
How to improve tool calling performance for LLM agents?



DMQA Open Seminar (2026. 10. 02)

김지현



김지현 (Jihyun Kim)

- 고려대학교 산업경영공학과 대학원 재학, Data Mining & Quality Analytics (DMQA) Lab.
- Ph.D. Candidate (2022.03 ~ Present, 석박사통합과정 10학기)

Research Interests

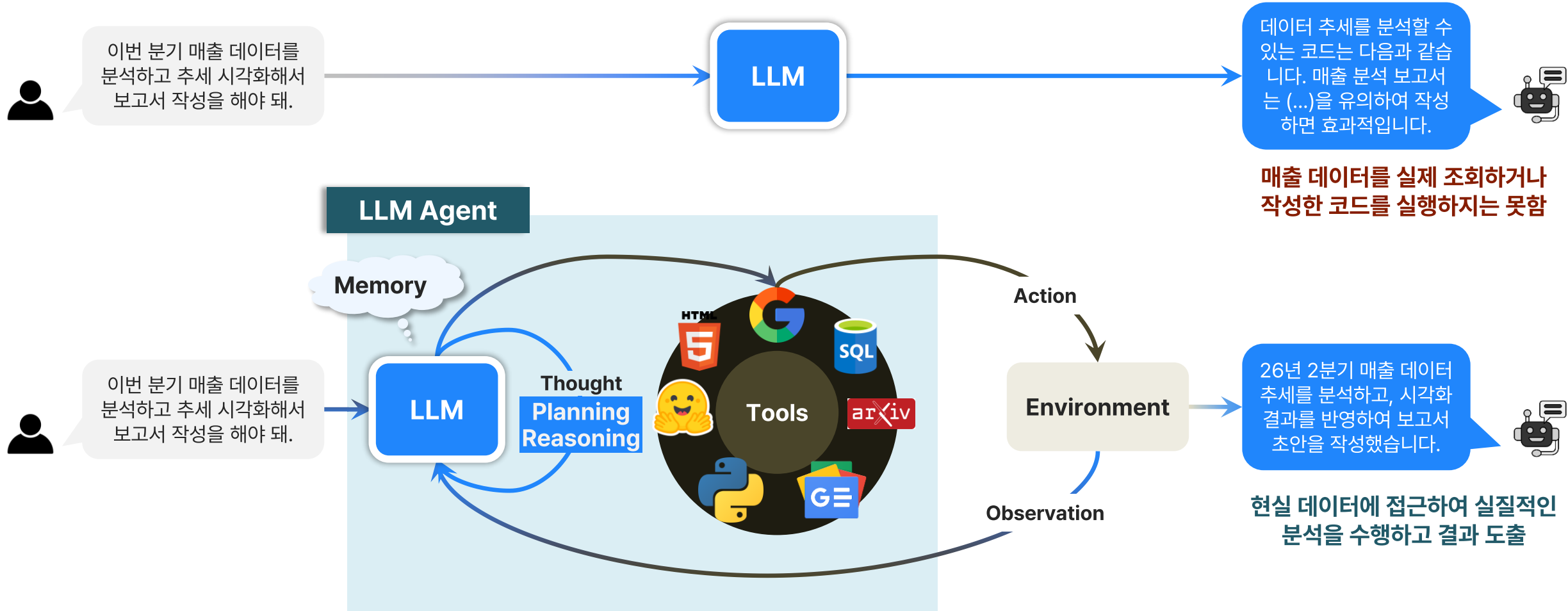
- 01 분포 변화에 대응 가능한 강건한 딥러닝 학습 기법 개발 (Domain adaptation / Test-time adaptation)
- 02 Vision-language models / Large language model agents

Contact

- jihyun_k@korea.ac.kr

LLM agents?

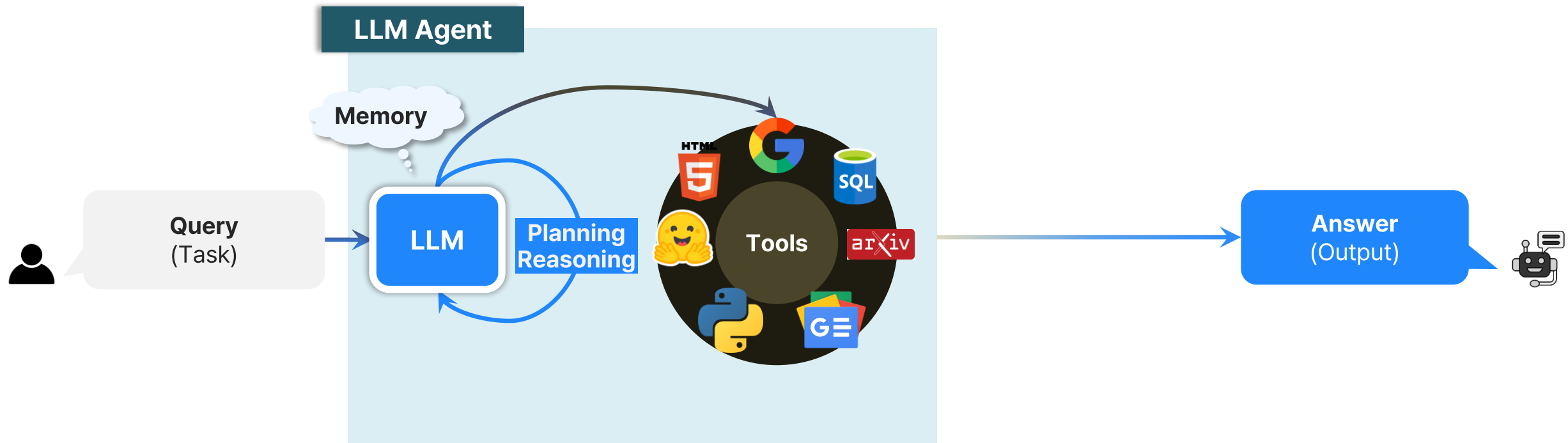
LLM (단순 질의 응답) vs. LLM Agent (사용자 요청을 능동적으로 해결)



LLM agents?

LLM (단순 질의 응답) vs. LLM Agent (사용자 요청을 능동적으로 해결)

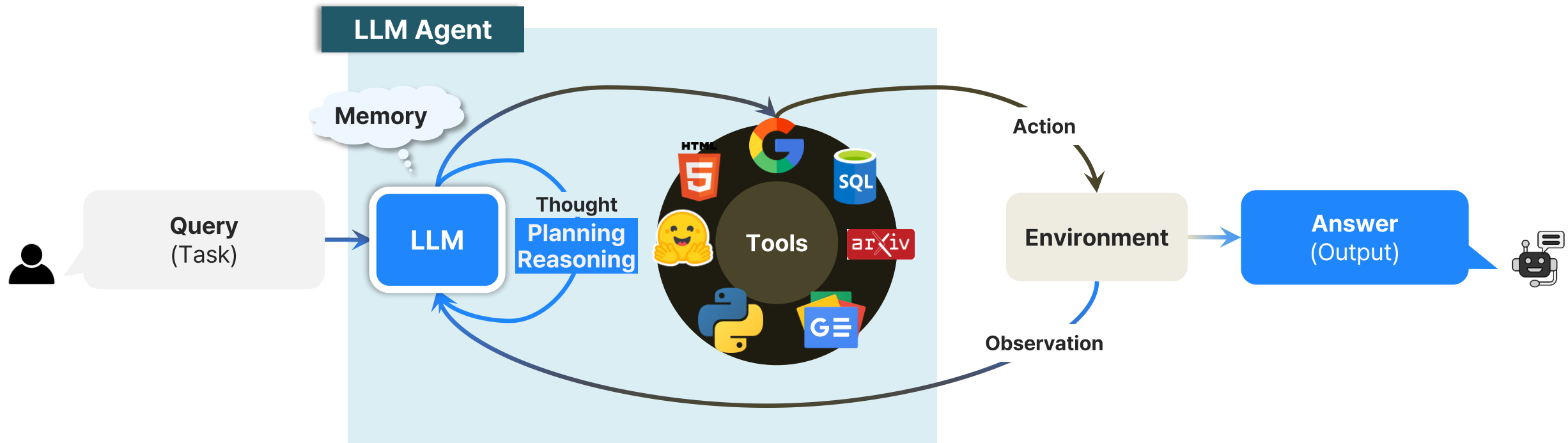
사용자의 요청을 해석하고, 요청 처리에 필요한 행동을 선택해 실행하고, 실행 결과를 확인하면서 작업을 진행하는 시스템



LLM agents?

LLM (단순 질의 응답) vs. LLM Agent (사용자 요청을 능동적으로 해결)

사용자의 요청을 해석하고, 요청 처리에 필요한 행동을 선택해 실행하고, 실행 결과를 확인하면서 작업을 진행하는 시스템



Tool calling?

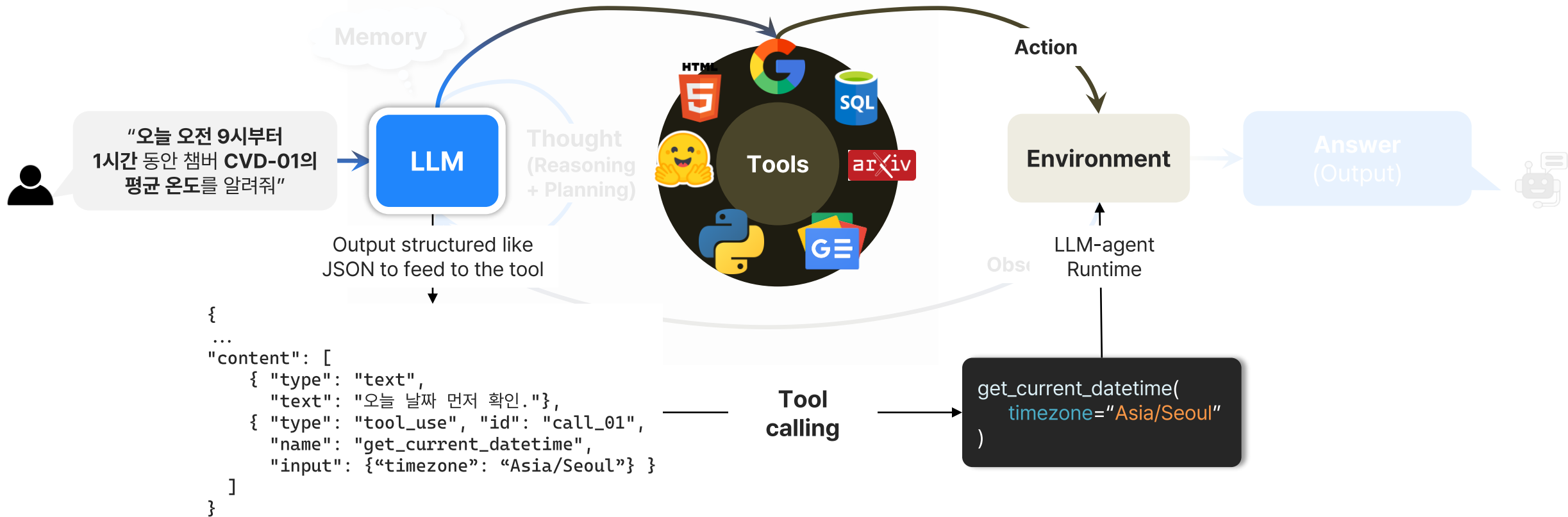
Tools as bridge to connect with external environments

Tool learning is to learn how bridge LLMs and external environments

Tools allow a given LLM to either **interact with an external environment** (such as databases) or **use external applications** (such as custom code to run)

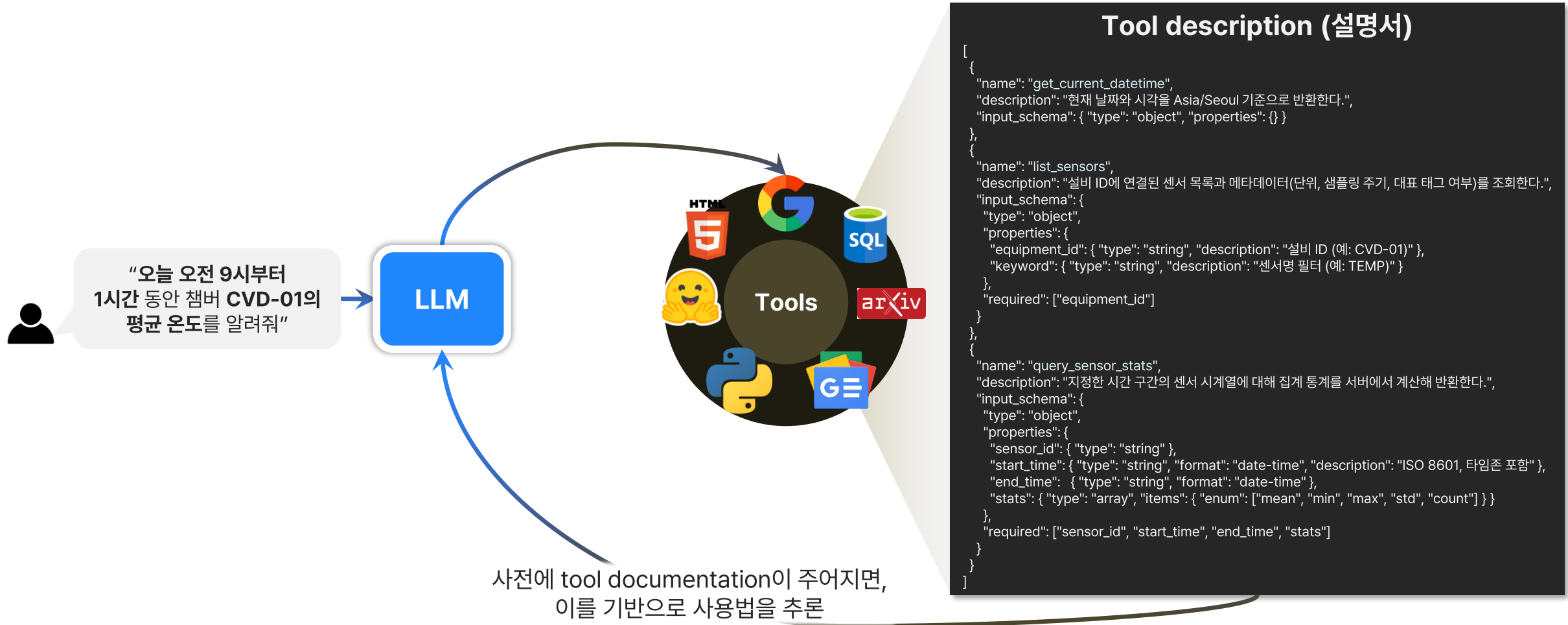
LLM Agent

Memory



How to improve tool calling performance?

Tool documentation is an indispensable component driving the success of tool learning!



기존의 tool description에는 어떤 문제가 있을까?

Existing tool documentation: **human-centered**, incomplete, redundant, inaccurate

Tool description (설명서)

```
{  
  "tool_name": "get_sensor_stats",  
  "description": "FDC 센서 데이터 조회 API입니다. 2019년 FDC 고도화 프로젝트의 일환으로 개발되어  
    공정 엔지니어의 설비 모니터링을 지원합니다. 설비의 센서 데이터를 가져옵니다. 문의: FDC 운영팀",  
  "parameters": {"sensor": "센서", "start": "시작 시간", "end": "종료 시간", "agg": "집계 방법"}  
}
```

기존의 tool description에는 어떤 문제가 있을까?

Existing tool documentation: **human-centered**, **incomplete**, redundant, inaccurate

Tool description (설명서)

```
{  
  "tool_name": "get_sensor_stats",  
  "description": "FDC 센서 데이터 조회 API입니다. 2019년 FDC 고도화 프로젝트의 일환으로 개발되어  
    공정 엔지니어의 설비 모니터링을 지원합니다. 설비의 센서 데이터를 가져옵니다. 문의: FDC 운영팀",  
  "parameters": {"sensor": "센서", "start": "시작 시간", "end": "종료 시간", "agg": "집계 방법"}  
}
```

`sensor`는 설비 ID 태그 형식만 허용 (e.g., CVD-01.CMB_TEMP)

`agg`는 "mean", "min", "max", "std", "count"만 허용

정보 누락

필요한 정보를 빼놓는 경우

기존의 tool description에는 어떤 문제가 있을까?

Existing tool documentation: **human-centered**, **incomplete**, **redundant**, **inaccurate**

Tool description (설명서)

```
{
  "tool_name": "get_sensor_stats",
  "description": "FDC 센서 데이터 조회 API입니다. 2019년 FDC 고도화 프로젝트의 일환으로 개발되어
    공정 엔지니어의 설비 모니터링을 지원합니다. 설비의 센서 데이터를 가져옵니다. 문의: FDC 운영팀",
  "parameters": {"sensor": "센서", "start": "시작 시간", "end": "종료 시간", "agg": "집계 방법"}
}
```

`sensor`는 설비 ID 태그 형식만 허용 (e.g., CVD-01.CMB_TEMP)

`agg`는 "mean", "min", "max", "std", "count"만 허용

정보 누락

필요한 정보를 빼놓는 경우

개발 연도, 프로젝트 명, 문의처는 도구 사용에 필요 없음

불필요한 정보

장황한 배경 설명이 너무 많아
중요 인자나 반환 값을 찾기 어려운 경우

기존의 tool description에는 어떤 문제가 있을까?

Existing tool documentation: **human-centered, incomplete, redundant, inaccurate**

Tool description (설명서)

```
{
  "tool_name": "get_sensor_stats",
  "description": "FDC 센서 데이터 조회 API입니다. 2019년 FDC 고도화 프로젝트의 일환으로 개발되어
    공정 엔지니어의 설비 모니터링을 지원합니다. 설비의 센서 데이터를 가져옵니다. 문의: FDC 운영팀",
  "parameters": {"sensor": "센서", "start": "시작 시간", "end": "종료 시간", "agg": "집계 방법"}
}
```

`sensor`는 설비 ID 태그 형식만 허용 (e.g., CVD-01.CMB_TEMP)

`agg`는 "mean", "min", "max", "std", "count"만 허용

정보 누락

필요한 정보를 빼놓는 경우

개발 연도, 프로젝트 명, 문의처는 도구 사용에 필요 없음

불필요한 정보

장황한 배경 설명이 너무 많아
중요 인자나 반환 값을 찾기 어려운 경우

원시 시계열 데이터가 아니라 통계 값 하나를 반환해야 함

부정확한 정보

실제 원하는 결과가 맞지 않는
엉뚱한 설명이 적힌 경우

기존의 tool description에는 어떤 문제가 있을까?

How to design LLM-centered tool documentation?

Tool description (설명서)

```
{  
  "tool_name": "get_sensor_stats",  
  "description": "FDC 센서 데이터 조회 API입니다. 2019년 FDC 고도화 프로젝트의 일환으로 개발되어  
    공정 엔지니어의 설비 모니터링을 지원합니다. 설비의 센서 데이터를 가져옵니다. 문의: FDC 운영팀",  
  "parameters": {"sensor": "센서", "start": "시작 시간", "end": "종료 시간", "agg": "집계 방법"}  
}
```

Q. 사람이 꼼꼼하게 고쳐 쓰면 되지 않을까?

Addressing these issues by manually correcting or enhancing tool documentation is
time-consuming and labor-intensive!

기존의 tool description에는 어떤 문제가 있을까?

How to design LLM-centered tool documentation?

Tool description (설명서)

```
{  
  "tool_name": "get_sensor_stats",  
  "description": "FDC 센서 데이터 조회 API입니다. 2019년 FDC 고도화 프로젝트의 일환으로 개발되어  
    공정 엔지니어의 설비 모니터링을 지원합니다. 설비의 센서 데이터를 가져옵니다. 문의: FDC 운영팀",  
  "parameters": {"sensor": "센서", "start": "시작 시간", "end": "종료 시간", "agg": "집계 방법"}  
}
```

Q. 사람이 꼼꼼하게 고쳐 쓰면 되지 않을까?

Addressing these issues by manually correcting or enhancing tool documentation is
time-consuming and labor-intensive!

Q. 그럼 LLM에게 '이 설명서를 더 좋게 다시 써줘'라고 시키면 되지 않을까? [1]

설명서만 보고 다시 쓰는 LLM은 `sensor` 인자에 설비 ID 태그 형식을 넣어야 한다는 것을 알 방법이 없음
원래 설명서에 없는 정보는 아무리 문장과 형식을 다듬어도 보충되지 않음

ICLR 2025 (Oral paper, Top 1.8%)

From Exploration to Mastery: Enabling LLMs to Master Tools via Self-Driven Interactions

사람이 새로운 도구에 익숙해지는 방식(trial and error)을 흉내 내보자!

→ 사람은 반복적인 상호작용과 직접 사용한 경험을 통해 도구 사용이 능숙해지며, 기능이 바뀌더라도 이해를 유지할 수 있음

DRAFT

Dynamically Refining tool documentation through the Analysis of Feedback and Trials

Published as a conference paper at ICLR 2025

FROM EXPLORATION TO MASTERY: ENABLING LLMs TO MASTER TOOLS VIA SELF-DRIVEN INTERACTIONS

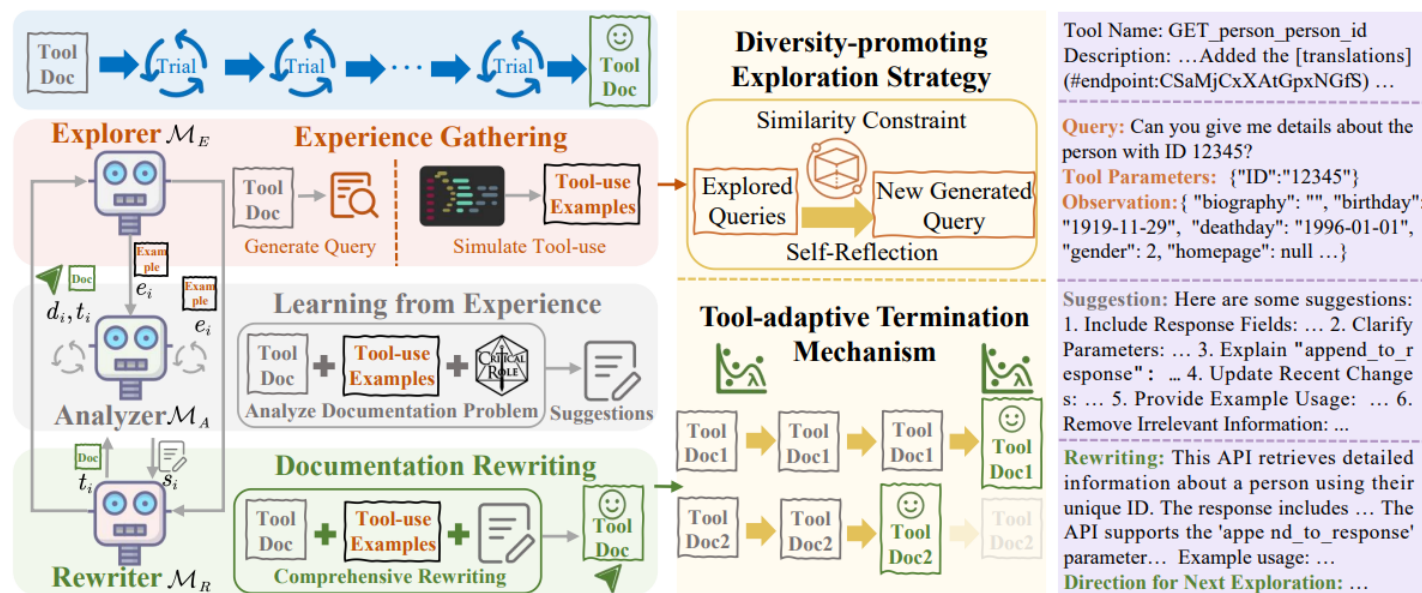
Changle Qu¹, Sunhao Dai¹, Xiaochi Wei², Hengyi Cai²,
Shuaiqiang Wang², Dawei Yin², Jun Xu¹, Ji-Rong Wen¹

¹Gaoling School of Artificial Intelligence, Renmin University of China, ²Baidu Inc.
{changlequ, sunhaodai, junxu, jrwen}@ruc.edu.cn, weixiaochi@baidu.com
caihengyi@ict.ac.cn, shqiang.wang@gmail.com, yindawei@aem.org

ABSTRACT

Tool learning enables Large Language Models (LLMs) to interact with external environments by invoking tools, serving as an effective strategy to mitigate the limitations inherent in their pre-training data. In this process, tool documentation plays a crucial role by providing usage instructions for LLMs, thereby facilitating effective tool utilization. This paper concentrates on the critical challenge of bridging the comprehension gap between LLMs and external tools due to the inadequacies and inaccuracies inherent in existing human-centric tool documentation. We propose a novel framework, **DRAFT**, aimed at **Dynamically Refining** tool documentation through the **Analysis of Feedback and Trials** emanating from LLMs' interactions with external tools. This methodology pivots on an innovative trial-and-error approach, consisting of three distinct learning phases: experience gathering, learning from experience, and documentation rewriting, to iteratively enhance the tool documentation. This process is further optimized by implementing a diversity-promoting exploration strategy to ensure explorative diversity and a tool-adaptive termination mechanism to prevent overfitting while enhancing efficiency. Extensive experiments on multiple datasets demonstrate that DRAFT's iterative, feedback-based refinement significantly ameliorates documentation quality, fostering a deeper comprehension and more effective utilization of tools by LLMs. Notably, our analysis reveals that the tool documentation refined via our approach demonstrates robust cross-model generalization capabilities.

[Framework Overview]



DRAFT (ICLR, 2025)

DRAFT, **D**ynamically **R**efining tool documentation through the **A**nalysis of **F**eedback and **T**rials

시행착오를 통해 tool documentation을 반복적으로 개선하는 방법론 제안:

Experience gathering → Learning from experience → Documentation Rewriting

Simulate plausible scenarios

Analyze the experience

Refine the tool documentation

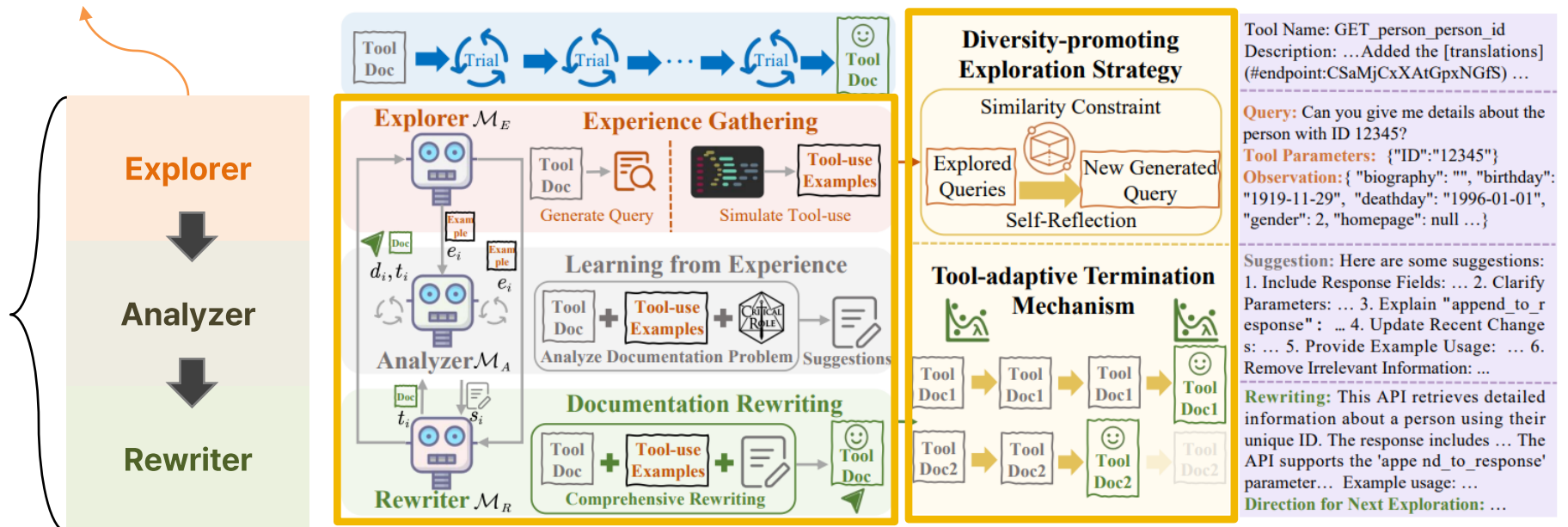
Diversity-promoting Exploration Strategy

도구 사용법 탐색 시 다양성 증진

Tool-adaptive Termination Mechanism

Tool마다 적절한 시점에 반복을
멈추는 도구 적응형 종료 메커니즘

[Framework Overview]



Explorer: 다양한 도구 사용 시나리오를 시뮬레이션하며, 현재 설명서로 LLM이 도구와 어떻게 상호작용하는지 확인

Explorer

Analyzer

Rewriter

1회차 explorer

Initial tool description t_0

```
{
  "tool_name": "get_sensor_stats",
  "description": "...",
  "parameters": {
    "sensor": "센서",
    "agg": "집계 방법" ... }
}
```

Explorer
 $\mathcal{M}_E$

`sensor` 인자에 설비 ID만 넣어서 호출 시도

Exploration instance e_i

```
{
  "User query" ( $e_i^q$ ): "어제 오후 2시부터 3시까지 CVD-01 챔버 온도가 평균 몇 도였는지 알려줘.",
  "Parameters" ( $e_i^p$ ): {"sensor": "CVD-01", "agg": "average", ...}
}
```

Tool feedback r_i

```
{
  "error": "INVALID_SENSOR_ID",
  "message": "sensor must be in '<equipment_id>.<tag>' format"
}
```

결과: 실패!

→에러 메시지를 통해 설명서에 없던 사실을 파악 (`sensor` 인자의 형식)

Explorer: 다양한 도구 사용 시나리오를 시뮬레이션하며, 현재 설명서로 LLM이 도구와 어떻게 상호작용하는지 확인

Analyzer: 앞서 얻은 데이터를 검토해 의도한 사용과 실제 사용 사이의 불일치를 찾아 개선안을 제시

Explorer

Analyzer

Rewriter

1회차 analyzer

Q. explorer의 결과를 바로 rewriter에 넘기면 안 될까? 왜 굳이 중간에 analyzer를 둘까?

A. 오답노트를 통해 틀린 문제를 다시 보고 분석하는 과정을 거쳐야 다음 시험에서 더 좋은 점수를 받는 것과 유사한 원리

Initial tool description t_0

```
{
  "tool_name": "get_sensor_stats",
  "description": "...",
  "parameters": {
    "sensor": "센서",
    "agg": "집계 방법" ... }
}
```

**Exploration instance e_i
and tool feedback r_i**

Analyzer
 $\mathcal{M}_A$

Revision suggestion s_i

{"Suggestions":

(일관성) 설명서는 `sensor`를 "센서"라고만 설명하지만, 실제 도구는 <설비ID>.<TAG> 형식을 요구하며, 설비 ID만 입력하는 경우에는 INVALID_SENSOR_ID 오류를 반환한다. 이 형식과 예시를 명시해야 한다.

(간결성) 개발 연도, 프로젝트 명, 문의처는 도구 사용과 무관하므로 삭제해야 한다.

(포괄성) "데이터를 가져온다"는 설명이 실제 반환 형태와 맞는지는 성공 응답을 아직 보지 못해 판단할 수 없다.}

기존 tool documentation의 세 가지 문제 (부정확, 불완전, 불필요)를 해결할 수 있는 진단 기준 (일관성, 포괄성, 간결성) 마련

Explorer: 다양한 도구 사용 시나리오를 시뮬레이션하며, 현재 설명서로 LLM이 도구와 어떻게 상호작용하는지 확인

Analyzer: 앞서 얻은 데이터를 검토해 의도한 사용과 실제 사용 사이의 불일치를 찾아 개선안을 제시

Rewriter: Analyzer가 전달한 분석 결과를 반영해 tool 설명서를 고치고, 다음에는 어떤 탐색을 수행할지 고민

Explorer

Analyzer

Rewriter

1회차 rewriter

Q. 설명서만 고치면 되지, 왜 다음 탐색 방향 (d_i)까지 제안할까?

A. 구체적인 탐색 방향이 없다면 explorer는 매번 설명서만 보고 무작위로 탐색을 수행하게 됨 (비용 낭비, 효율성 저하)

Initial tool description t_0

```
{
  "tool_name": "get_sensor_stats",
  "description": "...",
  "parameters": {
    "sensor": "센서",
    "agg": "집계 방법" ... }
}
```

Exploration instance e_i
and tool feedback r_i

Revision suggestion s_i

Rewriter
 $\mathcal{M}_R$

Updated version of the tool documentation t_i

```
{"description": "설비 센서의 데이터를 조회합니다. sensor는 '<설비 ID>.<태그>' 형식이어야 하며(예: CVD-01.CHB_TEMP), 설비 ID만 입력하면 INVALID_SENSOR_ID 오류가 반환됩니다."}
```

Suggestions for the next round
of exploration directions d_i

올바른 형식의 sensor 값으로 호출하여 성공 응답의 구조를 확인해 볼 것.
다른 설비나 다른 집계 방법도 시도해 볼 것.

기존 description에서 불필요한 내용은 삭제하고, 정확한 정보를 추가하여 수정,
이후 explorer가 다양한 방향으로 탐색할 수 있도록 구체적인 방향성 지시

Explorer: 다양한 도구 사용 시나리오를 시뮬레이션하며, 현재 설명서로 LLM이 도구와 어떻게 상호작용하는지 확인

Q. 그런데 Explorer가 수행하는 탐색이 매번 비슷하면, 여러 번 시행착오를 겪는 의미가 없어지지 않을까?

Explorer

Analyzer

Rewriter

Explorer

2회차 explorer

Current tool description t_1

```
{
  "tool_name": "get_sensor_stats",
  "description": "설비 센서의 데이터를 조회합니다. sensor는 '<설비ID>.<태그>' 형식이어야 하며 (예: CVD-01.CHB_TEMP), 설비 ID만 입력하면 INVALID_SENSOR_ID 오류가 반환됩니다."
  "parameters": {
    "sensor": "센서",
    "agg": "집계 방법" ... }
}
```

Next step exploration direction d_{i-1}

올바른 형식의 sensor 값으로 호출하여 성공 응답 구조를 확인해볼 것. 다른 설비나 집계 방법도 시도해 볼 것.



`sensor` 인자에 적절한 형식으로 인자 값을 적음
설비 ID → 설비ID.TAG

Exploration instance e_i

```
{
  "User query" ( $e_i^q$ ): "CVD-01 챔버의 어제 저녁 평균 온도 알려줘.",
  "Parameters" ( $e_i^p$ ): {
    "sensor": "CVD-01.CHB_TEMP",
    "agg": "average", ... }
}
```

이전 질의와 설비도, 측정 항목도, 통계 방법도 같고 시간대만 조금 다른 질문
→ 다른 설비, 다른 센서, 다른 통계 방법을 시도해야 도구의 다른 면을 볼 수 있음

Explorer: 다양한 도구 사용 시나리오를 시뮬레이션하며, 현재 설명서로 LLM이 도구와 어떻게 상호작용하는지 확인

Q. 그런데 Explorer가 수행하는 탐색이 매번 비슷하면, 여러 번 시행착오를 겪는 의미가 없어지지 않을까?

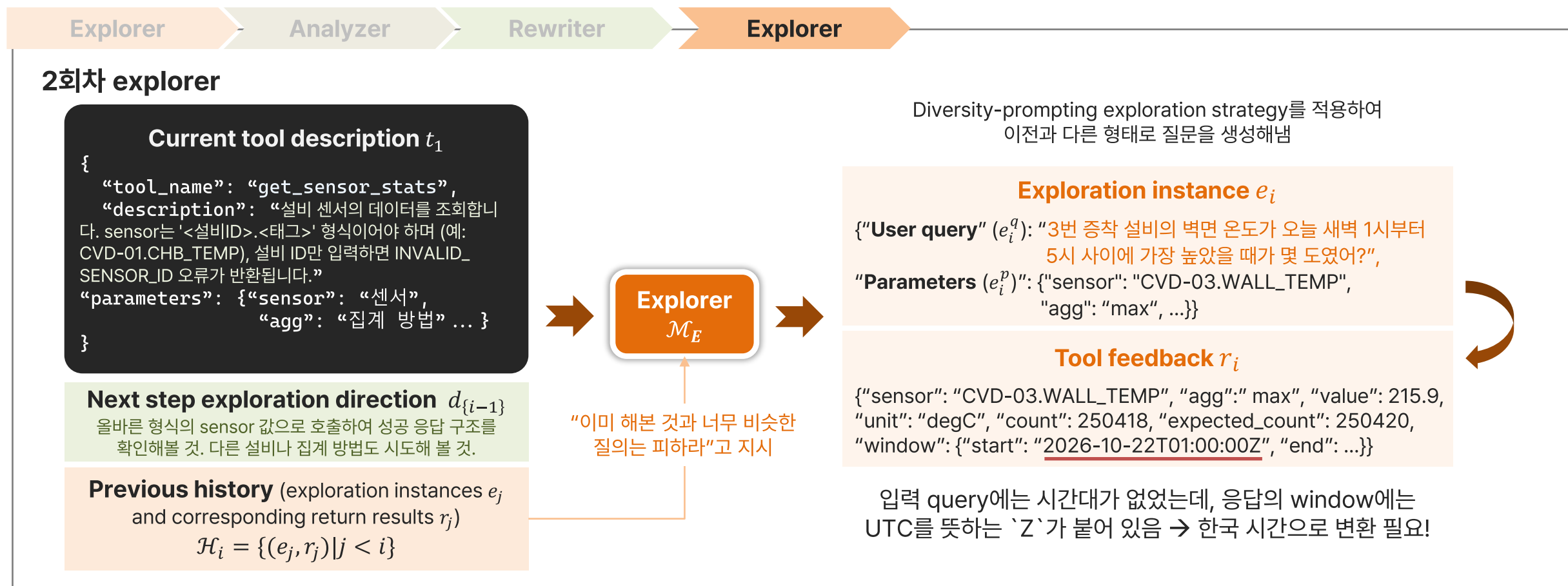
A. 탐색의 다양성을 높이기 위해 'diversity-promoting exploration strategy (다양성 촉진 탐색 전략)' 제안



Explorer: 다양한 도구 사용 시나리오를 시뮬레이션하며, 현재 설명서로 LLM이 도구와 어떻게 상호작용하는지 확인

Q. 그런데 Explorer가 수행하는 탐색이 매번 비슷하면, 여러 번 시행착오를 겪는 의미가 없어지지 않을까?

A. 탐색의 다양성을 높이기 위해 'diversity-promoting exploration strategy (다양성 촉진 탐색 전략)' 제안



Analyzer: 앞서 얻은 데이터를 검토해 의도한 사용과 실제 사용 사이의 불일치를 찾아 개선안을 제시

Q. Analyzer가 매번 같은 지적만 반복하면, revision suggestion의 퀄리티가 낮아지지 않을까?

A. 지금까지의 **documentation** 수정 이력을 함께 입력 받아서, 중복되거나 반복적인 제안을 피하고 아직 개선이 필요한 부분에 집중

Explorer

Analyzer

Rewriter

Explorer

Analyzer

2회차 analyzer

Current tool description t_1

```
{
  "tool_name": "get_sensor_stats",
  "description": "설비 센서의 데이터를 조회합니다. sensor는 '<설비ID>.<태그>' 형식이어야 하며 (예: CVD-01.CHB_TEMP), 설비 ID만 입력하면 INVALID_SENSOR_ID 오류가 반환됩니다.",
  "parameters": {
    "sensor": "센서",
    "agg": "집계 방법" ... }
}
```

Exploration instance e_i
and tool feedback r_i

The history of
documentation revisions

$\mathcal{T}_i = \{t_j | j < i\}$

Analyzer
 $\mathcal{M}_A$

Revision suggestion s_i

{"Suggestions":

(일관성) 설명서는 "센서 데이터를 조회한다"고 하지만, 실제로는 원시 시계 열이 아니라 지정 구간의 통계 값 하나('value')와 단위('unit'), 데이터 개수('count', 'expected_count')를 반환한다. 반환 구조를 명시해야 한다.

(일관성) 시간대 없이 입력한 시각이 응답에서 UTC(Z)로 표시되었다. 시간대를 명시하지 않으면 UTC로 해석된다는 점과, 한국 시간 기준이라면 +09:00을 붙여야 한다는 점을 명시해야 한다.

(포괄성) 'count'와 'expected_count'를 비교하면 결측 데이터 여부를 판단할 수 있다는 점을 추가하면 유용하다.}

Rewriter: Analyzer가 전달한 분석 결과를 반영해 tool 설명서를 고치고, 다음에는 어떤 탐색을 수행할지 고민

Q. 설명서를 수정할 때 이전 수정 이력은 왜 참고할까?

A. 각 버전의 설명서가 이전 버전을 기반으로 쌓아 올려지고, 매번 처음부터 다시 쓰면 앞선 정보를 잊을 수 있기 때문

Explorer

Analyzer

Rewriter

Explorer

Analyzer

Rewriter

2회차 rewriter

Current tool description t_1

```
{
  "tool_name": "get_sensor_stats",
  "description": "설비 센서의 데이터를 조회합니다. sensor는 '<설비ID>.<태그>' 형식이어야 하며 (예: CVD-01.CHB_TEMP), 설비 ID만 입력하면 INVALID_SENSOR_ID 오류가 반환됩니다.",
  "parameters": {
    "sensor": "센서",
    "agg": "집계 방법" ... }
}
```

Exploration instance e_i
and tool feedback r_i

Revision suggestion s_i

The history of
documentation revisions

$\mathcal{T}_i = \{t_j | j < i\}$



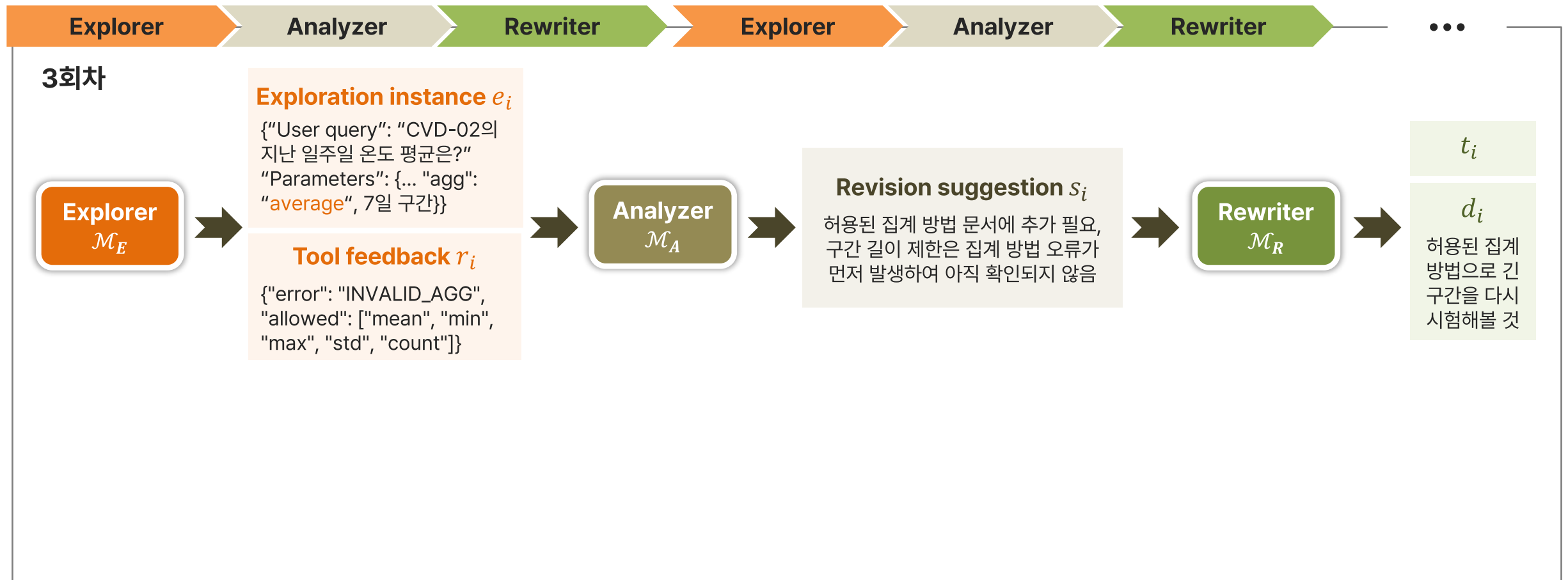
Updated version of the tool documentation t_i

```
{
  "description": "설비 센서의 시계열에 대해 지정한 구간의 통계값 하나를 반환합니다(원시 시계열은 반환하지 않음). sensor는 '<설비ID>.<태그>' 형식이어야 하며(예: CVD-01.CHB_TEMP), 설비 ID만 입력하면 INVALID_SENSOR_ID 오류가 반환됩니다. start와 end에 시간대를 명시하지 않으면 UTC로 해석되므로, 한국 시간 기준이라면 '+09:00'을 붙여야 합니다. 응답은 value, unit, count, expected_count를 포함하며, count가 expected_count보다 작으면 결측 데이터가 있음을 뜻합니다."
}
```

Suggestions for the next round
of exploration directions d_i

agg 파라미터에 어떤 값들이 허용되는지,
조회 구간의 길이에 제한이 있는지 확인해볼 것

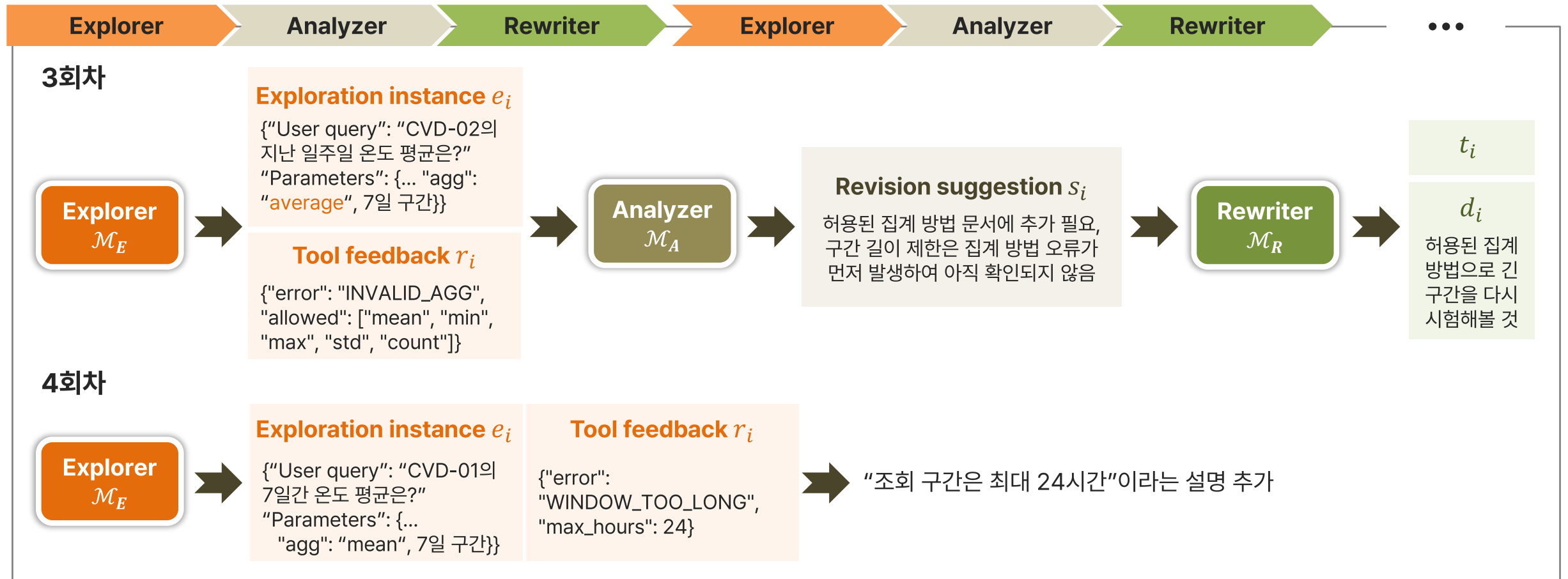
Q. 이 과정을 반복하면 설명서는 어떻게 변할까?
3회차: agg 파라미터에 입력 가능한 인자가 무엇인지를 확인



Q. 이 과정을 반복하면 설명서는 어떻게 변할까?

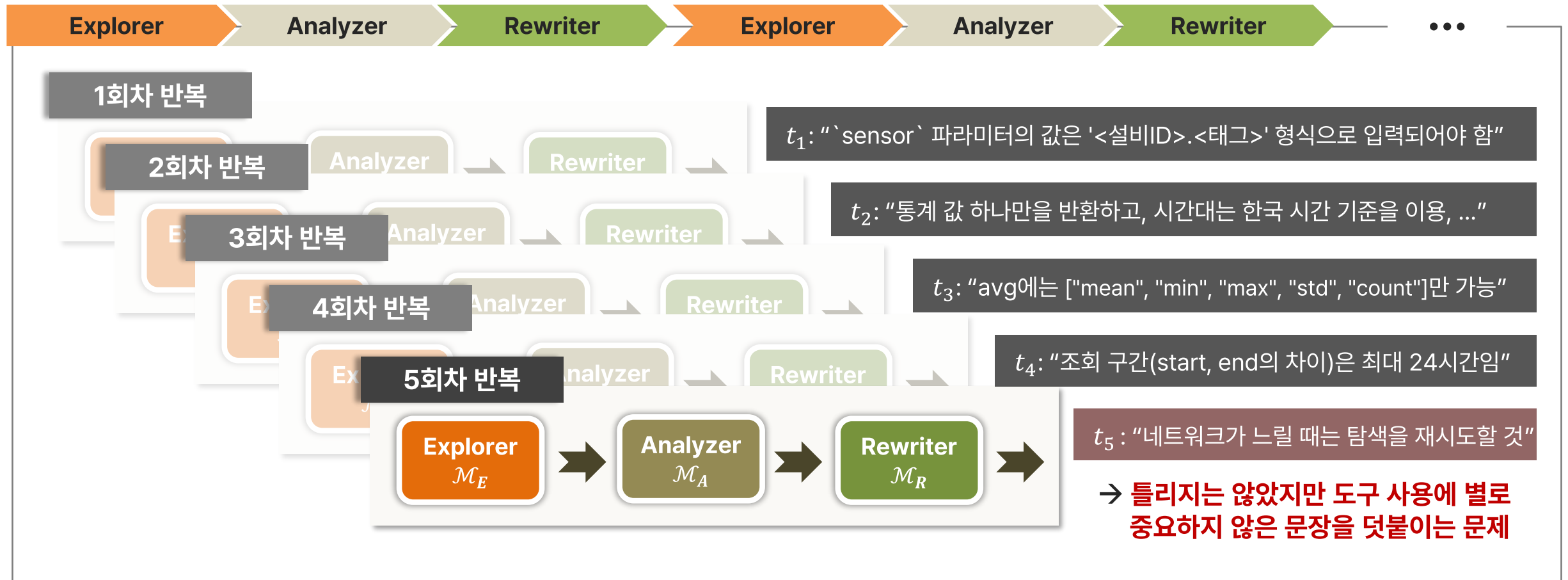
3회차: agg 파라미터에 입력 가능한 인자가 무엇인지를 확인

4회차: 조회 구간 길이가 최대 24시간인 것을 확인



Q. 그렇다면 반복은 많이 할 수록 좋을까?

No! 반복이 늘어날 수록 설명서가 점점 길어지고, 중복 정보가 포함되어 **효과적 도구 사용을 오히려 방해할 수도 있음**



Tool-adaptive termination mechanism: 탐색을 언제 종료할 지를 결정

연속된 두 버전의 설명서 사이에 변화가 거의 없으면, 반복이 수렴했다고 판단



Tool-adaptive termination mechanism: 탐색을 언제 종료할 지를 결정

연속된 두 버전의 설명서 사이에 변화가 거의 없으면, 반복이 수렴했다고 판단



Tool-adaptive termination mechanism: 탐색을 언제 종료할 지를 결정

연속된 두 버전의 설명서 사이에 변화가 거의 없으면, 반복이 수렴했다고 판단



DRAFT (ICLR, 2025)

Q. 제안 방법론의 효과를 어떻게 평가할 수 있을까?

데이터셋: ToolBench와 RestBench 두 벤치마크를 사용

DRAFT backbone model: GPT-4o

ToolBench

범주: 날씨 (weather)

- ↳ 도구: OpenWeather 서비스
 - ↳ API: 현재 날씨 조회
 - ↳ API: 5일 예보 조회
 - ↳ API: 대기질 조회

...

범주: 금융 (finance)

- ↳ 도구: 환율 서비스
 - ↳ API: 실시간 환율 조회
 - ↳ API: 통화 변환

I3, 서로 다른 범주의 도구들에 대한 통합적 이해를 필요로 하는 벤치마크
(e.g., "다음 주 도쿄 출장 날씨를 확인하고, 100만원을 엔화로 바꾸면 얼마인지 확인해.")

RestBench

TMDB

영화 관련 API 54개

Spotify

음악 관련 API 40개

도구를 여러 번 호출할 때, 도구 간 의존 관계를 이해해야 하는 벤치마크

(e.g., "앤 해서웨이가 출연한 영화는 몇 편이야?"

- 1. 인물 검색 API를 통해 person_id 얻기,
- 2. 영화 크레딧 API 통해 작품 개수 세기)

Initial tool description

```
{
  "tool_name": "get_sensor_stats",
  "description": "...",
  "parameters": {"sensor": "센서",
    "agg": "집계 방법"...}
}
```



Explorer

Analyzer

Rewriter

GPT-4o



Revised tool description

```
{
  "tool_name": "get_sensor_stats",
  "description": "...",
  "parameters": {"sensor": "센서",
    "agg": "집계 방법"...}
}
```



Task
LLM

GPT-4o,
GPT-4o-mini,
Llama-3-70B

Q. 제안 방법론의 효과를 어떻게 평가할 수 있을까?

평가지표 1. Correct Path rate (CP)%

평가지표 2. Win rate (Win)%

Correct path rate (CP, 정답 경로 비율)%

모델이 생성한 API 호출 경로가 정답 API 호출 경로를
부분 수열로 포함하면 올바른 경로를 생성한 것으로 판단

사용자 요청: "Taylor swift 노래 세 곡으로 플레이리스트 만들고,
리스트 이름은 'swiftie'로 해줘."

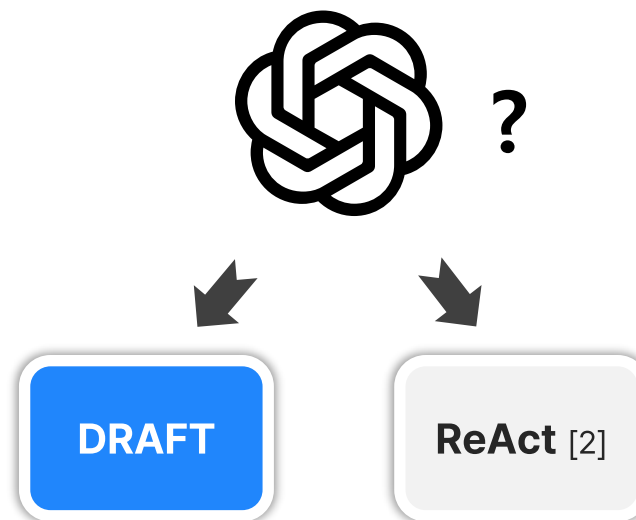
정답 경로:

- 1) GET /search (곡 검색)
- 2) GET /me (내 사용자 ID 확인)
- 3) POST /users/{user_id}/playlists (플레이리스트 생성)
- 4) POST /playlists/{playlist_id}tracks (곡 추가)

풀이	호출 순서	CP 판정	과제 해결 여부
A	search → me → 생성 → 추가	1	해결
B	search → search → me → artist → 생성 → 추가	1	해결 (비효율적)
C	me → search → 생성 → 추가	0	미해결
D	search → me → 생성	0	미해결
E	me → search → me → 생성 → 추가	1	해결 (비효율적)
F	search → me → 생성 → 추가 (엉뚱한 곡)	0	미해결

Win rate (Win, 승률)%

ChatGPT (LLM judge)가 ReAct와 DRAFT의 호출 결과를 비교하여 채점
(따라서 ReAct는 정의상 win% 성능이 50%)



Q. 제안 방법론은 정말 효과가 있을까?

모든 LLM이 DRAFT로 수정한 설명서를 사용할 때 더 좋은 성능을 보인 것을 확인

Table 1: Performance comparison of different methods on three datasets. Win% is calculated by comparing each method with ReAct. The term “-” means that EasyTool has not been implemented on the Spotify dataset. The best result for each LLM is in **bold**.

Model	Method	RestBench-TMDB		RestBench-Spotify		ToolBench	
		CP%	Win %	CP%	Win%	CP%	Win %
GPT-4o-mini	ReAct	48.00	50.00	24.56	50.00	35.00	50.00
	DFSdT	50.00	68.00	35.08	61.40	37.00	84.00
	EasyTool	56.00	75.00	-	-	42.00	85.00
	DRAFT (Ours)	62.00	82.00	43.85	78.94	47.00	88.00
Llama-3-70B	ReAct	72.00	50.00	26.31	50.00	41.00	50.00
	DFSdT	74.00	38.00	63.15	61.40	42.00	54.00
	EasyTool	76.00	64.00	-	-	46.00	60.00
	DRAFT (Ours)	86.00	64.00	66.66	64.91	53.00	62.00
GPT-4o	ReAct	71.00	50.00	28.07	50.00	37.00	50.00
	DFSdT	74.00	61.00	64.91	56.14	41.00	73.00
	EasyTool	79.00	62.00	-	-	45.00	77.00
	DRAFT (Ours)	88.00	71.00	70.17	84.21	51.00	78.00

[Main Result]

- 1) 모든 LLM, 데이터셋, 성능 평가 지표에 대해서 가장 우수한 성능을 보이며 DRAFT 방법론의 효과 확인
- 2) 기반 모델이 좋아질 수록 성능 향상 수준이 더 높아짐
- 3) 특히 ToolBench에서는 DRAFT를 적용한 GPT-4o-mini(47.00)가 DRAFT 없는 GPT-4o(37.00)를 넘어섬
→ 좋은 설명서를 만드는 것이, 더 크고 똑똑한 모델을 쓰는 것 이상으로 효과적일 수 있음을 보여주는 결과
- 4) GPT-4o로 고친 설명서를 다른 모델 (Llama)에 적용해도 여전히 효과가 있음

Q. 수정 반복 횟수는 성능에 어떤 영향을 줄까?

설명서를 반복해서 고치고 tool마다 적절한 시점에 멈추는 메커니즘의 효과를 확인하기 위해 반복 횟수에 따른 성능 차이 확인

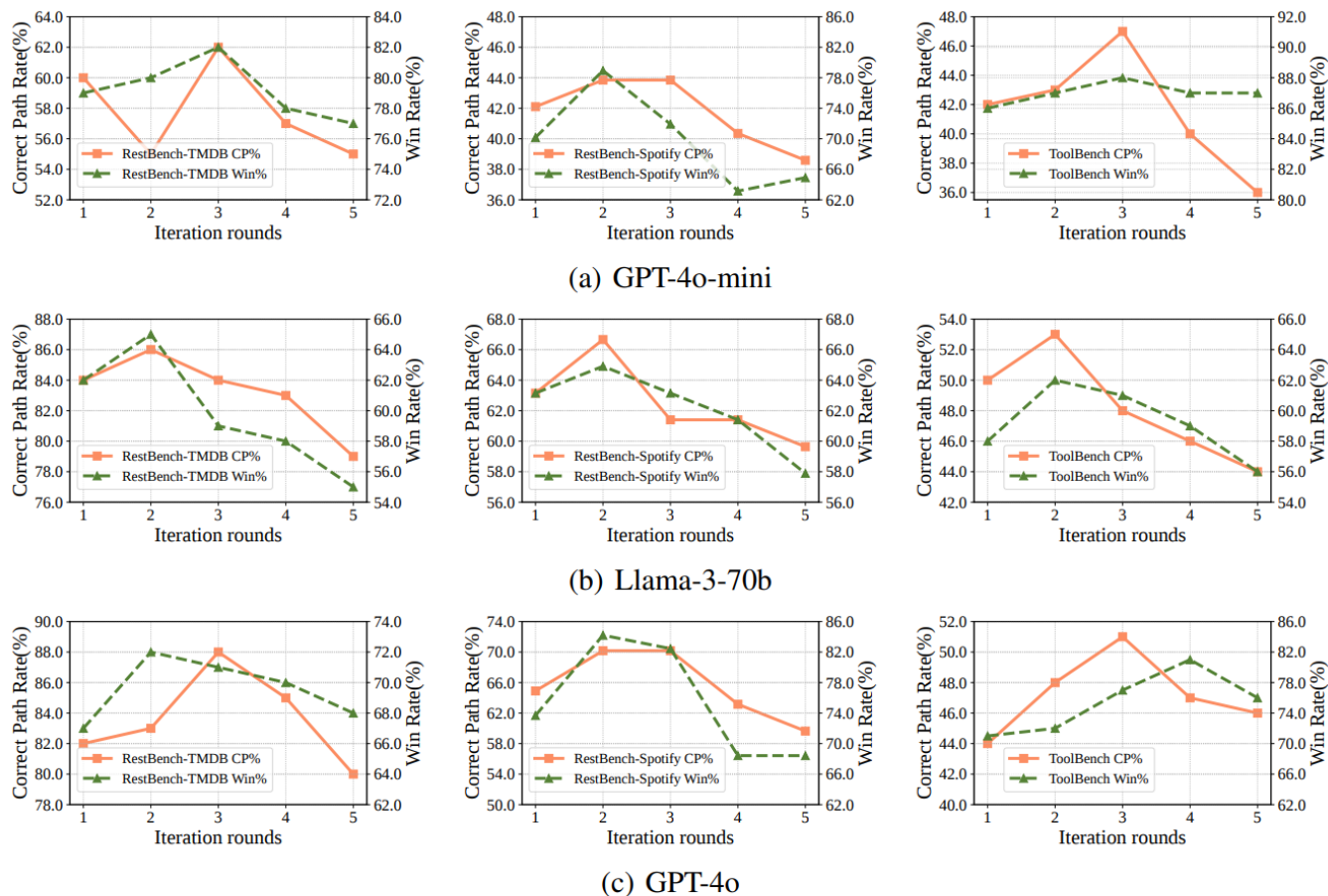


Figure 6: A comparative analysis of performance based on varying numbers of iteration rounds.

[모든 tool에 대해 같은 반복 횟수 일괄 적용 결과]

- 1) 모든 모델과 데이터셋에서 반복 횟수가 늘어날 수록 성능이 올라가다가 이후 떨어지는 공통된 경향 확인
→ 다양한 예시를 탐색하다가, 이후에는 오히려 불필요한 정보가 누적되었을 가능성
- 2) 설명서 수정을 불필요하게 많이 할 필요가 없음을 확인 하였으나, 이것이 "도구마다 최적 횟수가 다르다"는 것을 직접 보여주지는 않는 한계

Q. 두 가지 모듈(diversity, adaptive)은 성능 향상에 정말 효과가 있었을까?

Table 2: Ablation study of the proposed DRAFT.

유사도 제약 없이 explorer가
생성한 모든 탐색 사례를 받아들임

조기 종료 없이 항상 지정된
최대 반복 횟수까지 수정함

Mehtods	TMDB	
	CP%	Win%
DRAFT	88.00	71.00
w/o diversity	84.00	69.00
w/o adaptive	80.00	68.00

Model	Method	RestBench-TMDB		RestBench-Spotify		ToolBench	
		CP%	Win%	CP%	Win%	CP%	Win%
GPT-4o-mini	DRAFT (Ours)	62.00	82.00	43.85	78.94	47.00	88.00
	w/o diversity	60.00	79.00	42.10	70.17	42.00	86.00
	w/o adaptive	55.00	77.00	38.59	57.89	36.00	87.00
Llama-3-70B	DRAFT (Ours)	86.00	64.00	66.66	64.91	53.00	62.00
	w/o diversity	84.00	62.00	63.15	63.15	48.00	61.00
	w/o adaptive	79.00	55.00	59.64	57.89	44.00	56.00
GPT-4o	DRAFT (Ours)	88.00	71.00	70.17	84.21	51.00	78.00
	w/o diversity	84.00	69.00	64.91	73.68	44.00	72.00
	w/o adaptive	80.00	68.00	59.64	68.42	46.00	76.00

[Results]

- 1) **Diversity** 전략이 없으면 매 회차 탐색 사례가 서로 비슷해져 같은 예산에서 성능을 향상시키는 데 한계
- 2) **Adpative** 전략이 없으면 과도한 반복으로 중복된 정보가 생겨 과적합이 될 수 있음
- 3) 어떤 모듈이 더 중요한 지는 데이터셋마다 다르지만, 대부분 **adaptive** 모듈을 제외했을 때 성능이 더 하락
→ 다만, 최대 반복 횟수가 몇 번인지는 기재되지 않음
- 4) Adaptive termination 메커니즘의 장점 중 하나인 **효율성(계산 자원과 시간 절약)**은 실험되지 않음

ACL Findings 2025

Play2Prompt: Zero-shot Tool Instruction Optimization for LLM Agents via Tool Play

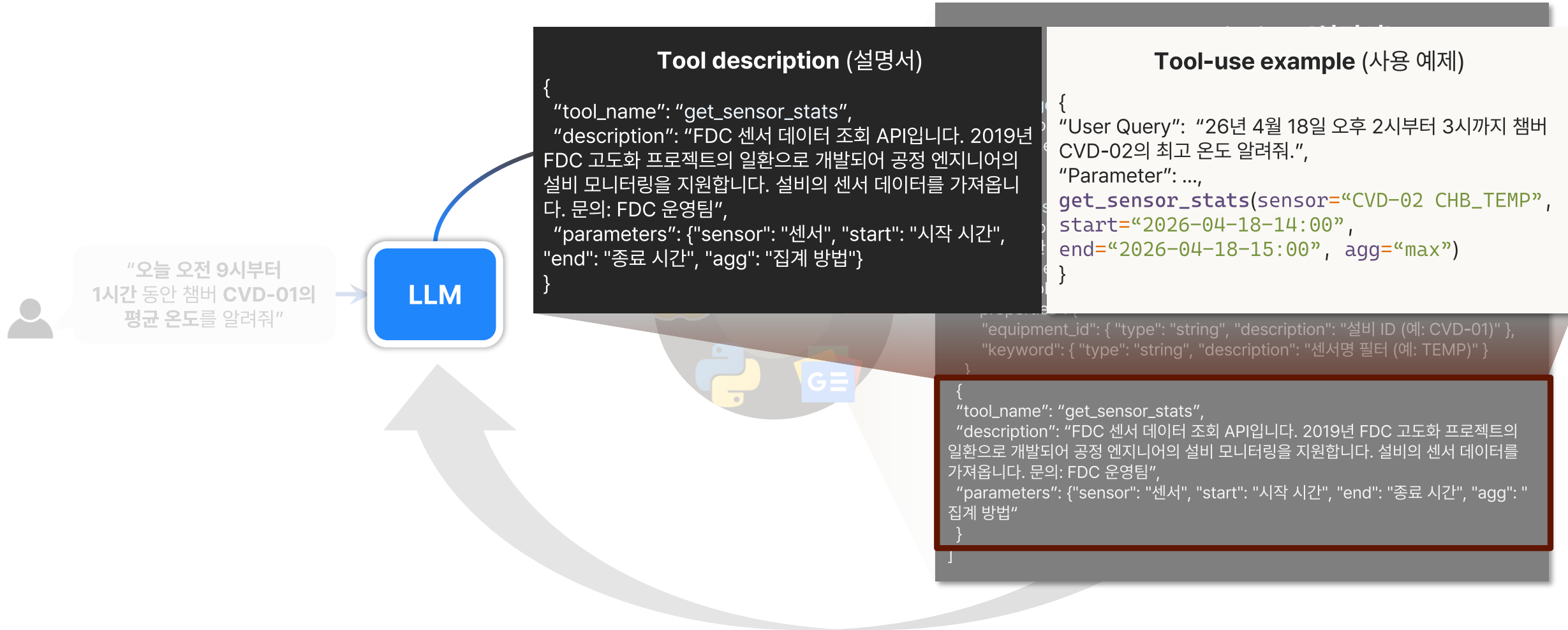
DRAFT로 해결되지 않는 것이 뭘까?

Tool-calling 성능 향상을 위한 2가지: ① **Comprehensive tool documentation**, ② **In-context demonstrations**



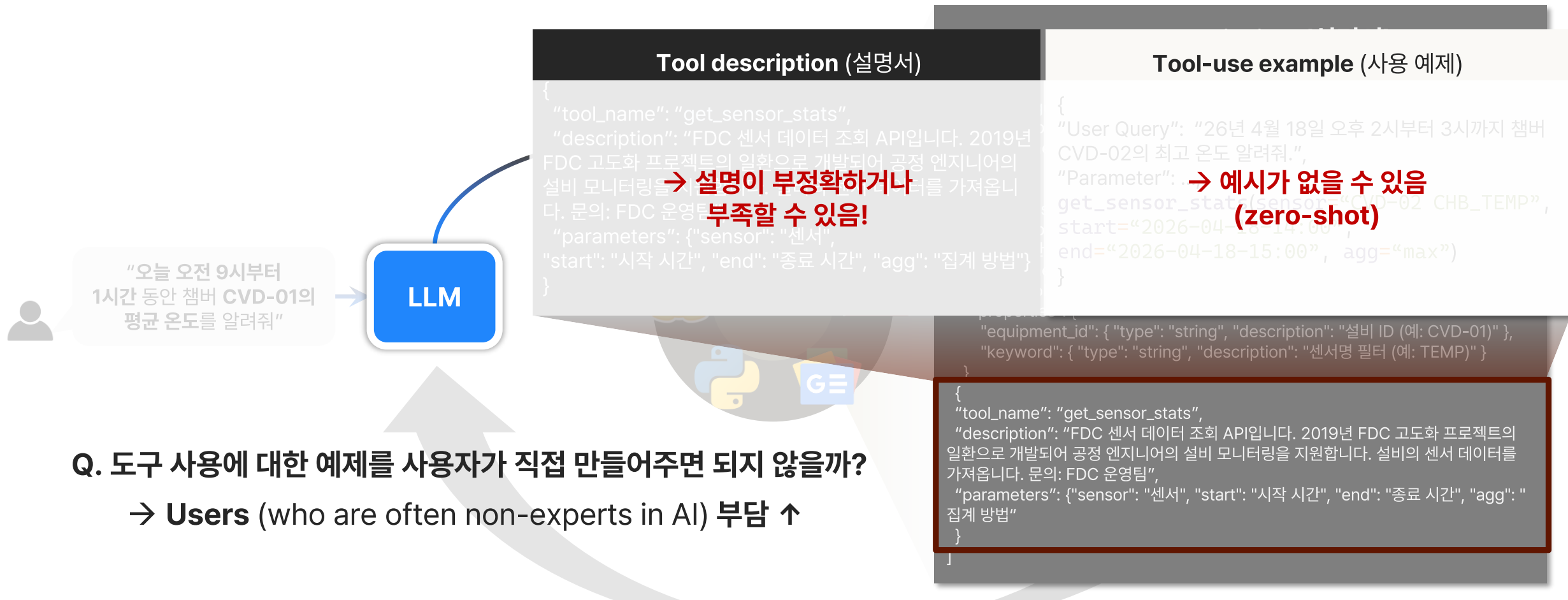
DRAFT로 해결되지 않는 것이 될까?

Tool-calling 성능 향상을 위한 2가지: ① **Comprehensive tool documentation**, ② **In-context demonstrations**



DRAFT로 해결되지 않는 것이 뭘까?

Tool-calling 성능 향상을 위한 2가지: ① Comprehensive tool documentation, ② In-context demonstrations



→ 연구목표: 양질의 tool 설명서와 사용 예제를 **자동으로 생성**할 수 있는 프레임워크를 구축해서 tool-calling 성능을 높이자

Play2Prompt (ACL Findings, 2025)

How to generate both high-quality tool documentation and tool-use demonstrations automatically?

연구가설: 모델이 스스로 시행착오를 거치며, tool 설명서와 사용 예시를 생성하고 분석하는 과정을 반복하면 도구 호출 성능을 향상 시킬 수 있을 것이다

연구방법: 2단계로 구성된 (Tool-use example generation → Tool documentation optimization) **Play2Prompt** 프레임워크 제안

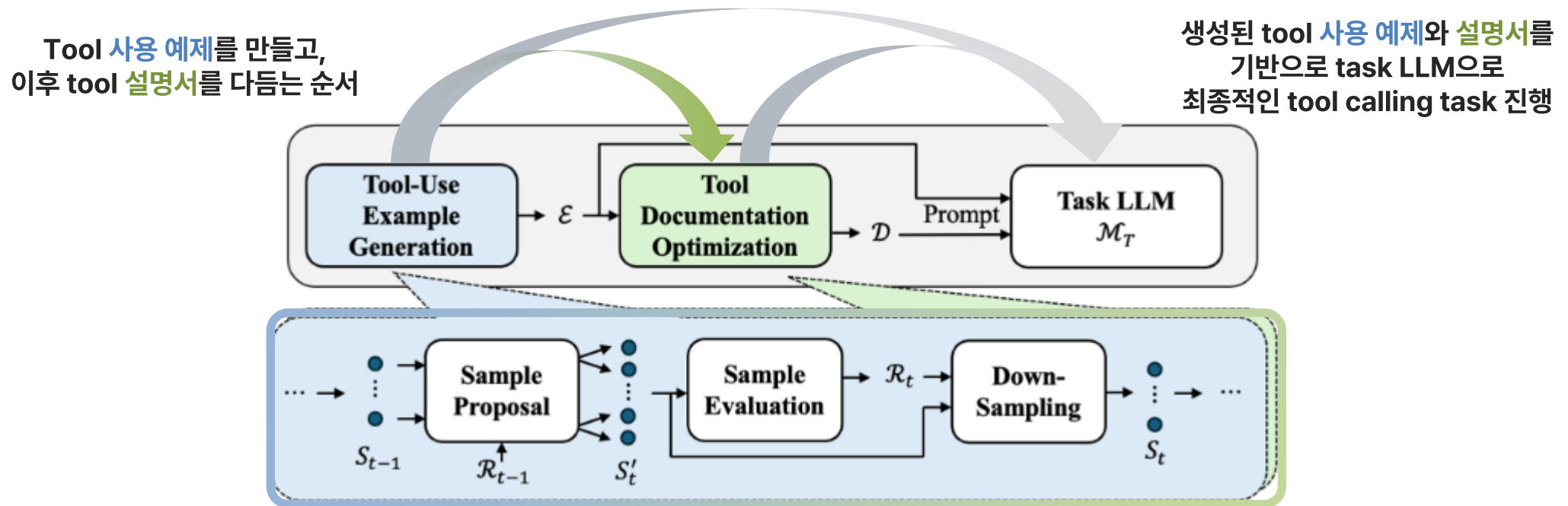


Figure 1: The Play2Prompt framework

Tool 사용 예제와 설명서 모두 3단계를 거쳐 생성
(sample proposal → sample evaluation → down-sampling)

Play2Prompt (ACL Findings, 2025)

How to generate both high-quality tool documentation and tool-use demonstrations automatically?

연구가설: 모델이 스스로 시행착오를 거치며, tool 설명서와 사용 예시를 생성하고 분석하는 과정을 반복하면 도구 호출 성능을 향상시킬 수 있을 것이다

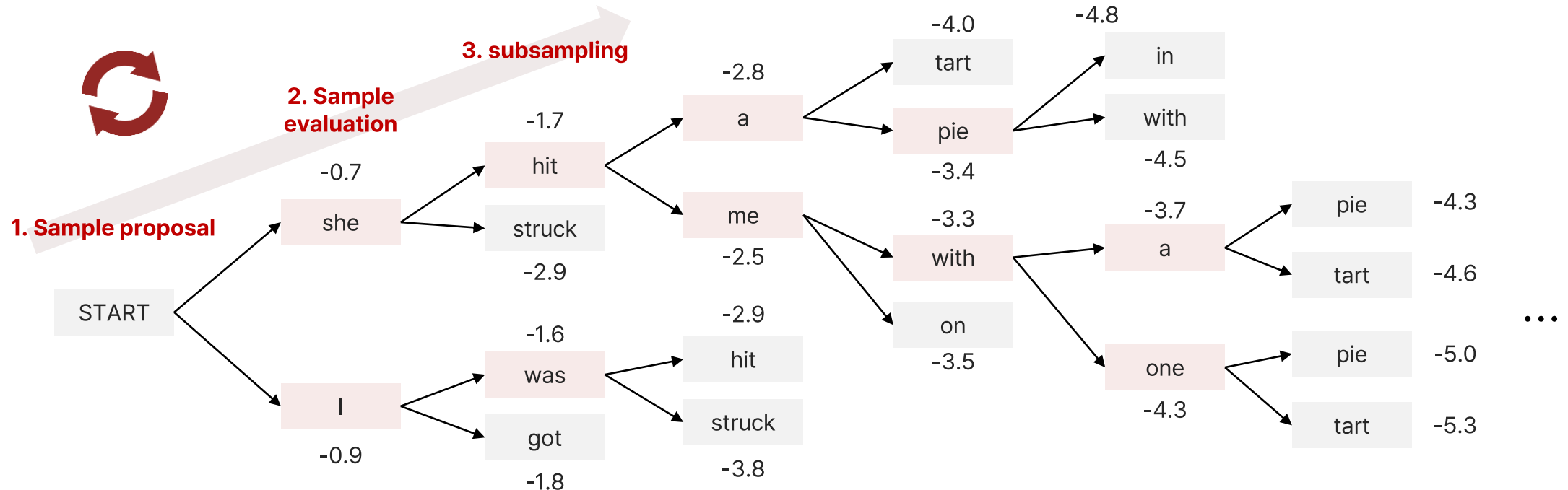
Q. 각 단계 안에서 어떻게 '더 좋은 생성 결과'를 찾을 수 있을까? → Search Framework가 필요!

Why? 설명서를 수정하거나 새로운 예제를 생성할 때, "UTC라고 명시하기"와 "한국 시간 변환 예시를 추가하기" 중 어느 표현이 LLM에 효과적인지 미리 알기 어려움.
따라서 여러 표현을 실제로 시험하며 적절한 표현을 탐색하는 기법을 이용하는 것이 중요.

e.g., Beam Search, Exhaustive Search, Greedy search, Monte Carlo, Monte Carlo Tree Search, ... 등 다양한 탐색 기법이 있음

Beam Search

각 탐색 단계에서 L 개 후보군을 생성하고, 점수(보상)가 높은 후보를 W 개 걸러내며 더 높은 품질의 문장을 출력하는 기법 ($W = 2, L = 2$ 예제)



Play2Prompt (ACL Findings, 2025)

How to generate both high-quality tool documentation and tool-use demonstrations automatically?

연구가설: 모델이 스스로 시행착오를 거치며, tool 설명서와 사용 예시를 생성하고 분석하는 과정을 반복하면 도구 호출 성능을 향상시킬 수 있을 것이다

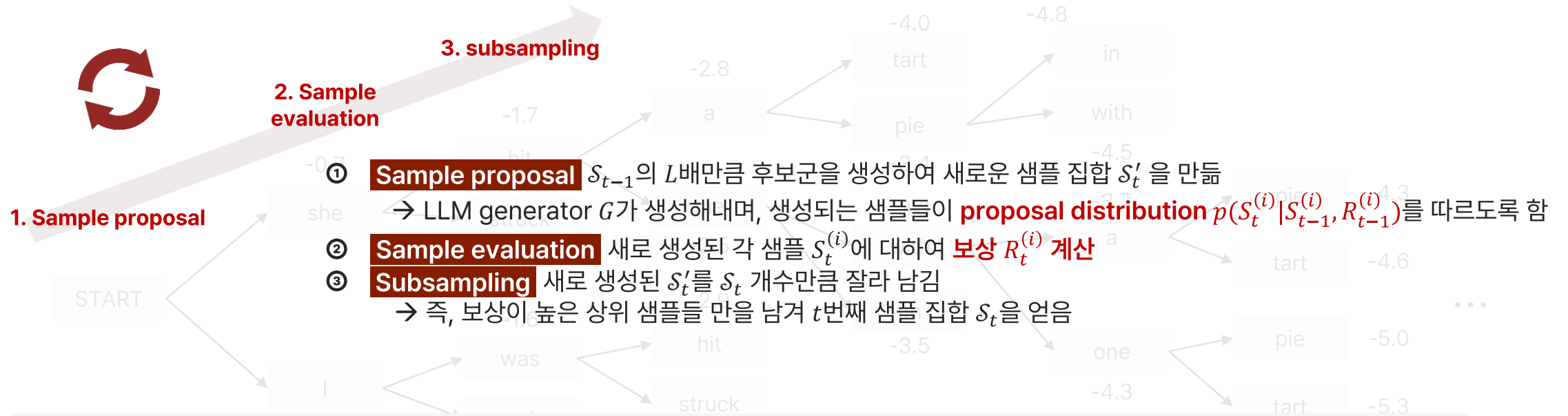
Q. 각 단계 안에서 어떻게 '더 좋은 생성 결과'를 찾을 수 있을까? → Search Framework가 필요!

Why? 설명서를 수정하거나 새로운 예제를 생성할 때, "UTC라고 명시하기"와 "한국 시간 변환 예시를 추가하기" 중 어느 표현이 LLM에 효과적인지 미리 알기 어려움.
따라서 여러 표현을 실제로 시험하며 적절한 표현을 탐색하는 기법을 이용하는 것이 중요.

e.g., Beam Search, Exhaustive Search, Greedy search, Monte Carlo, Monte Carlo Tree Search, ... 등 다양한 탐색 기법이 있음

Beam Search

각 탐색 단계에서 L 개 후보군을 생성하고, 점수(보상)가 높은 후보를 W 개 걸러내며 더 높은 품질의 문장을 출력하는 기법 ($W = 2, L = 2$ 예제)



효과적인 샘플링을 위해서는 (1) 후보를 어떻게 만들 것인가, (2) 후보를 어떻게 채점할 것인가 두 가지를 설계해야 함

Play2Prompt (ACL Findings, 2025)

1단계: 도구를 '가지고 놀면서' 예제 만들기

Q. 사용 예제를 만들고자 한다면, 어떤 질문을 할 지 떠올리고 그에 맞는 호출을 생각해야 하지 않을까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

No! tool 사용법을 하나도 모르는 상황에서, **불완전한 도구 설명서에만 의지해 질문과 호출 예시를 생성해내면 해당 호출은 실패**할 가능성이 매우 높음
즉, tool을 올바르게 호출하기 위해 **정확히 어떤 정보가 필요한지 몰라**, 적절한 질문-답변 예제를 생성하기 어려운 문제

$\mathcal{D}_0$, 초기 tool 설명서

```
{
  "tool_name": "get_sensor_stats",
  "description": "FDC 센서 데이터 조회 API입니다. 2019년 FDC 고도화 프로젝트의 일환으로 개발되어 공정 엔지니어의 설비 모니터링을 지원합니다. 설비의 센서 데이터를 가져옵니다. 문의: FDC 운영팀",
  "parameters": {"sensor": "센서", "start": "시작 시간", "end": "종료 시간", "agg": "집계 방법"}
}
```

예제 생성기 G_E

LLM

해당 도구에 대한 사용 예제를 생성해보자:

```
{"User query": "어제 오후 2시부터 3시까지 CVD-02 챔버 온도가  
평균 몇 도였는지 알려줘.",
"Parameters": {"sensor": "CVD-01", "agg": "average", ...}}
```

→ 결과: **error! Invalid sensor id!**

"sensor must be in '<equipment_id>.<tag>' format."

Play2Prompt (ACL Findings, 2025)

1단계: 도구를 '가지고 놀면서' 예제 만들기

Q. "Reverse" order?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

먼저 유효한 호출(I)을 찾고, 도구의 출력을 관찰한 뒤, 그 호출에 맞는 질문(x)과 답변(y)을 나중에 만들
→ 도구의 행동을 이해한 다음에 사용 사례를 정의하는 방식

$\mathcal{D}_0$, 초기 tool 설명서

```
{
  "tool_name": "get_sensor_stats",
  "description": "FDC 센서 데이터 조회 API입니다. 2019년 FDC 고도화 프로젝트의 일환으로 개발되어 공정 엔지니어의 설비 모니터링을 지원합니다. 설비의 센서 데이터를 가져옵니다. 문의: FDC 운영팀",
  "parameters": {"sensor": "센서", "start": "시작 시간", "end": "종료 시간", "agg": "집계 방법"}
}
```

예제 생성기 G_E

LLM

제대로 동작하는 호출을 먼저 찾자:

```
{"Parameters": {"sensor": "CVD-03.CMB_TEMP", "agg": "max", ...}}
```

호출 성공

유효한 호출을 기반으로 도구 사용 예제를 생성해보자:

```
{"User query": "2026년 4월 18일 CVD-03의 최고 온도 알려줘.", "Parameters": {"sensor": "CVD-03", "agg": "max", ...}}
```

Reverse order: 잘 모르는 상태에서 문제 해설을 만들려 하지 말고, 정답을 먼저 맞춘 뒤에 왜 맞출 수 있었는지 역으로 고민하며 해설을 쓰자

Play2Prompt (ACL Findings, 2025)

1단계: 도구를 '가지고 놀면서' 예제 만들기

Q. "유효한" 호출을 어떻게 찾을 수 있는 걸까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

먼저 **유효한 호출(I)**을 찾고, 도구의 출력을 관찰한 뒤, 그 호출에 맞는 질문(x)과 답변(y)을 나중에 만듦

m1. 호출을 생성하는 프롬프트

자리표시자(placeholder)가 아닌 실제 값을 채울 것,
필수 파라미터는 반드시 포함해볼 것,
설명서에 없는 파라미터를 지어내지 말 것, ...

m2. 실행결과를 판정하는 프롬프트

오류가 없으면 err_code: 0를 리턴할 것,
오류가 있으면 err_code: 1를 리턴할 것,
오류 원인 분석과 개선 제안까지 함께 파악할 것, ...

예제 생성기 G_E

LLM

초기 설명서만 보고 생성한 호출 (시행착오 #1)

`{"sensor": "CVD-01", "agg": "average", ...}`

호출 실패

→ 결과: **error! Invalid sensor id!**

"sensor must be in '<equipment_id>.<tag>' format."

분석: `sensor` 값이 설비 ID만으로 되어 있어서
형식 오류가 발생했구나. '<설비ID>.<태그>' 형식,
예를 들어 'CVD-01.CHB_TEMP'처럼 태그를 붙여야 한다.

Play2Prompt (ACL Findings, 2025)

1단계: 도구를 '가지고 놀면서' 예제 만들기

Q. "유효한" 호출을 어떻게 찾을 수 있는 걸까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

먼저 **유효한 호출(I)**을 찾고, 도구의 출력을 관찰한 뒤, 그 호출에 맞는 질문(x)과 답변(y)을 나중에 만듦

m1. 호출을 생성하는 프롬프트

자리표시자(placeholder)가 아닌 실제 값을 채울 것,
필수 파라미터는 반드시 포함해볼 것,
설명서에 없는 파라미터를 지어내지 말 것, ...

m2. 실행결과를 판정하는 프롬프트

오류가 없으면 err_code: 0를 리턴할 것,
오류가 있으면 err_code: 1를 리턴할 것,
오류 원인 분석과 개선 제안까지 함께 파악할 것, ...

예제 생성기 G_E

LLM

첫 번째 결과를 참고하여 재생성 (시행착오 #2)

`{"sensor": "CVD-01.CHB_TEMP", "agg": "average", ...}`

호출 실패

→ 결과: **error! Invalid agg!**

`"allowed": ["mean", "min", "max", "std", "count"]`

분석: `sensor` 형식 문제는 해결되었지만,
`agg` 값에 "average"가 허용되지 않는구나.
평균을 계산하려면 "mean"을 사용해야겠다.

Play2Prompt (ACL Findings, 2025)

1단계: 도구를 '가지고 놀면서' 예제 만들기

Q. "유효한" 호출을 어떻게 찾을 수 있는 걸까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

먼저 **유효한 호출(l)**을 찾고, 도구의 출력을 관찰한 뒤, 그 호출에 맞는 질문(x)과 답변(y)을 나중에 만듦

m1. 호출을 생성하는 프롬프트

자리표시자(placeholder)가 아닌 실제 값을 채울 것,
필수 파라미터는 반드시 포함해볼 것,
설명서에 없는 파라미터를 지어내지 말 것, ...

m2. 실행결과를 판정하는 프롬프트

오류가 없으면 err_code: 0를 리턴할 것,
오류가 있으면 err_code: 1를 리턴할 것,
오류 원인 분석과 개선 제안까지 함께 파악할 것, ...

예제 생성기 G_E

LLM

두 번째 결과를 참고하여 재생성 (시행착오 #3)

`{"sensor": "CVD-01.CHB_TEMP", "agg": "mean", ...}`

→ 결과: `{"value": 398.6, "unit": "degC", "count": 3542, "expected_count": 3600, ...}`

호출 성공

유효한 호출 1개 확보!

해당 도구에 대하여, 유효한 호출 $L - 1$ 개를
더 확보할 필요가 있음

Reject sampling: 설명서를 보고 호출을 하나 만들어 실제 실행하고, 출력과 오류 메시지를 관찰하고, 성찰한 뒤, 다음 호출을 만드는 과정을 반복

1단계: 도구를 '가지고 놀면서' 예제 만들기

Q. "유효한" 호출을 어떻게 찾을 수 있는 걸까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

2. Tool Documentation Optimization

Tool 설명서 최적화 단계: 생성된 도구 사용 예시를 활용해 tool 설명서 정교화

먼저 **유효한 호출(I)**을 찾고, 도구의 출력을 관찰한 뒤, 그 호출에 맞는 질문(x)과 답변(y)을 나중에 만듦

호출	내용	결과
I_a	`CVD-01.CHB_TEMP`, 한국 시간(+09:00) 1시간, mean	418.96 degC
I_b	`CVD-03.WALL_TEMP`, 시간대 없이 4시간, max	215.42 degC
I_c	`ETCH-02.RF_POWER`, 한국 시간 30분, std	4.20 W

동일한 방식을 거쳐, 해당 도구에 대해 $L=3$ 개의 유효한 호출을 모았다고 가정

Reject sampling: 설명서를 보고 호출을 하나 만들어 실제 실행하고, 출력과 오류 메시지를 관찰하고, 성찰한 뒤, 다음 호출을 만드는 과정을 반복

호출	내용	결과
I_a	`CVD-01.CHB_TEMP`, 한국 시간(+09:00) 1시간, mean	418.96 degC

1단계: 도구를 '가지고 놀면서' 예제 만들기

Q. 유효한 호출을 찾았으면, 이제 "적절한 예제(질의응답)"는 어떻게 만들까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

2. Tool Documentation Optimization

Tool 설명서 최적화 단계: 생성된 도구 사용 예시를 활용해 tool 설명서 정교화

먼저 유효한 호출(I)을 찾고, 도구의 출력을 관찰한 뒤, 그 호출에 맞는 **질문(x)**과 **답변(y)**을 나중에 만듦

m3. 사용자 질문(user query)을 생성하는 프롬프트

질의는 해당 도구 없이는 풀 수 없는 문제여야 함,
도구를 호출하라고 직접 요청해서는 안 됨,
호출에 들어간 모든 파라미터 값을 질의 안에 포함해야 함,
하지만 함수 이름이나 파라미터 이름을 드러내서는 안 됨,
사용자의 1인칭 관점으로 작성해야 함, ...

m4. 답변 생성용 프롬프트

도구 실행 결과에만 근거해 짧고 자연스러운 답을 써야 함,
오류가 있었다면 지어내지 말고 오류를 보고해야 함, ...

예제 생성기 G_E

LLM

I_a 기반 질문 x 생성

"get_sensor_stats로 CVD-01 평균 온도 조회해줘."

I_a 기반 답변 y 생성

"CVD-01 챔버 온도의 해당 구간 평균은 418.96°C 입니다."

Q. 만들어진 예제가 좋은지 나쁜지는 어떻게 판단할까?

호출	내용	결과
I_a	`CVD-01.CHB_TEMP`, 한국 시간(+09:00) 1시간, mean	418.96 degC

1단계: 도구를 '가지고 놀면서' 예제 만들기

Q. 유효한 호출을 찾았으면, 이제 "적절한 예제(질의응답)"는 어떻게 만들까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

2. Tool Documentation Optimization

Tool 설명서 최적화 단계: 생성된 도구 사용 예시를 활용해 tool 설명서 정교화

먼저 유효한 호출(I)을 찾고, 도구의 출력을 관찰한 뒤, 그 호출에 맞는 **질문(x)**과 **답변(y)**을 나중에 만듦

m5. 채점 기준을 정하고 보상을 생성하는 프롬프트

1점~3점 척도로 구성됨,
질의가 비어있거나 함수 이름이나 파라미터 이름을 포함하면 1점,
3문장 또는 300자를 넘거나, 호출 파라미터 값 중 하나라도 질의
에 없다면 1점, 답변이 오류 메시지면 1점, 답변만으로 해결 여부
를 판단하기 어려우면 2점, 해결되었다고 확신하면 3점

m6. 자기 성찰용 프롬프트

고득점과 저득점 질의의 패턴을 대조하고
저득점 질의를 개선할 방향을 제시하게 함

예제 생성기 G_E

LLM

I_a 기반 질문 x 생성

"get_sensor_stats로 CVD-01 평균 온도 조회해줘."

I_a 기반 답변 y 생성

"CVD-01 챔버 온도의 해당 구간 평균은 418.96°C 입니다."

$$\text{보상: } R^i = R_q^i + \lambda R_e^i$$

예제 품질 예제 난이도

호출	내용	결과
I_a	`CVD-01.CHB_TEMP`, 한국 시간(+09:00) 1시간, mean	418.96 degC

1단계: 도구를 '가지고 놀면서' 예제 만들기

Q. 유효한 호출을 찾았으면, 이제 “적절한 예제(질의응답)”는 어떻게 만들까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

먼저 유효한 호출(I)을 찾고, 도구의 출력을 관찰한 뒤, 그 호출에 맞는 **질문(x)**과 **답변(y)**을 나중에 만듦

$\mathcal{D}_0$, 초기 tool 설명서

```
{
  "tool_name": "get_sensor_stats",
  "description": "FDC 센서 데이터 조회 API입니다. 2019년 FDC 고도화 프로젝트의 일환으로 개발되어 공정 엔지니어의 설비 모니터링을 지원합니다. 설비의 센서 데이터를 가져옵니다. 문의: FDC 운영팀",
  "parameters": {"sensor": "센서", "start": "시작 시간", "end": "종료 시간", "agg": "집계 방법"}
}
```

예제 생성기 G_E



I_a 기반 질문 x 생성

생성한 예제 없이 원래 설명서 $\mathcal{D}_0$ 만 주고 이 질의를 풀게 했을 때 성능에 **마이너스**를 붙인 값

** 설명서만으로도 잘 푸는 예제는 중요하게 고려할 필요가 없음. 따라서 점수를 낮춰서 어려운 예제를 생성하도록 장려

"CVD-01 찬" $R_e^{(i)} = -\mathcal{P}\{\mathcal{M}_T(x^{(i)}; \mathcal{D}_0, \emptyset); y^{(i)}, F_k, I_k^{(i)}\}$

$$\text{보상: } R^i = R_q^i + \lambda R_e^i$$

예제 품질 예제 난이도

2. Tool Documentation Optimization

Tool 설명서 최적화 단계: 생성된 도구 사용 예시를 활용해 tool 설명서 정교화

호출	내용	결과
I_a	`CVD-01.CHB_TEMP`, 한국 시간(+09:00) 1시간, mean	418.96 degC

Play2Prompt (ACL Findings, 2025)

1단계: 도구를 '가지고 놀면서' 예제 만들기

Q. 유효한 호출을 찾았으면, 이제 “적절한 예제(질의응답)”는 어떻게 만들까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

2. Tool Documentation Optimization

Tool 설명서 최적화 단계: 생성된 도구 사용 예시를 활용해 tool 설명서 정교화

먼저 유효한 호출(I)을 찾고, 도구의 출력을 관찰한 뒤, 그 호출에 맞는 **질문(x)**과 **답변(y)**을 나중에 만듦

m5. 채점 기준을 정하고 보상을 생성하는 프롬프트

1점~3점 척도로 구성됨,
질의가 비어있거나 함수 이름이나 파라미터 이름을 포함하면 1점,
3문장 또는 300자를 넘거나, 호출 파라미터 값 중 하나라도 질의
에 없다면 1점, 답변이 오류 메시지면 1점, 답변만으로 해결 여부
를 판단하기 어려우면 2점, 해결되었다고 확신하면 3점

m6. 자기 성찰용 프롬프트

고득점과 저득점 질의의 패턴을 대조하고
저득점 질의를 개선할 방향을 제시하게 함

예제 생성기 G_E

LLM

Round #1, 보상: $R^i = R_q^i + \lambda R_e^i$ (1점-0점)

I_a 기반 질문 x 생성

“get_sensor_stats로 CVD-01 평균 온도 조회해줘.”

함수 명시,
1점

I_a 기반 답변 y 생성

“CVD-01 챔버 온도의 해당 구간 평균은 418.96°C 입니다.”

호출 실패,
-0점

성찰: 함수 이름을 언급하면 안 됨.
호출에 쓰인 식별자 (CVD-01.CHB_TEMP)와
시간 구간을 질문에 포함해야 함

Play2Prompt (ACL Findings, 2025)

호출	내용	결과
I_a	`CVD-01.CHB_TEMP`, 한국 시간(+09:00) 1시간, mean	418.96 degC

1단계: 도구를 '가지고 놀면서' 예제 만들기

Q. 유효한 호출을 찾았으면, 이제 "적절한 예제(질의응답)"는 어떻게 만들까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

2. Tool Documentation Optimization

Tool 설명서 최적화 단계: 생성된 도구 사용 예시를 활용해 tool 설명서 정교화

먼저 유효한 호출(I)을 찾고, 도구의 출력을 관찰한 뒤, 그 호출에 맞는 **질문(x)**과 **답변(y)**을 나중에 만듦

m3. 사용자 질문(user query)을 생성하는 프롬프트

질의는 해당 도구 없이는 풀 수 없는 문제여야 함,
도구를 호출하라고 직접 요청해서는 안 됨,
호출에 들어간 모든 파라미터 값을 질의 안에 포함해야 함,
하지만 함수 이름이나 파라미터 이름을 드러내서는 안 됨,
사용자의 1인칭 관점으로 작성해야 함, ...

m4. 답변 생성용 프롬프트

도구 실행 결과에만 근거해 짧고 자연스러운 답을 써야 함,
오류가 있었다면 지어내지 말고 오류를 보고해야 함, ...

+ **성찰**: '함수 이름을 언급하면 안 됨. 호출에 쓰인 식별자 (CVD-01.CHB_TEMP)와 시간 구간을 질문에 포함해야 함'

예제 생성기 G_E

LLM

Round #2, 보상: $R^i = R_q^i + \lambda R_e^i$ (1점-0점)

I_a 기반 질문 x 생성

"CVD-01 챔버 온도가 어제 오후에 평균 몇 도였나요?"

파라미터 누락,
1점

I_a 기반 답변 y 생성

"어제 오후 CVD-01 챔버 온도 평균은 418.96 °C 입니다."

호출 실패,
-0점

성찰: 모호한 표현 ("어제 오후")을 피하고
정확한 날짜와 시작, 종료 시각을 쓸 것,
센서 식별자 전체를 포함할 것.

호출	내용	결과
I_a	`CVD-01.CHB_TEMP`, 한국 시간(+09:00) 1시간, mean	418.96 degC

Play2Prompt (ACL Findings, 2025)

1단계: 도구를 '가지고 놀면서' 예제 만들기

Q. 유효한 호출을 찾았으면, 이제 “적절한 예제(질의응답)”는 어떻게 만들까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

2. Tool Documentation Optimization

Tool 설명서 최적화 단계: 생성된 도구 사용 예시를 활용해 tool 설명서 정교화

먼저 유효한 호출(I)을 찾고, 도구의 출력을 관찰한 뒤, 그 호출에 맞는 **질문(x)**과 **답변(y)**을 나중에 만듦

m3. 사용자 질문(user query)을 생성하는 프롬프트

질의는 해당 도구 없이는 풀 수 없는 문제여야 함,
도구를 호출하라고 직접 요청해서는 안 됨,
호출에 들어간 모든 파라미터 값을 질의 안에 포함해야 함,
하지만 함수 이름이나 파라미터 이름을 드러내서는 안 됨,
사용자의 1인칭 관점으로 작성해야 함, ...

m4. 답변 생성용 프롬프트

도구 실행 결과에만 근거해 짧고 자연스러운 답을 써야 함,
오류가 있었다면 지어내지 말고 오류를 보고해야 함, ...

+ **성찰**: ‘모호한 표현을 피하고 정확한 날짜와 시작, 종료 시각을
쓸 것, 센서 식별자 전체를 포함할 것.’

예제 생성기 G_E

LLM

Round #3, 보상: $R^i = R_q^i + \lambda R_e^i$ (3점-0점)

I_a 기반 질문 x 생성

“어제 (10월 1일) 한국 시간 오후 2시부터 3시까지 CVD-01.CHB_TEMP 센서 온도의 평균이 어느정도였어?”

3점

I_a 기반 답변 y 생성

“어제 오후 2시~3시 CVD-01 챔버 온도
평균은 418.96°C 입니다.”

호출 실패,
-0점

Self-reflection step 수(N_E)가 3번이라면, 여기서 종료.
이후 다음 후보들에 대해서 동일 과정 반복

Play2Prompt (ACL Findings, 2025)

2단계: 만들어진 예제로 설명서를 채점하며 고치기

Q. 설명서 후보는 어떻게 만들 수 있을까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

2. Tool Documentation Optimization

Tool 설명서 최적화 단계: 생성된 도구 사용 예시를 활용해 tool 설명서 정교화

설명서를 생성하는 LLM에게 **tool-use** 예제를 생성하는 LLM이 저지른 오류들을 보여주고, 이를 기반으로 수정을 유도

$\mathcal{D}_0$, 초기 tool 설명서

```
{
  "tool_name": "get_sensor_stats",
  "description": "FDC 센서 데이터 조회 API입니다. 2019년 FDC 고도화 프로젝트의 일환으로 개발되어 공정 엔지니어의 설비 모니터링을 지원합니다. 설비의 센서 데이터를 가져옵니다. 문의: FDC 운영팀",
  "parameters": {"sensor": "센서", "start": "시작 시간", "end": "종료 시간", "agg": "집계 방법"}
}
```

LLM

잘못된 호출, 잘못된 예제 등
그동안 저지른 오류들
(설명서의 부족한 부분을 보여줌)

설명서 생성기 G_D

LLM

$\mathcal{D}_1$, 수정된 tool 설명서

```
{
  "tool_name": "get_sensor_stats",
  "description": "...(수정된 내용)...",
  "parameters": {"sensor": "센서", "start": "시작 시간", "end": "종료 시간", "agg": "집계 방법"}
}
```

Play2Prompt (ACL Findings, 2025)

2단계: 만들어진 예제로 설명서를 채점하며 고치기

Q. 정답이 없는데, 수정된 설명서가 좋아졌는지 아닌지 어떻게 알 수 있을까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

2. Tool Documentation Optimization

Tool 설명서 최적화 단계: 생성된 도구 사용 예시를 활용해 tool 설명서 정교화

1단계에서 만들었던 예제들을 성능을 검증할 수 있는 데이터셋으로 활용

$\mathcal{D}_1$, 수정된 tool 설명서

```
{
  "tool_name": "get_sensor_stats",
  "description": "...(수정된 내용)...",
  "parameters": {"sensor": "센서", "start": "시작 시간", "end": "종료 시간", "agg": "집계 방법"}
}
```

LLM

앞서 생성한 user query로 검증

Q. "어제 오후 2시부터 3시까지 CVD-01.CHB_TEMP 센서 온도 평균이 ..."

Task LLM

LLM

$$R^{(i)} = \mathbb{E}_{(x,F,I,y) \in \mathcal{E}} [\mathcal{P}\{\mathcal{M}_T(x; \mathcal{D}^{(i)}, \emptyset); y, F, I\}]$$

→ 수정된 설명서만 주어진 때,
task LLM이 앞서 생성한 user query를 얼마나 잘 풀어내는지 채점

Play2Prompt (ACL Findings, 2025)

2단계: 만들어진 예제로 설명서를 채점하며 고치기

Q. 정답이 없는데, 수정된 설명서가 좋아졌는지 아닌지 어떻게 알 수 있을까?

1. Tool-Use Example Generation

Tool 사용 예제 생성 단계: 도구의 입출력 동작 탐색을 위한 시행착오 과정

2. Tool Documentation Optimization

Tool 설명서 최적화 단계: 생성된 도구 사용 예시를 활용해 tool 설명서 정교화

1단계에서 만들었던 예제들을 성능을 검증할 수 있는 데이터셋으로 활용

후보 A

평균 조회에는 "mean", 최댓값 조회에는 "max"를 사용한다.

후보 B

평균 조회에는 "mean"을 사용한다. "average"는 허용되지 않는다.

후보 C

`agg`에는 평균, 최대, 최소, 표준편차, 중앙값을 계산할 수 있다.



앞서 생성한 user query로 검증

Q. "어제 오후 2시부터 3시까지 CVD-01.CHB_TEMP 센서 온도 평균이 ..."

Task LLM



$$R^{(i)} = \mathbb{E}_{(x,F,I,y) \in \mathcal{E}} [\mathcal{P}\{\mathcal{M}_T(x; \mathcal{D}^{(i)}, \emptyset); y, F, I\}]$$

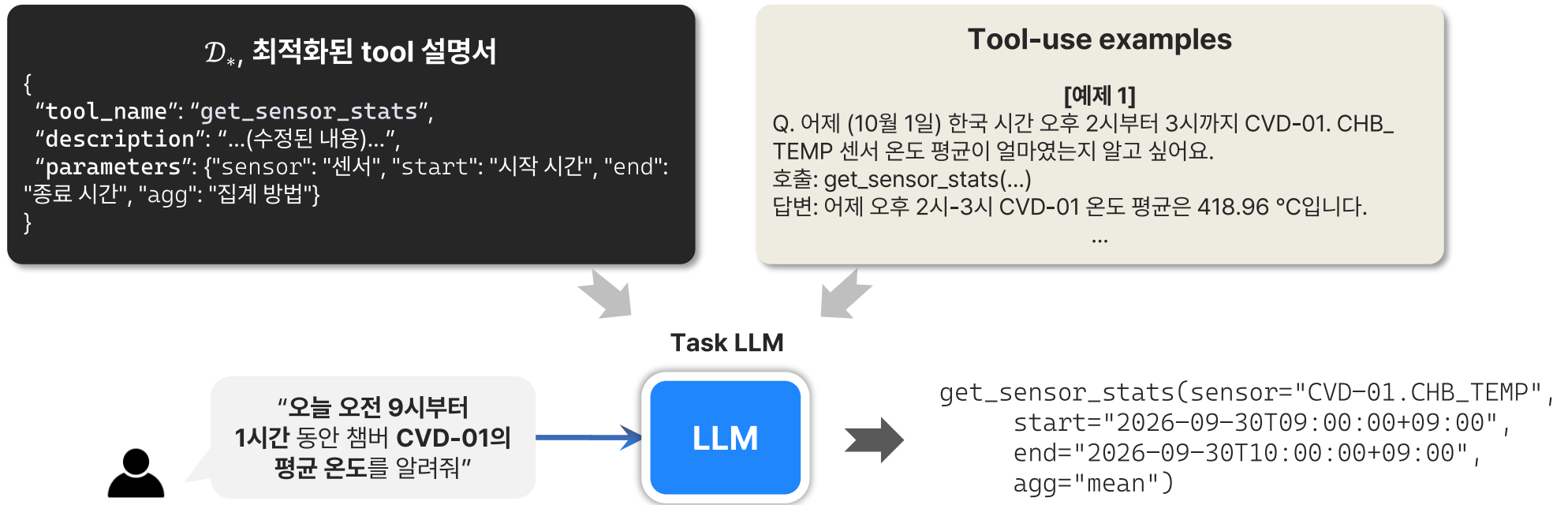
→ 수정된 설명서만 주어졌을 때,
task LLM이 앞서 생성한 user query를 얼마나 잘 풀어내는지 채점

후보	평가 질문 5개 중 성공	보상
A	3개	0.6
B	5개	1.0
C	2개	0.4

보상을 최대화하고, 보상 간의 편차를 최소화하는 것을 목표로 tool documentation 최적화 수행

Play2Prompt (ACL Findings, 2025)

3단계: 최종 추론



설명서로 tool 호출 규칙을 이해하고, 예제를 통해 “구체적으로 어떻게 쓰는지”를 확인한 뒤 올바르게 호출
▶ ‘한국 시간’임을 명시하지 않아도 한국 시간으로 변환한 것은, 예제 1의 패턴을 따라한 덕분! (설명서와 예제가 서로 보완하는 모습)

Play2Prompt (ACL Findings, 2025)

Q. 제안 방법론의 효과를 어떻게 평가할 수 있을까?
데이터셋: BFCL과 StableToolBench 두 벤치마크를 사용
DRAFT backbone model: GPT-4o

Berkely Function-Calling Leaderboard (BFCL)

범주	문제 수	BFCL 정의	예시
Simple python	100	함수 설명서 하나, 호출 하나	삼각형 넓이 구해줘
Simple REST	70	실제 REST API 호출 하나	서울 날씨 알려줘
Multiple	50	한 질문을 해결하기 위해 2~4개 함수 설명서 중 하나만 골라 호출	날씨, 환율, 번역 함수 중 알맞은 하나를 선택
Parallel	50	한 질문으로 같은 함수를 여러 번 동시 호출	서울, 도쿄, 파리 날씨 알려줘
Multiple-Parallel	40	여러 함수 설명서, 각 함수를 1번 이상 호출	서울 날씨와 100달러 환율 알려줘.

생성된 호출 실행 결과가 얼마나 정확한지 **accuracy** 계산

Avg. = 다섯가지 범주 정확도의 단순 평균
Weighted Avg. = 두 가지 'simple' 범주에 0.5 가중치를 둔 평균

Tool calling 시 호출 인자 수준에서 정확도 향상이 가능한지 확인

기본적인 tool calling 모델들 (GPT, ToolACE, Hammer)과 비교,
DRAFT와 같은 설명서 최적화 방법과 비교하지 않음

StableToolBench

범주: 날씨 (weather)

- 도구: OpenWeather 서비스
- API: 현재 날씨 조회
- API: 5일 예보 조회 (X)
- API: 대기질 조회

범주: 금융 (finance)

- 도구: 환율 서비스
- API: 실시간 환율 조회
- API: 통화 변환

→ 기존 ToolBench에서 고장나거나 동작하지 않는 API를 제외하고,
만일 실제 API가 동작하지 않으면 LLM이 대신 그럴듯한 응답을 생성

Solvable pass rate

LLM Judge (LlaMa-3.3-70B-Instruct)가 답변이 적절했는지를 판단,
해결 가능한 질의 중 해결되었다고 판정된 비율을 계산

실전 API 호출에서도 다른 방법론 대비 우수한 성능 달성이 가능한지를 확인

두 가지 설명서 최적화 기법 (EasyTool, DRAFT)과 비교

Play2Prompt (ACL Findings, 2025)

Q. 제안 방법론은 정말 효과가 있을까?

BFCL Executable: 함수를 올바른 인자로 호출 가능한가? → Yes

StableToolBench: 실제 API 호출을 필요로 하는 복잡한 현실 세계에서 동작하는가? → Yes

Berkely Function-Calling Leaderboard (BFCL)

Base Model	Method	Simple-Python	Simple-REST	Multiple	Parallel	Multiple-Parallel	Weighted Avg	Avg
LLaMA-8B	Prompting	96.0	70.0	96.0	90.0	77.5	86.6	85.9
	+PLAY2PROMPT	97.0	87.1	96.0	92.0	92.5	93.1	92.9
LLaMA-70B	Prompting	100.0	91.4	96.0	84.0	82.5	89.6	90.8
	+PLAY2PROMPT	100.0	91.4	98.0	88.0	95.0	94.2	94.5
GPT-3.5	Function-calling	97.0	94.3	90.0	86.0	67.5	84.8	87.0
	+PLAY2PROMPT	98.0	95.7	94.0	90.0	85.0	91.5	92.5
GPT-4o	Function-calling	98.0	98.6	94.0	94.0	77.5	91.0	92.4
	+PLAY2PROMPT	99.0	95.7	98.0	92.0	90.0	94.3	94.9
ToolACE-8B	Function-calling	96.0	92.9	92.0	86.0	70.0	85.6	87.4
	+PLAY2PROMPT	95.0	90.0	94.0	90.0	82.5	89.8	90.3
Hammer-7B	Function-calling	96.0	72.9	88.0	86.0	70.0	82.1	82.6
	+PLAY2PROMPT	95.0	82.8	92.0	86.0	80.0	86.7	87.2

Table 1: Results on BFCL Executable. Accuracy scores are shown.

- 1) 모든 방법론에 대해서, **제안 방법론(P2P)**을 수행했을 때 **성능 향상**
- 2) 특히나 여러 도구를 여러 번 반복 호출해야 하는 **어려운 상황** (Multiple-Parallel)에서 **가장 높은 성능 향상** 달성 (+10~17.5)
- 3) **작은 모델 LLaMA-8B + P2P (93.1)**이 **큰 모델 GPT-4o 기본 (91.0)**과 LLaMA-70B (89.6)보다 **높은 성능** (연구 분야의 중요성 확인)

StableToolBench

Base Model	Method	I1-Inst	I1-Cat	I1-Tool	I2-Inst	I2-Cat	I3-Inst	Avg
LLaMA-8B	ReAct	50.6	59.3	53.2	58.0	61.3	52.3	55.8
	ReAct+EasyTool	54.7	58.9	57.9	56.1	64.2	48.1	56.7
	ReAct+DRAFT	-	-	-	-	62.6	57.9	-
	ReAct+PLAY2PROMPT	56.6	65.7	60.9	62.5	63.5	63.8	62.2
LLaMA-70B	ReAct	58.4	68.3	61.1	63.1	63.2	64.3	63.1
	ReAct+EasyTool	59.0	70.3	63.3	67.6	70.6	63.0	65.6
	ReAct+DRAFT	-	-	-	-	68.5	62.3	-
	ReAct+PLAY2PROMPT	67.5	73.6	64.1	66.5	72.8	70.7	69.2
GPT-3.5	ReAct	56.0	64.6	67.4	56.0	63.4	57.4	60.7
	ReAct+EasyTool	57.3	61.4	69.5	61.5	68.3	60.7	63.1
	ReAct+DRAFT	-	-	-	-	68.3	56.1	-
	ReAct+PLAY2PROMPT	61.1	66.6	66.2	67.0	68.3	66.3	65.9
GPT-4o	ReAct	54.0	69.7	66.1	59.8	65.3	61.7	62.8
	ReAct+EasyTool	49.9	69.0	65.2	61.3	65.3	52.5	60.5
	ReAct+DRAFT	-	-	-	-	62.1	63.9	-
	ReAct+PLAY2PROMPT	60.7	68.7	71.9	58.8	63.7	65.6	64.9

Table 2: Results on StableToolBench. Solvable pass rates are shown.

- 1) 다른 설명서 최적화 방법론 대비, P2P 적용 시 가장 큰 성능 향상되며, 가장 어려운 벤치마크인 I3-Inst에서 향상 폭이 가장 큼
→ 현실과 가까운 어려운 문제일 수록 설명서를 잘 이해하는 것이 중요
- 2) 그러나 오히려 P2P 적용 시 baseline 대비 성능이 하락한 경우도 있음

Play2Prompt (ACL Findings, 2025)

Q. 초기 설명서가 엉망진창인 어려운 상황에서도 강건하게 동작할까?

BFCL Executable 데이터셋의 초기 설명서는 비교적 잘 적힌 형태이기 때문에, P2P의 효과를 온전하게 확인하기는 어려움
따라서 함수 파라미터에 대한 설명을 확률 p 만큼 지워가며 성능 확인

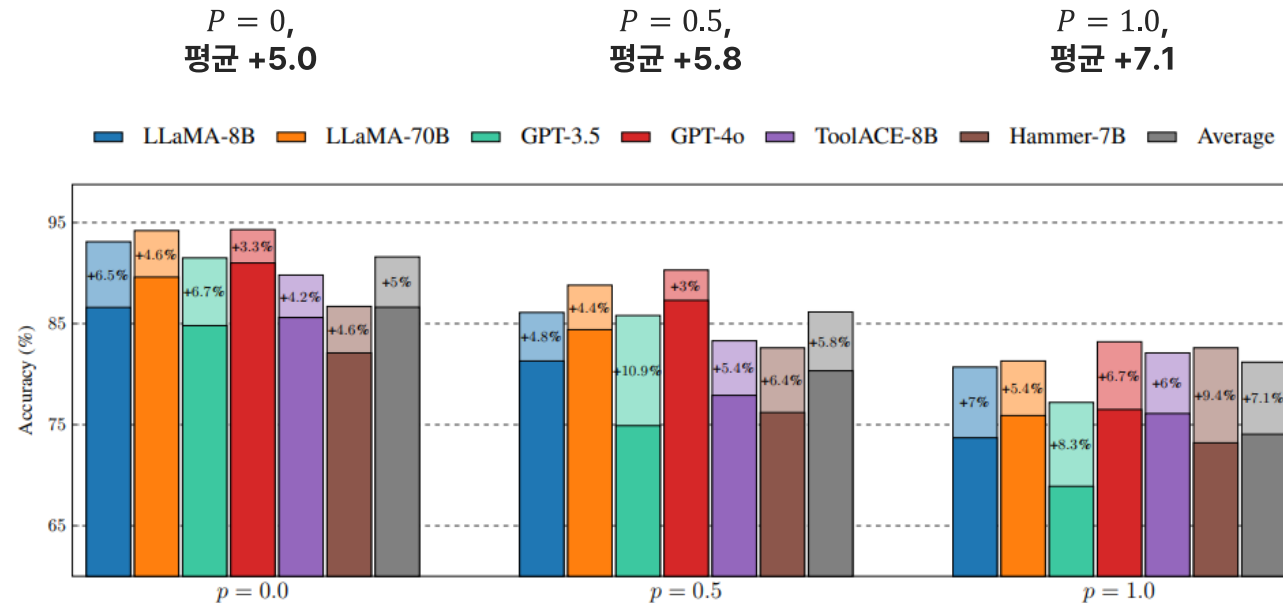


Figure 2: Average (weighted) accuracy improvements with PLAY2PROMPT on BFCL, for different parameter description dropout p .

설명서가 나빠질 수록 제안 방법론의 향상 폭이 커지는 것을 확인
($p=1.0$ 은 모든 파라미터 설명이 사라진 상태)
→ 누락된 파라미터 정보를 보완하는 데 매우 효과적임을 시사

Play2Prompt (ACL Findings, 2025)

Q. Tool documentation과 Tool-use example 중 어느 것이 더 효과적일까?

Play2Prompt는 DRAFT와 달리 tool-use examples까지 고려하며, 해당 접근 방식에 대한 효과를 정량적으로 확인

Base Model	Method	BFCL	STB
LLaMA-8B	Baseline	85.9	55.8
	+P2P-Desc	89.9	57.9
	+P2P-Demo	90.8	59.5
	+P2P	92.9	62.2
LLaMA-70B	Baseline	90.8	63.1
	+P2P-Desc	93.6	64.4
	+P2P-Demo	92.7	67.8
	+P2P	94.5	69.2
GPT-3.5	Baseline	87.0	60.7
	+P2P-Desc	89.0	63.5
	+P2P-Demo	91.9	65.3
	+P2P	92.5	65.9

Table 3: Ablation on PLAY2PROMPT (P2P) using generated example demonstrations only (P2P-Demo) and generated descriptions only (P2P-Desc). Scores indicate average accuracy for BFCL and average solvable pass rate for StableToolBench (STB).

P2P-Dsec: 개선된 설명서만 쓰고 예제는 사용하지 않음

P2P-Demo: 새로 만든 예제를 초기 설명서와 함께 사용

- 1) 예제(demonstration)만 사용하는 것이 설명서(description)만 쓰는 것 보다 효과가 크고, 둘을 합칠 때 항상 더 좋은 성능 도출
- 2) 특히나 StableToolBench는 도구 당 오로지 예제 1개만을 이용하고도 baseline 대비 높은 성능 향상을 달성
- 3) DRAFT에서 간과한 tool-use example 사용의 중요성을 확인

Play2Prompt (ACL Findings, 2025)

Q. 어려운 예제를 고르는 것이 정말 도움이 될까?

예제에 대한 보상으로 R_e (문제 난이도)를 역으로 고려해, 모델이 어려운 예제를 선호하도록 설계

Base Model	Method	STB	
LLaMA-8B	ReAct	55.8	} +77%
	+PLAY2PROMPT-Easy	60.7	
	+PLAY2PROMPT	62.2	
LLaMA-70B	ReAct	63.1	} +84%
	+PLAY2PROMPT-Easy	68.2	
	+PLAY2PROMPT	69.2	

Table 4: Ablation on demonstration difficulty for PLAY2PROMPT. We report average solvable pass rates.

Play2Prompt-Easy: 예제 생성 시 R_e 를 빼고 품질 점수 R_q 만 사용

- 1) 모델 파라미터 크기와 관계 없이, 어려운 예제를 무시하면 성능이 평균 1~2점 떨어지므로, 난이도 높은 예제가 도구 사용을 향상시키는 것을 정량적으로 확인
- 2) 그러나 성능 향상의 대부분은 예제 생성 그 자체에 있음

Conclusion

DRAFT vs. Play2Prompt

“사람이 쓴 설명서는 LLM에게는 부족하며, LLM이 도구를 직접 써봐야 적절한 수정이 가능하다”는 같은 문제의식에서 출발

DRAFT

도구가 실제로 어떻게 동작하는지 설명서에 충실히 담아보자

- ✓ 최적화 대상? → tool documentation
- ✓ 탐색 순서? → 질의를 먼저 만들고 호출
- ✓ 설명서 수정 근거? → 도구 호출 결과를 analyzer가 자연어로 해석
- ✓ 탐색 구조? → 한 줄로 반복 수정
- ✓ 다양성 확보? → 임베딩 유사도 제약 + self-reflection
- ✓ 종료 조건? → 변화량을 정량화하고 임계 값에 따라 종료
- ✓ Task LLM의 역할? → 없음
- ✓ 최적화 모델? → GPT-4o

Play2Prompt

Task LLM이 문제를 잘 풀 수 있는 방향으로 설명서를 고쳐보자

- ✓ 최적화 대상? → tool documentation + too-use examples
- ✓ 탐색 순서? → 호출을 먼저 찾고, 질의를 역으로 생성
- ✓ 설명서 수정 근거? → task LLM이 저지른 오류와 점수
- ✓ 탐색 구조? → Beam search
- ✓ 다양성 확보? → Beam search + 프롬프트로 지시
- ✓ 종료 조건? → 고정 깊이까지 탐색 후 최고 점수로 선택
- ✓ Task LLM의 역할? → 생성 결과(예제)의 품질을 채점
- ✓ 최적화 모델? → LLaMA-3.1-8B

Thank You