Planning and Editing What You Retrieve for Enhanced Tool Learning

Tenghao Huang[1] **Dongwon Jung**[1] **Muhao Chen**[2]
[1]University of Southern California
[2]University of California, Davis
{tenghaoh, dongwonj}@usc.edu;
muhchen@ucdavis.edu;

Abstract

Recent advancements in integrating external tools with Large Language Models (LLMs) have opened new frontiers, with applications in mathematical reasoning, code generators, and smart assistants. However, existing methods, relying on simple one-time retrieval strategies, fall short on effectively and accurately shortlisting relevant tools. This paper introduces a novel PLUTO (Planning, Learning, and Understanding for TOols) approach, encompassing "Plan-and-Retrieve (P&R)" and "Edit-and-Ground (E&G)" paradigms. The P&R paradigm consists of a neural retrieval module for shortlisting relevant tools and an LLM-based query planner that decomposes complex queries into actionable tasks, enhancing the effectiveness of tool utilization. The E&G paradigm utilizes LLMs to enrich tool descriptions based on user scenarios, bridging the gap between user queries and tool functionalities. Experiment results demonstrate that these paradigms significantly improve the recall and NDCG in tool retrieval tasks, significantly surpassing current state-of-the-art models.

1 Introduction

The community has shown increasing interest in integrating external tools and interfaces with LLMs since tools often provide complementary functionalities in complex tasks such as dialogues (Bubeck et al., 2023), mathematical reasoning (Lu et al., 2022), and code generation (Yadav et al., 2023). To realize tool augmentation, LLM systems typically employ a retriever mechanism to select relevant tools from a candidate pool and write function API calls based on the retrieved tools. The introduction of external tools also allows LLMs to address complicated user queries. Schick et al. 2023 show that LLMs, incorporating simple tools, achieve better performance on downstream tasks. Gupta and Kembhavi 2023 attempt to solve compositional visual tasks via image processing modules and language-instructed computer vision models. More recently, the integration of LLMs and tools empower LLMs, opening up new possibilities in areas like scientific discovery (Yang et al., 2023), automated efficiency, and smart assistant applications (Shu et al., 2022).

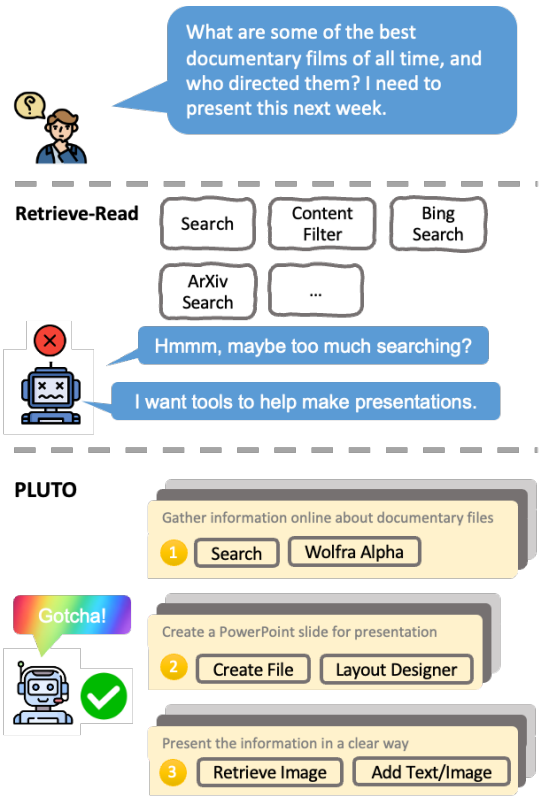

Figure 1: Comparison between conventional Retrieve-and-Read and PLUTO paradigm. Unlike the conventional one-time Retrieve-and-Read paradigm that may lead to retrieving an ineffective set of tools, PLUTo efficiently parses a complex query and distills it into actionable sub-queries that facilitate accurate retrieval of appropriate tools.

Nonetheless, emergent approaches for LLMs with tool integration present several distinct challenges. One primary concern is that current LLM agents still adopt simple retrieval-and-read strategies (Patil et al., 2023; Qin et al., 2023), lacking

the dynamic adaptability required for addressing complex queries. As shown in Fig. 1, the conventional Retrieve-and-Read paradigm, solely relying heavily on similarity matching, falls short of retrieving diverse types of tools to address a complex user query. This limitation is further exacerbated by the semantic gap between user queries and tool descriptions. Particularly, user queries can be ambiguous and complex, often requiring a deep understanding of the user's intent and the context of the query (Kulkarni et al., 2023). On the other hand, human-written tool descriptions can be abstract and lack essential details for deciding their utilities, leading to a mismatch between what the user needs and what the tool is perceived to offer. Additionally, current models tend to finetune on static tools, posing challenges to their robustness in the ever-evolving tool environment where new tools emerge and existing ones become obsolete (Lübke et al., 2019). There is limited research on retrieval enhancement strategies in non-finetuned settings. These gaps highlight crucial areas for future research and development in LLM and tool integration.

In this paper, we leverage LLM's world knowledge and reasoning ability to augment the retrieval and utility of tools in response to complex user queries, by designing a novel framework PLUTo (Planning, Learning, and Understanding for TOols) [1]. Our first contribution is the introduction of a novel *Plan-and-Retrieve* for tool integration. While prior Retrieve-and-Read approaches only retrieve once at the beginning, our *Plan-and-Retrieve* paradigm is designed to adaptively adjust its strategies based on the outcomes of its self-evaluations, ensuring a continuous refinement of the tool selection process. This paradigm is structured into two core modules. The first module, the retriever, leverages neural (dense) retrieval techniques (Karpukhin et al., 2020) and LM-likelihood scoring mechanisms (Song et al., 2023a) to efficiently shortlist relevant tools from a vast pool of candidates in response to a user query. This process ensures that the most pertinent tools are identified quickly, laying a foundation for more effective tool utilization. Inspired by recent advancements of adaptive retrieval-augmented generation (RAG; Jiang et al. 2023; Yoran et al. 2023), we design an LLM-based query planner that autoregressively decomposes complex user queries into manageable, task-oriented actions as the second module. Following the decompositions, the query planner selects the most suitable ones from the retrieved tools. It goes further by evaluating the effectiveness of selected tools and proposing the next action toward addressing the user query. This *Plan-and-Retrieve* paradigm operates dynamically, embodying a sophisticated feedback loop that interlinks the retrieval of tools with subsequent refinement, evaluation, and planning stages.

Our second contribution is the proposal of *Edit-and-Ground* paradigm that utilizes user queries' rich contextual information and LLM's extensive world knowledge for enriching descriptions of tool functionalities. Research has shown that informed tool documentations can enhance the interaction between LLMs and tools (Hsieh et al., 2023). However, documenting tool functionalities at scale can be tedious for humans. Yang et al. 2023 show LLMs can follow instructions and optimize real-world applications. Leveraging the optimization ability of the LLM, our tool-grounding agent optimizes under-informative tool descriptions by learning and abstracting information from tools' user scenarios. By editing tool descriptions to make them more aligned with tools' user scenarios, the agent bridges the gap between user queries and tool functionalities, enhancing the overall effectiveness of tool retrieval and usage.

In conclusion, this paper advances the field of tool integration with LLMs by introducing the novel Plan-and-Retrieve and Edit-and-Ground paradigms. Experiments show that our paradigms improve the recall and NDCG of tool retrieval tasks, significantly outperforming current state-of-the-art (SOTA). Our downstream evaluation suggests that the improvement gained during the retrieval phase, such as higher accuracy and relevance in responses, significantly contribute to successfully addressing the user queries.

2 Related Works

Retrieval-Augmented LLM. Early studies on Retrieval-Augmented LLMs typically incorporate embeddings of retrieved passages as a part of the latent representation of the LM (Chen et al., 2017; Lee et al., 2019). More recent works like REALM (Guu et al., 2020) and RAG (Lewis et al., 2021) have demonstrated the effectiveness of in-context augmentation and its improvement on knowledge-

[1]Code is available at https://github.com/tenghaohuang/PLUTo

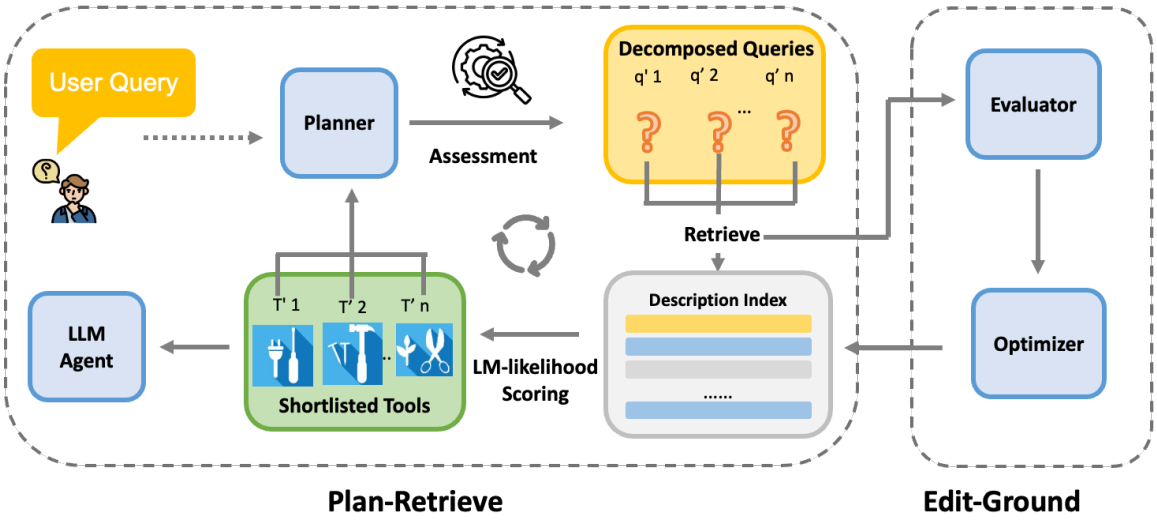

Figure 2: An overview of the PLUTO approach.

intensive tasks. There is also work (Mallen et al., 2023) that explores how Chain-of-Thought (CoT) could guide a multi-turn Retrieve-and-Read process to solve open-domain questions and perform fact verification.

However, the massive action space and tool functionality variance in tool-oriented tasks pose challenges to LLMs during planning. An erroneous step in planning can lead to a faulty loop, such as continually calling a tool in the wrong way or hallucinating non-existing tools. Our Plan-and-Retrieve paradigm, employing furtherest planning assessment (Zhu et al., 2023), enforces reasonable and goal-oriented decompositions of user queries. The recently proposed ReAct framework (Yao et al., 2022) asks LLM to plan future actions based on its observation of environments. In the context of tool-oriented tasks, the plan builds upon the execution results of retrieved tools. Such practice running and verifying each tool at retrieval time can be expensive and time-consuming at scale. In contrast, our Plan-and-Retrieve paradigm fully leverages LLM's internal representation of world knowledge to propose plans in response to user queries, therefore guaranteeing both time and cost efficiency as an execution-free paradigm.

Tool Learning. Tool learning refers to the process where LLMs not only process and generate language-based responses but also learn to interact with and utilize external tools to enhance their capabilities (Nakano et al., 2022; Schick et al., 2023; Shen et al., 2023; Qian et al., 2023; Song et al., 2023b; Xu et al., 2023; Li et al., 2023; Hao et al., 2023; Zhang et al., 2023). By incorporating tools, LLMs can offer solutions in various areas, including visual-language processing (Gupta and Kembhavi, 2023; Wu et al., 2023), mathematical reasoning (Lu et al., 2023), and tasks in specialized domains (Jin et al., 2023; Tang et al., 2023b).

However, previous research on tool learning mainly focused on teaching LLMs to use tools, but ignores the importance of shortlisting relevant tools. In this paper, we focus on using LLMs to improve the tool retrieval process. In contrast to previous researches that heavily rely on finetuning retrievers (Schick et al., 2023; Patil et al., 2023) to shortlist tools, we propose a novel Edit-and-Ground paradigm, leveraging LLMs' parametric knowledge to learn and create more informative descriptions for tools. This approach seeks to provide richer information for the retriever, leading to more accurate retrieval.

3 Task and Data

We hereby formulate the task of tool retrieval and describe the dataset for this task.

3.1 Task Definition

The tool retrieval process involves taking a user query Q and an index base of tool descriptions $D = \{d(t_1), d(t_2), \ldots, d(t_n)\}$ as input, where each $d(t)$ represents the description of each tool t. The retriever then sifts through the tool descriptions in D and shortlists a relevant tool set $T = \{t_1, t_2, \ldots, t_k\}$ that are potentially suited to address aspects of the user query Q. It is essential to underline that unlike conventional retrieval tasks, the task of tool retrieval is goal-oriented in nature,

which means the set of retrieved tools T should be able to address the user query Q.

The systems are expected to accurately retrieve relevant tools and understand the user intents and complex synergy between tools, thus truly assisting users in problem-solving processes.

3.2 Dataset

Existing datasets for tool learning, such as those delineated in (Li et al., 2023; Patil et al., 2023; Tang et al., 2023a; Xu et al., 2023), provide insights into the field. Nonetheless, these datasets exhibit limitations, where they only cover a limited number of tools or solely support simple single-tool usage scenarios, where user queries are simple and could be addressed by a single tool.

Contrastingly, Qin et al. (2023) proposed ToolBench, a dataset covering more than 3,000 tools from 49 categories (such as advertising, data analysis, and transportation) and support complex, multi-tool user scenarios. In these scenarios, a single user query necessitates the sequential application of multiple tools, each contributing uniquely to the resolution of the query. The ToolBench dataset synergizes with the RapidAPI Hub, a prominent API marketplace that consolidates a vast array of real-world APIs. The multi-tool query creation process involves selecting representative tools within each category or collection, crafting queries to mimic real-world problem-solving scenarios.

Given our research focus and the nature of our study, we have chosen to concentrate on the *Intra-Category* setting of the ToolBench dataset. The intra-category setting provides high-quality user queries, where the hierarchies of tools are clearly defined based on their main functionalities. It motivates understanding complex interactions and synergies between tools that share a common functional domain. The setting mirrors real-world situations where problem-solving often demands a multifaceted and integrative use of diverse tools. The ToolBench dataset annotates paths of executed tools that successfully address the user queries as solution paths. The average length of the solution paths is 4. We take the annotated solution paths as the ground truth for our task.

4 Method

In this section, we describe the proposed framework to integrate tools with LLMs for addressing complex user queries. Our methodology is grounded in two innovative paradigms: the Plan-and-Retrieve (P&R; §4.2) and Edit-and-Ground (E&G; §4.3). We discuss the coordination between two paradigms in §4.4.

4.1 Method Overview

PLUTO integrates two key paradigms, Plan-and-Retrieve (P&R) and Edit-and-Ground (E&G), to effectively address complex user queries with LLMs.

The **Plan-and-Retrieve** paradigm is a two-stage process. The *Plan* stage decomposes user queries into focused sub-queries, while the *Retrieve* stage matches these sub-queries with relevant tools.

The **Edit-and-Ground** paradigm, consisting of the *Evaluator* and *Optimizer*, focuses on enhancing tool descriptions.

These paradigms are designed to work in tandem. P&R paradigm addresses immediate user queries, while E&G actively identifies and collects under-informative tool descriptions for optimization.

4.2 Plan-and-Retrieve

The Plan-and-Retrieve (P&R) paradigm is designed as a two-stage process to effectively address complex user queries.

Plan. In the *Plan* stage, a LLM-based planner autoregressively decomposes the user query Q into sub-queries $q_1, q_2, \ldots, q_n$. To ensure the robustness and quality of the decomposed sub-queries, we follow Zhu et al. (2023). Specifically, for each step of sub-query generation, the planner first generates a batch of hypotheses. Then, we cluster the generated hypotheses along with previously created sub-queries via K-means clustering algorithm. Finally, we select a sub-query from the hypotheses that distinguishes the most from the previous sub-queries to proceed[2].

As shown in Fig. 2, the planner autoregressively decomposes the user query Q into more fine-grained sub-queries based on assessments at inference time. After the generation of a sub-query q_t, the planner evaluates whether the original query Q has been satisfactorily achieved based on the current planning history. If the evaluation determines that the goal has been met, the iterative process concludes. Otherwise, the planner proceeds to generate the subsequent sub-query q_{t+1}. This active and autoregressive planning at inference time facilitates a more focused understanding of the tools. We use the following prompt template for the planner.

[2]Please refer to Appx. §A for algorithm implementation.

Planner Prompt

You are given a user query requiring multi-step actions and reasoning. Output an immediate next action to achieve the user query.
Previous Actions: {*Planning History*}
User Query: {Q}
Output:

Retrieve. In the *Retrieve* stage, for each sub-query q_i, the retriever shortlists the most suitable tools $T_i \in D$. We first retrieve a pool of candidate tools that matches q_i, represented as

$$T_i' = Ret(q_i), \tag{1}$$

where Ret represents the retriever.

To enhance the robustness of retrieval, we re-rank the candidate tool set T_i' by LM-likelihood score between the sub-query q_i and each tool $t_j \in T_i$, which is calculated as follows:

$$\text{LM-likelihood}(q_i, t_j) = -\log P(q_i, d(t_j)). \tag{2}$$

Based on the re-ranked tools, we choose the top-5 tools $T'_{i,top-5}$ and feed them into a LLM-based predictor, which outputs a shortlisted tool set T_i from the candidate tool set $T'_{i,top-5}$ that are relevant to q_i. We use this prompt for the predictor.

Predictor Prompt

You are given candidate tools that can be potentially used to solve current goal. Among candidate tools, select a list of relevant tools which would help achieve the current goal.
Current Goal: {q}
Candidate Tools: {T'}
Output:

As a result, the final shortlisted tool set T is formed by

$$T = \bigcup_{i=1}^{n} T_i, \forall i \in [1, n] \cap \mathbb{Z}. \tag{3}$$

For the choice of Ret, we adopt a neural (dense) retriever method. For each sub-query q_i, the dense vector representation $\mathbf{q}_i$ is obtained by passing q_i through a dense encoder. Similarly, we obtain dense representation $\mathbf{d}$ through a dense encoder for each tool description d. The tool index corpus D is formed as a collection of $\mathbf{d}$.

The P&R module interleaves *Plan* and *Retrieve* until the planner evaluates that the user query has been sufficiently decomposed and addressed through the retrieved tools. The module then returns T as the relevant tools to address the user query.

Algorithm 1 Edit-and-Ground Algorithm

Input: Trainset, Devset, Toolset, Failure_Threshold, Max_Rounds
Output: Optimized Tool Descriptions
Initialize cache for tools in *Toolset*
cur_round = 0

while cur_round < Max_Rounds **do**
  ## Phase 1: Evaluate Retrieval Performance
  for each (*query*, *gt_tools*) in *Trainset* **do**
    predicted_tools ← P&R(*query*)
    for each *tool* in *gt_tools* **do**
      tool.trials += 1
      if *tool* not in *predicted_tools* **then**
        tool.failure += 1
        tool.queries.add(query) ▷ Failure queries
      end if
    end for
  end for

  ## Phase 2: Failed Tool Description Optimization
  for each *tool* in *Toolset* **do**
    if $\frac{tool.failure}{tool.trials}$ > *Failure_Threshold* **then**
      U ← Remove specific entities from *tool*.queries
      R ← Predict *reasons* for failure of U
      d(tool) ← *tool.description*
      d'(tool) ← E&G(*tool*, *d(tool)*, U, R)

      ## Phase 3: Evaluate Performance of d'(tool)
      cur_recall ← Eval(*Devset*, *d'(tool)*)
      if *tool.recall* < *cur_recall* **then**
        tool.description ← *d'(tool)*
        tool.recall ← *cur_recall*
      end if
    end if
  end for
  cur_round += 1
end while

4.3 Edit-and-Ground

The Edit-and-Ground (E&G) paradigm focuses on refining under-informative tool descriptions to align them with user queries. As shown in Alg. 1, the evaluator examines the quality of tool descriptions by retrieval results. A tool description is viewed as under-informative if the number of failure cases of retrieval exceeds a pre-defined threshold. We collect such tools for later optimization.

Entity Filtering Prompt

Given an input query, remove specific entity information and return only the general query template. Make sure your method is generalizable to handle other queries.
{*Demonstrations*}
User Queries: {Q}
Output:

Subsequently, the optimizer takes a tool t with its base description $d(t)$ and U, a batch of relevant user queries, as input. To avoid the optimizer overfitting to a local batch, we use an LLM to filter out specific entities for each query in U. The entity

filtering prompt template is shown as above.

To assist the optimizer in improving underperformed tool descriptions, we prompt LLM to generate reasons R explaining why the tool could be related and helpful in addressing user queries. The functionality assessment prompt template is shown below:

Functionality Assessment Prompt

Given a batch of user queries and a tool, predict a reason why the tool would be helpful to address the user queries.
{*Demonstrations*}
User Queries: {U}
Tool: {*t: d(t)*}
Output:

Finally, by prompting LLM with 1) base tool description $d(t)$, 2) entity-filtered user queries U, and 3) the reasons R, we obtain an enriched tool description $d'(t)$. Please refer to Fig. 4 in Appendix C for the prompt template. We formally represent this process as

$$d'(t) = E\&G(t, d(t), U, R). \tag{4}$$

The optimization process is executed in multiple rounds as described in Alg. 1. In each round, we evaluate the retrieval recall on the development set for each tool and compare it with the previous round. If the current round's recall is better than the previous one, we update the tool's description; otherwise, we keep the original description.

The Edit-and-Ground involves using the LLM's extensive world knowledge, combined with the contextual details provided by U, to edit and enhance $d(t)$. The result of this task is an enriched tool description $d'(t)$, expected to resonate more closely with real-world user scenarios and increase the utility of the tool in practical applications.

4.4 Paradigm Coordination and Inference

Our PLUTo framework employs strategic coordination of the Plan-and-Retrieve (P&R) and Edit-and-Ground (E&G) paradigms, phased to optimize the process of tool retrieval. This section elucidates the interaction between these paradigms during the optimization phase and the subsequent inference phase.

Optimization Phase. During the optimization phase, P&R and E&G operate alternatively. P&R is tasked with decomposing a user query Q into manageable sub-queries $q_1, q_2, \ldots, q_n$. These sub-queries facilitate a more focused retrieval of tools from the tool set D, ensuring that the process is aligned with specific aspects of the query.

During planning, the E&G paradigm is actively engaged in optimizing the descriptions of the tools within D. This optimization, leveraging the LLM's extensive knowledge base, is particularly targeted at tools that exhibit underperformance in retrieval effectiveness. By enriching these tool descriptions, E&G significantly enhances the overall retrieval process, making the toolset more responsive and aligned with the practical demands of diverse queries.

Inference Phase. At the time of inference, the P&R paradigm remains active, utilizing the previously enriched and optimized tool descriptions. In this phase, the E&G paradigm ceases its operation and does not engage in any further optimization of tool descriptions. The refined tool descriptions, already enhanced by E&G, now serve as a comprehensive resource for the retriever to draw upon in response to the decomposed sub-queries.

5 Experiments

In this section, we evaluate the proposed PLUTo framework for tool retrieval and compare it with baseline methods. We will delve into the details of our experimental setup (§5.1), discuss the results (§5.2) obtained, and perform an ablation study to understand strengths of different components (§5.3). By executing the retrieved tools, we evaluate their correctness in addressing user queries to further validate our findings (§5.4). We present case studies to qualitatively evaluate the strength of PLUTo framework (§5.5).

5.1 Experiment Setup

Evaluation Protocol. We evaluate using three metrics to assess the effectiveness of our tool retrieval system. *Recall (Rec)* measures the proportion of relevant tools that are successfully retrieved by our system. High indicates that the system is effective in identifying a comprehensive set of relevant tools for a given query and is more likely to yield a solution to address the user query. We also report the *Normalized Discounted Cumulative Gain (NDCG)* that evaluates the relevance and quality of ranked search results. In addition, we report *pass rate*, an automatic evaluation metric of ToolBench (Qin et al., 2023). The pass rate measures a system's ability to successfully address the user query with

Model	Retriever	Non-Finetuned		Finetuned	
		Rec	NDCG	Rec	NDCG
BM25	–	18.82	37.44	–	–
ToolRetriever	DPR†	19.58	50.98	27.80	71.21
	Contriever	31.78	74.70	42.77	79.16
PLUTo	DPR	36.65	75.10	43.27	79.93
	Contriever	**46.57**	**82.93**	**48.47**	**84.73**

Table 1: This table compares various tool retrieval models using Recall and NDCG metrics in both Non-Finetuned and Finetuned settings. It includes an ablation study on the impact of using different retrievers, demonstrating the generalizability of PLUTo. † indicates the previous SOTA implementation, as specified in (Qin et al., 2023).

a retrieved subset of tools in limited budgets by interacting with real-world RESTful APIs (§5.4).

To test the generalizability of our approach, we benchmark the tool retrieval performance under a Non-Finetuned setting, where we directly apply an off-the-shelf retriever model to comprehensively showcase PLUTo's adaptivity. To test the model's practical applicability, we also benchmark retrieval performance under Finetuned setting, where we finetune the retriever model on domain-specific knowledge. We evaluate 500 user queries for each setting.

Baselines. We compare our system against several representative retrieval methods. These include: (1) *BM25:* a widely-used probabilistic retrieval framework, calculating the relevance of documents to a query based on the frequency of query terms in each document; (2) *ToolRetriever*: a neural retrieval approach that achieves the current state-of-the-art (SOTA) performance on ToolBench retrieval task (Qin et al., 2023). To understand the flexibility of our framework, we benchmark PLUTo's performance when incorporated with different retrievers. Specifically, we use DPR (Karpukhin et al., 2020) and Contriever (Izacard et al., 2022).

Implementation Details. For the implementation of PLUTo, we use DSPy framework (Khattab et al., 2023) to facilitate efficient interaction between retriever and LLM. We choose ChatGPT[3] as our main LLM for both P&R and E&G. The maximum round for the E&G module is set to 5. For ToolRetriever, we retrieve top-5 tools using the respective retrievers. The data is divided into 70-15-15 splits for training, development, and testing, respectively. For our experiment, we randomly select 500 data samples from the test split for each setting mentioned in Evaluation Protocol section.

For the Finetuned settings, we finetune the neural dense retriever model by including negative samples during in-batch training (Karpukhin et al., 2020). For each positive pair of query q_j and its relevant tool d_j^+, we include n negative tools as negative samples. We use a cross-entropy loss with softmax function over the batch B:

$$L = -\frac{1}{B}\sum_{j=1}^{B}\log\left(\frac{e^{\mathbf{q}_j\cdot\mathbf{d}_j^+}}{e^{\mathbf{q}_j\cdot\mathbf{d}_j^+}+\sum_{i=1}^{n}e^{\mathbf{q}_j\cdot\mathbf{d}_{ij}^-}}\right) \quad (5)$$

5.2 Results

The experimental results, detailed in Tab. 1, underscore the significant advantages of our proposed PLUTo models. In the Non-Finetuned setting, PLUTo with Contriever showcases remarkable scores, achieving 46.57% in Recall, outperforming the best baseline by 9.92 points. This result shows the model's robust ability to identify relevant tools without the necessity for specific finetuning, a critical advantage in dynamic tool retrieval environments. We observe a consistent trend in the Finetuned setting, with the model scoring 48.47% in Recall, demonstrating a 5.7 points lead when compared with the Contriever baseline. This indicates that our model is highly effective on retrieving relevant tools.

Furthermore, our model outperforms baselines across all settings on NDCG scores. In the Non-Finetuned setting, our model leads by 8.23 points. In the Finetuned setting, our model beats the baseline by 4.57 points. These results reflect PLUTo not only the relevance of the tools retrieved but also their ranking in order of utility and applicability to the user's query, which is a indication to the model's nuanced understanding of tool utility.

[3]OpenAI. (2023). ChatGPT (November 21st version).

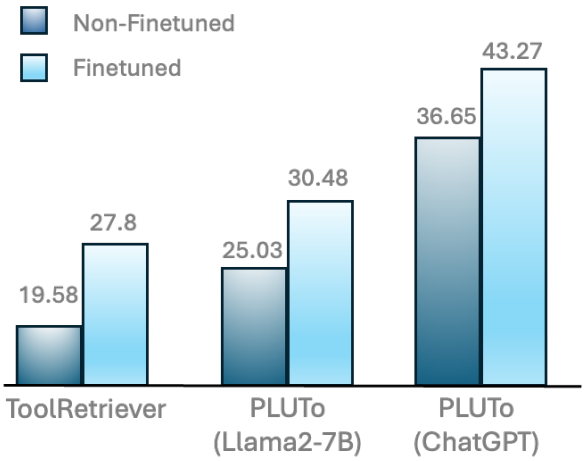

Figure 3: Performance comparison among different LLMs for Plan-and-Retrieve paradigm using Recall score. The backbone retriever is DPR.

To show the generalizability of PLUTO, we select different retrievers for the Plan-and-Retrieve (P&R) paradigm. We observe that PLUTO has synergy with both DPR and Contriever models, regardless of their different architecture, that achieves higher Recall and NDCG scores than the baselines. This indicates that PLUTO is a plug-n-play and retriever-agnostic framework that features effectiveness and flexibility under different circumstances.

The experimental results highlight the superior performance of PLUTO framework. Together, the P&R and E&G paradigms establish a dynamic and effective framework, which not only accurately interprets and responds to user queries but also maintains an evolving understanding of tool functionality. This duality ensures that PLUTO remains highly effective and adaptable in various setups, consistently aligning user needs with the most suitable tools and their capabilities.

5.3 Ablation Study

As shown in Fig. 3, we observe that both the Llama2 (Touvron et al., 2023) and ChatGPT variants show considerable improvements in tool retrieval capabilities, with notable increases in Recall and NDCG scores compared to baseline models. This consistent improvement across different LLM integrations conclusively demonstrates the robustness and effectiveness of our method. This finding is particularly important as it suggests that our approach is not overly reliant on any single LLM, thereby showcasing the broad applicability and potential of our methods in diverse settings.

As shown in Tab. 2, the ablation experiment on

Model	Non-Finetuned		Finetuned	
	Rec	NDCG	Rec	NDCG
PLUTO- full	**46.57**	**82.93**	**48.47**	**84.73**
- w/o E&G	42.55	80.70	44.90	81.10
- w/o P&R	38.12	77.60	47.07	81.90

Table 2: Ablation Study.

the PLUTO- full, focusing on the removal of Edit-and-Ground (E&G) and Plan-and-Retrieve (P&R) components, provides intriguing insights into their roles in tool retrieval tasks. Generally, removing E&G leads to decreased Recall and NDCG scores across settings, underscoring its critical role in enhancing what the model seeks to retrieve. On the other hand, excluding P&R tends to diminish more of the model's performance in Non-Finetuned settings, particularly impacting Recall. This highlights P&R's importance in effectively retrieving relevant information. A comparative analysis reveals that the full implementation of PLUTO-ChatGPT, incorporating both E&G and P&R, consistently delivers strong performance across all metrics and settings, emphasizing the synergistic strength of these components. The variants of the model, lacking either E&G or P&R, provide valuable insights into the unique contributions of each component to the model's overall efficacy.

5.4 Execution Pass Rate

We evaluate the pass rate of the execution schema generated by ChatGPT using the DFSDT approach (Qin et al., 2023). Using the ToolEval package, we assessed two distinct retrieval tools, ToolRetriever and PLUTO, for their correctness and efficiency in responding to user queries. The PLUTO achieves **72.3%** for pass rate, while the previous SOTA system ToolRetriever scored **69.3%**.

This experiment's findings emphasize the pivotal role of advanced retrieval strategies in enhancing user query response quality. The improvement gained during the retrieval phase, such as higher accuracy and relevance in responses, significantly contribute to the downstream tasks.

5.5 Case Study

As shown in Tab. 3, we compare our PLUTO against the ToolRetriever baseline to underscore PLUTO's proficiency in retrieving relevant tools for diverse user queries. Through selected exam-

Question	Gold Answer	PLUTo Answer	ToolRetriever Answer
I'm planning a weekend getaway with my partner and I want to surprise them with a romantic playlist. Could you fetch the reels and posts from romantic music artists on Instagram? Additionally, could you search for books about love and relationships on Open Library?	Instagram Reels and post Downloader, Open Library	Instagram Reels and post Downloader, Instagram, Open Library, Instagram Downloader	Love Quotes by LoveMelon, The Love Calculator, Book Finder, fb-video-reels, Reading Home APIs
I'm planning a family movie night and I need a movie recommendation. Can you fetch the trending images for movie posters and provide me with the details of the most popular movie from the past month? Also, check the status of the movie session and download the completed movie.	Magisto, Bing Image Search	Magisto, gogoanime-data-api, Youtube video info, Advanced Movie Search, Image Service, Memes, Bing Image Search, Netflix Data	TikTok Info, Tiktok Video Feature Summary, TikTok Full Video Info, TikTok Downloader - Download Videos without watermark
I'm a music blogger and I'm searching for interesting radio stations to feature on my website. Can you help me find radio stations that play a mix of genres? Also, provide me with the details of the master for the track with the ID '987654' in the LANDR Mastering.	LANDR Mastering v1, 50K Radio Stations	GMC Radio, LANDR Mastering v1, 50K Radio Stations, 60K Radio Stations	LANDR Mastering v1, Spotify_v2, TuneIn, Spotify Scraper, Spotify_v3

Table 3: Performance comparison of PLUTo and ToolRetriever in retrieving relevant tools for user queries. This table demonstrates the effectiveness of PLUTo in closely aligning with the gold standard answers for diverse queries, showcasing its superior ability to understand and fulfill user needs compared to ToolRetriever. The highlighted tools are the correctly retrieved ones.

ples, PLUTo's superior understanding and comprehensive response capabilities are highlighted, especially in scenarios requiring nuanced tool selection.

For instance, for organizing a romantic weekend in the first example, PLUTo not only identifies all essential tools but also enhances the search with additional relevant resources, showcasing its broad and accurate grasp of user needs. This is contrasted with ToolRetriever, where the retrieved tools are only similar on a surface level (the majority of the tools contain the term "Love") and fail to understand the user's intent. This emphasizes PLUTo's improved relevance and precision in tool retrieval. We also showcase the descriptions of tools before and after optimization by the Edit-and-Ground paradigm in Tab. 4.

By leveraging the Plan-and-Retrieve (P&R) and Edit-and-Ground (E&G) components, PLUTo marks a significant advancement over conventional retrieval systems, demonstrating its adaptability and utility in fulfilling diverse user requirements.

6 Conclusion

We introduced PLUTo, a framework composed of the Plan-and-Retrieve and Edit-and-Ground paradigms, which marks a distinctive departure from traditional methodologies, setting a new standard for tool retrieval. The empirical results illustrate the superiority of PLUTo across critical retrieval performance metrics as well as pass rate in real-world tool-use evaluation. These metrics collectively attest to the model's efficacy in identifying relevant tools and successfully addressing complex user queries. We hope the adaptability and efficiency of PLUTo can empower a multitude of domains where accurate and timely retrieval of tools is paramount. From autonomous scientific discovery to software development, the potential applications are as diverse as they are impactful.

Acknowledgement

We appreciate the reviewers for their insightful comments and suggestions. Tenghao Huang and Muhao Chen were supported by an Amazon Research Award, a Keston Exploratory Research Award, and the NSF Grant ITE 2333736.

Limitation

Our study, while enhancing tool learning by planning and editing strategies, is notably constrained by its reliance on English language datasets. This focus on English limits the model's applicability to other languages with distinct syntax and semantics and confines its evaluation to specific English data

sources, leaving its performance on diverse language setups unexplored. Future research should address this limitation by developing multilingual capabilities and conducting evaluations across varied data sources.

The Edit-and-Ground (E&G) may be executed to further optimize the descriptions. However, due to the cost, we currently set a relatively loose stop criterion that is enough to demonstrate the effectiveness of the presented method.

Ethical Consideration

In conducting this research, we have adhered to ethical guidelines and legal norms to ensure responsible data usage. The data used in this study was obtained from public datasets, specifically ToolBench. We ensured not to violate any terms of service of the data sources.

References

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, pages 1870–1879. Association for Computational Linguistics (ACL).

Tanmay Gupta and Aniruddha Kembhavi. 2023. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14953–14962.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings.

Cheng-Yu Hsieh, Si-An Chen, Chun-Liang Li, Yasuhisa Fujii, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. 2023. Tool documentation enables zero-shot tool-usage with large language models. *arXiv preprint arXiv:2308.00675*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.

Qiao Jin, Yifan Yang, Qingyu Chen, and Zhiyong Lu. 2023. Genegpt: Augmenting large language models with domain tools for improved access to biomedical information.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines.

Mandar Kulkarni, Kyung Kim, Nikesh Garera, and Anusua Trivedi. 2023. Label efficient semi-supervised conversational intent classification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 96–102, Toronto, Canada. Association for Computational Linguistics.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2022. Dynamic prompt learning

via policy gradient for semi-structured mathematical reasoning. In *The Eleventh International Conference on Learning Representations*.

Daniel Lübke, Olaf Zimmermann, Cesare Pautasso, Uwe Zdun, and Mirko Stocker. 2019. Interface evolution patterns: balancing compatibility and extensibility across service life cycles. pages 1–24.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Cheng Qian, Chi Han, Yi R. Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. 2023. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face.

Raphael Shu, Elman Mansimov, Tamer Alkhouli, Nikolaos Pappas, Salvatore Romeo, Arshit Gupta, Saab Mansour, Yi Zhang, and Dan Roth. 2022. Dialog2api: Task-oriented dialogue with api description and example programs. *arXiv preprint arXiv:2212.09946*.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023a. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, Ye Tian, and Sujian Li. 2023b. Restgpt: Connecting large language models with real-world restful apis.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023a. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases.

Xiangru Tang, Anni Zou, Zhuosheng Zhang, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. 2023b. Medagents: Large language models as collaborators for zero-shot medical reasoning. *arXiv preprint arXiv:2311.10537*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. 2023. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*.

Prateek Yadav, Qing Sun, Hantian Ding, Xiaopeng Li, Dejiao Zhang, Ming Tan, Parminder Bhatia, Xiaofei Ma, Ramesh Nallapati, Murali Krishna Ramanathan, et al. 2023. Exploring continual learning for code generation models. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Zonglin Yang, Xinya Du, Junxian Li, Jie Zheng, Soujanya Poria, and Erik Cambria. 2023. Large language models for automated open-domain scientific hypotheses discovery. *arXiv preprint arXiv:2309.02726*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2023. Making retrieval-augmented language models robust to irrelevant context.

Kexun Zhang, Hongqiao Chen, Lei Li, and William Wang. 2023. Syntax error-free and generalizable tool use for llms via finite-state decoding.

Yin Zhu, Zhiling Luo, and Gong Cheng. 2023. Furthest reasoning with plan assessment: Stable reasoning path with retrieval-augmented large language models. *arXiv preprint arXiv:2309.12767*.

A K-means Algorithm for Furthest Planning

Here, we present the algorithm for selecting the optimal sub-query to proceed with the Plan-and-Retrieve paradigm.

Algorithm 2 Sub-query Selection

Let Q_{prev} be the set of previous queries, and Q_{cand} the set of candidate queries.
if $Q_{\text{prev}} = \emptyset$ **then**
  return Q_{cand}
end if
$Q_{\text{total}} = Q_{\text{cand}} \cup Q_{\text{prev}}$
$V = \text{TFIDFVectorizer}(Q_{\text{total}})$
$C = \text{KMeans.fit}(V)$
Let L_{prev} be the cluster labels for Q_{prev} in C.
$Q_{\text{filtered}} = \{q \mid q \in Q_{\text{cand}}, \text{label}(q, C) \notin L_{\text{prev}}\}$
if $Q_{\text{filtered}} = \emptyset$ **then**
  return Q_{cand}
end if
return $\text{random.choice}(Q_{\text{filtered}})$

B Evaluation Framework for NDCG Assessment

In the process of evaluating the correspondence between the retrieved digital tools and the user's query, a nuanced approach is employed to assign relevance scores. This scoring paradigm operates on a scale from 0 to 2. A score of '2' is allocated exclusively to those tools that exhibit either an exact match or a functional equivalence to the predefined standards, referred to as 'ground-truths.' A score of '1' is designated for tools that are deemed to be of moderate relevance. Conversely, a score of '0' is reserved for tools that are determined to be irrelevant to the user's query. We hire graduate students to carry out this task.

C Edit-Ground Prompt Template

Please refer to Fig. 4. The task explanation and demonstration are shown in orange. The input is shown in blue.

D Case Studies for Edit-and-Ground paradigm

From Tab. 4, we observe that humans tend to craft tool descriptions that are abstract and simple, which can omit vital details necessary for both users and automated systems. Abstract descriptions can create a gap in understanding, particularly in domains where the nuances of functionality are key to effective use.

Contrastingly, tool descriptions that are contextualized and grounded by PLUTo offer a more detailed narrative, linking the tool's features with specific use cases and demonstrating its utility in practical scenarios. This detailed approach not only aids human users in understanding how a tool fits their specific needs but also enhances the precision of automated retrieval systems in matching tools to user queries.

Tool Name	Before	After
Magisto	Magically transform your videos. Magisto turns your everyday videos into exciting, memorable movies you'll want to watch again and again.	The Magisto API is a powerful tool that can transform user videos into exciting and memorable movies. It offers a range of capabilities to enhance the user's movie experience, provide movie recommendations, and optimize video content for discovery on platforms like Vimeo. The API allows users to choose visually appealing movies ...
Bing Image Search	An AI service from Microsoft Azure that turns any app into an image search resource.	The Bing Image Search API is a comprehensive and powerful tool that enables users to search for and retrieve a wide range of images based on their specific requirements. With its extensive database and advanced search capabilities, this API offers a variety of features to cater to different use cases and scenarios. The API provides the ability to fetch trending images for movie posters. Additionally, the API offers the functionality to retrieve details of the most popular movie from the past month, enabling users to make informed decisions about their movie selection.
LANDR Mastering v1	Give your users an instant and customized audio mastering solution by harnessing the power of LANDR's industry-leading, AI-driven mastering engine and its patented machine learning capabilities.	The "LANDR Mastering v1" API is a comprehensive and powerful audio mastering solution designed to cater to the needs of music producers, music lovers, and anyone interested in high-quality audio production. With its instant and customizable features, the API offers a range of capabilities to enhance the music production process. For music producers, the API allows them to download a specific track by providing the track ID. Additionally, the API can fetch track credits, enabling the user to gather essential information for their music production needs ...

Table 4: Comparative Analysis of Tool Descriptions before and after optimization by the Edit-and-Ground paradigm. This table showcases the evolution of tool descriptions for selected tools, illustrating the substantial enhancement in clarity, detail, and functionality offered to users.

Task: Currently, the description of the tool is not clear. Improve the description of a tool based on its relevance score to the usecases.

Input:
source tool description: This is the original description of the tool as provided.
usecases: These are the scenarios where the tool would be useful.
reasoning: This is an explanation for why the tool is relevant and useful to address the usecases.

Demonstration:

Input:
source_tool_description: {t}
usecases: {U}
reasoning: {R}

Output:
improved_tool_description: {t'}

Input:
source_tool_description: {t}
usecases: {U}
reasoning: {R}

Output:
improved_description:

Figure 4: Edit-and-Ground template.