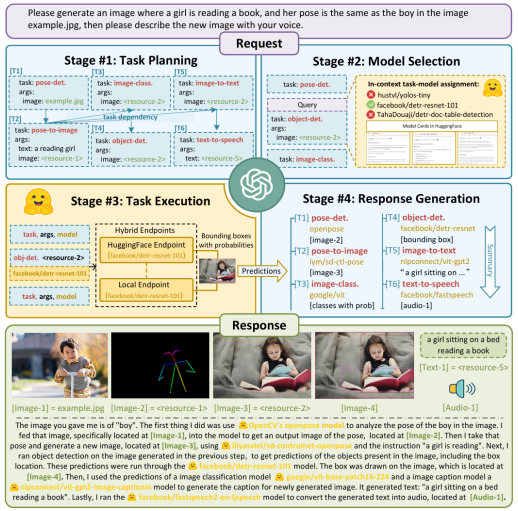
1) ReAct

Thought-Action-Observation 루프로
추론과 도구 사용의 결합



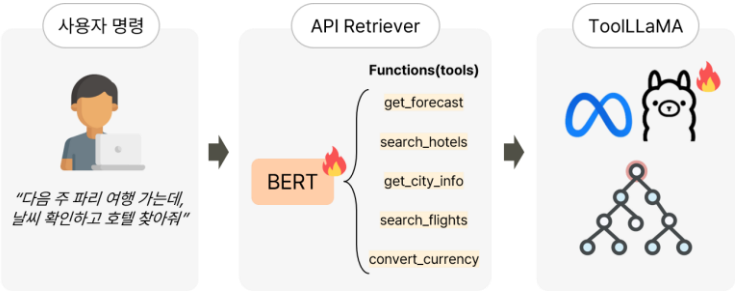
2) HuggingGPT

LLM의 계획 능력으로
다양한 AI 도구의 연결과 조율



3) ToolLLM

대규모 API 데이터셋과 학습을 통한
도구 선택 및 호출 능력 확장



Thank You

References

[1] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS. In 11th International Conference on Learning Representations, ICLR 2023.

[2] Shen, Y., Song, K., Tan, X., Li, D., Lu, W., & Zhuang, Y. (2023). Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. Advances in Neural Information Processing Systems, 36, 38154-38180.

[3] Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., ... & Sun, M. (2024, May). Toolllm: Facilitating large language models to master 16000+ real-world apis. In International Conference on Learning Representations (Vol. 2024, pp. 9695-9717).