Re-Invoke: Tool Invocation Rewriting for Zero-Shot Tool Retrieval

Yanfei Chen[1]*, **Jinsung Yoon**[1], **Devendra Singh Sachan**[2],
Qingze Wang[3], **Vincent Cohen-Addad**[3], **Mohammadhossein Bateni**[3],
Chen-Yu Lee[1], **Tomas Pfister**[1]
[1]Google Cloud AI Research, [2]Google DeepMind, [3]Google

Abstract

Recent advances in large language models (LLMs) have enabled autonomous agents with complex reasoning and task-fulfillment capabilities using a wide range of tools. However, effectively identifying the most relevant tools for a given task becomes a key bottleneck as the toolset size grows, hindering reliable tool utilization. To address this, we introduce Re-Invoke, an unsupervised tool retrieval method designed to scale effectively to large toolsets without training. Specifically, we first generate a diverse set of synthetic queries that comprehensively cover different aspects of the query space associated with each tool document during the tool indexing phase. Second, we leverage LLM's query understanding capabilities to extract key tool-related context and underlying intents from user queries during the inference phase. Finally, we employ a novel multi-view similarity ranking strategy based on intents to pinpoint the most relevant tools for each query. Our evaluation demonstrates that Re-Invoke significantly outperforms state-of-the-art alternatives in both single-tool and multi-tool scenarios, all within a fully unsupervised setting. Notably, on the ToolE datasets, we achieve a 20% relative improvement in nDCG@5 for single-tool retrieval and a 39% improvement for multi-tool retrieval.

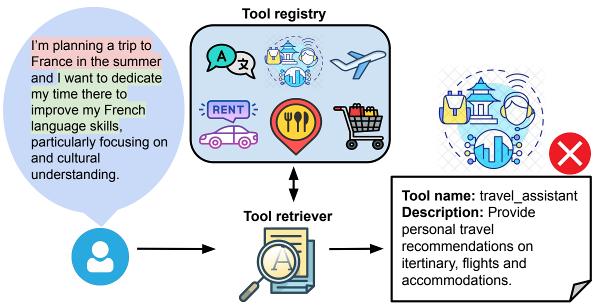

Figure 1: An example of low-performance retrieval methods failing to identify the actual user intents "improve French language skills" due to the context "planning a trip to France". It selects similar, but incorrect travel assistant tool instead of the ground-truth language learning tool from the given pool of tools.

1 Introduction

Recently, large language models (LLMs) have demonstrated impressive capabilities on a variety of complex tasks, including math, reasoning and coding (OpenAI, 2023c; Anil et al., 2023; Google, 2023b). They can even surpass average human performance on standardized exams such as college entrance tests, law school admission, and math competitions (Zhong et al., 2024). However, LLMs are pre-trained on a static corpus, limiting their adaptability to the rapidly evolving real world, and frequent fine-tuning (Wei et al., 2021) is computationally expensive.

In contrast, humans leverage a vast array of tools to interact with the external world, using search engines for information retrieval, maps for navigation, calculators for algebraic tasks, and so on. Augmenting LLMs with external tools, rather than relying solely on their internal knowledge, could unlock their potential to tackle even more challenging problems. This insight has driven recent interests in both academic research (Parisi et al., 2022; Schick et al., 2023; Lu et al., 2023; Cai et al., 2023; Patil et al., 2023; Hsieh et al., 2023; Qin et al., 2023) and industrial applications. Examples include ChatGPT plugins (OpenAI, 2023a), with supported third-party APIs, and Bard extensions (Google, 2023a) connecting to Google APIs and services.

Common approaches to integrate tools with LLMs often rely on supervised methods to generate tool calling functions (Schick et al., 2023; Patil et al., 2023; Parisi et al., 2022; Qin et al., 2023; Hao et al., 2023) or in-context learning by providing the tool documents and few-shot demonstrations (Xu

* Correspondence to: {yanfeichen, jinsungyoon, chenyulee}@google.com

Findings of the Association for Computational Linguistics: EACL 2024, pages 4705–4726
November 12-16, 2024

et al., 2023; Lu et al., 2023; Hsieh et al., 2023). However, these methods face practical challenges when scaling to a large number of tools on complex tasks: (a) *Input Token Length Limitations*: LLMs have inherent input token length limitations, making it infeasible to include a comprehensive list of tools within a single prompt. Moreover, LLMs can struggle to effectively process relevant information from lengthy input contexts (Liu et al., 2024). (b) *Evolving Tool Pool*: LLMs are often paired with a tool retriever trained on labeled query-tool pairs. However, the ideal LLM toolkit should be vast and dynamic, with tools undergoing frequent updates. Providing and maintaining labels for such an extensive and evolving toolset is impractical. Continuous retraining would also require extensive production maintenance. (c) *Ambiguous User Intents*: User contexts in the queries could obfuscate the underlying intents and failure to identify the intents could lead to calling the wrong tools (See Fig. 1). Retrieving the relevant tools to address all the intents described in the user query remains a challenging task.

To address these unique challenges, we introduce **Re-Invoke**, a novel unsupervised retrieval method to enable effective retrievals even when user intent is multifaceted or tool document is lacking. To the best of our knowledge, Re-Invoke is the first fully unsupervised approach to tackle multi-tool retrieval use cases. It leverages LLMs for both tool document enrichment and user intent extraction, thereby enhancing tool retrieval performance across various use cases. Our approach consistently and significantly improves upon state-of-the-art alternatives, achieving a 20% relative improvement in nDCG@5 on single-tool retrieval tasks and 39% improvement on multi-tool retrieval tasks with ToolE dataset.

2 Related Work

Tool Retrievals for Tool-Use. ReAct (Yao et al., 2023) pioneers the interaction and reasoning with diverse tools using in-context reasoning traces, particularly in decision-making and multi-step reasoning environments. Schick et al. (2023) proposes a self-supervised training method with API demonstrations. Patil et al. (2023) and Hsieh et al. (2023) demonstrate that augmenting LLMs with tool document significantly improves their ability to generate correct API calls by mitigating hallucinations, compared to prompting with demonstrations alone. Yuan et al. (2024) also shows unifying tool instruction leads to better tool usage. However, tool document retrieval for LLM tool learning is currently under-explored, as most work simply uses LLM agents to retrieve a limited number of tools. Patil et al. (2023) first demonstrate that LLMs generate more reliable outputs with the integration of a retrieval system using BM25 (Robertson et al., 2009) and GPT-index (Liu, 2022), but still introduce more hallucination and errors compared to the ground truth retriever. Some works (Qin et al., 2023; Kong et al., 2023; Gao et al., 2024) train a Sentence-Bert transformer model using the fully labeled query-API document pairs as a tool retriever. The key distinction between our approach and existing tool retrieval systems lies in our emphasis on zero-shot usage, eliminating the need for any labeled data.

Generative Document Expansion. Appending relevant terms, such as queries, to documents effectively enriches document representation for sparse retrievals. Nogueira et al. (2019) demonstrated this by using a language model to generate search queries for improved retrieval in search engines. Lewis et al. (2021) introduced Probably Asked Questions (PAQ) by generating the question given a passage and an answer, and a retriever trained using PAQ demonstrated the strength in accuracy, speed, and space efficiency for selective QA. Ma et al. (2023a) trains a dense retriever after applying document expansion. Our approach also leverages document expansion with generative language models, but with a focus on tool selection rather than search engine or question-answering tasks. We further emphasize the ability to extract the user intents from queries to better match the varying complexities of downstream tasks.

Generative Query Expansion. Augmenting user queries with hypothetical information is a popular approach in both dense and sparse retrieval methods. Query2doc (Wang et al., 2023) expands queries with pseudo-generated documents through few-shot prompting. Jagerman et al. (2023) further extends this idea by studying different prompting methods. Liu et al. (2022) improves query expansion by balancing diversity and relevance through a combination of effective filtering and documents fusion. Shen et al. (2024) augments queries with potential answers by prompting LLMs with a composition of the query and its in-domain candidates. Mackie et al. (2023) enriches the original query with useful terms from diverse generation

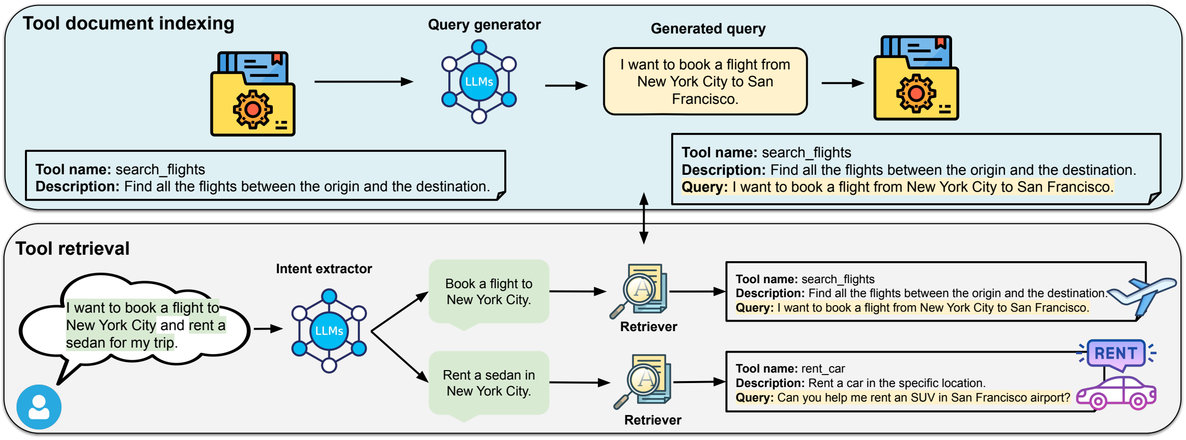

Figure 2: An overview of Re-Invoke for tool retrieval tasks. (**Top**) A query generator generates diverse synthetic queries from the tool documents and each synthetic query is concatenated with the tool document to create multiple copies of the expanded tool document. (**Bottom**) An intent extractor synthesizes multiple underlying intents from the user queries in order to retrieve the relevant tools.

subtasks. Chuang et al. (2023) proposes a query expansion and reranking approach to train a reranker after query expansion. Alternatively, Gao et al. (2023) proposes a zero-shot dense retrieval system by first instructing LLMs to generate a hypothetical document given the query for semantic retrievals. Those approaches primarily focus on generative pseudo-relevance feedback by enriching user queries within the retrieval system. They are fundamentally different from our approach, which focuses on query understanding rather than query expansion.

Query Rewriting. LLM-aided query rewriting is commonly used in conversational search engine to precisely understand user's contextual search intent through in-context learning (Yu et al., 2020; Ye et al., 2023; Mao et al., 2023; Anand et al., 2023). Some works even train the query rewriter in a rewrite-retrieve-read pipeline, allowing interaction with the search engine (Feng et al., 2023; Ma et al., 2023b). While LLMs are primarily used to summarize user context in conversations in these works, Re-Invoke focuses on extracting underlying intents for tool uses, rather than solely for information retrieval.

3 Method: Re-Invoke

We formulate the tool retrieval task as retrieving the most relevant tools that a downstream agent can execute to fulfill user queries, given a list of tool documents describing the intended tool usage.

Re-Invoke, our proposed fully unsupervised retrieval method designed for tool retrieval tasks, is illustrated in Fig. 2. It consists of two core components: (1) *Query generator*: for each tool document, LLMs generate diverse synthetic queries answerable by the corresponding tool. These queries enrich the tool document and are then indexed by encoding them into the embedding space when the tool documents are ingested offline. (2) *Query intent extractor*: during online inference, LLMs extract the core tool-related intent(s) from user queries, filtering out irrelevant background context. Each user intent is then encoded into the same embedding space as the tool documents for similarity matching. Pseudo-code of Re-Invoke is described in Algorithm 1.

3.1 Query Generator

Tool documents, provided by developers to explain tool usage, are often vague or incomplete, which can lead to incorrect tool retrievals. Additionally, the existing text embedding model designed for information retrieval tasks may not accurately model the semantic relationship between tool usage and user queries (Patil et al., 2023; Qin et al., 2023). In practice, tool developers often include usage examples in the tool documents to help users better understanding how to use the tools.

Following this intuition, we instruct LLMs to predict user queries by reading the provided tool document. The generated queries then serve as examples of intended tool usages. We encourage LLMs to produce creative and complex queries that the tool can address. This process is compatible with any LLMs, including enterprise models such

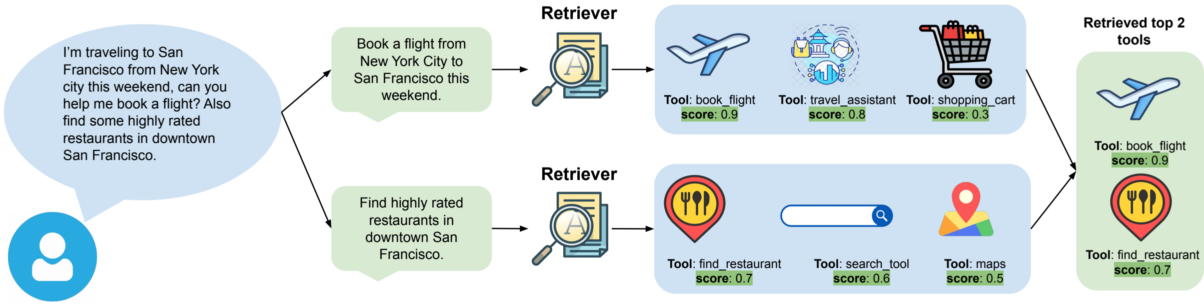

Figure 3: An illustration of the multi-view similarity ranking algorithm during retrieval. Multiple intents can be extracted from the user query and we first compute the similarity scores between the expanded tool documents and each intent in the embedding space. We rank and retrieve the top tools from each intent as the final retrieved tools.

Algorithm 1: Pseudo-code of Re-Invoke

Data: Query Q, List of tool documents D, The number of tools to retrieve k, Large language model $\mathcal{L}$, Text embedding model f_{enc}, m number of synthetic queries generated per each tool document

Result: k retrieved tool documents $\hat{D}_{1,\ldots,k}$

Function $\texttt{ReInvoke}(q, D, k, \mathcal{L}, f_{enc})$:
- **for** $d \in D$ **do**
  - **for** $i = 1, \ldots, m$ **do**
    - $d_i \leftarrow \texttt{Concat}(d, \mathcal{L}(d))$;
  - **end**
- **end**
- $q_1, \ldots, q_n \leftarrow \mathcal{L}(Q)$;
- $\hat{s}(Q, D) \leftarrow \texttt{rank}(q_{1\ldots n}, d_{1\ldots m}, f_{\text{enc}})$;
- $\hat{D}_{1,\ldots,k} = \texttt{Argmax}(\hat{s}(Q, D), k)$;

return $\hat{D}_{1,\ldots,k}$

as GPT-4 (OpenAI, 2023b), Gemini API (Google, 2024), and open-source models like LLaMa (Touvron et al., 2023). The prompt template is detailed in Appendix A.

To introduce variation and cover the potential query space, we increase the sampling temperature and sample the model response multiple times. Examples of the query generator outputs can be found in Appendix B. Finally, each synthesized query is concatenated with the original tool document to create augmented tool documents, facilitating better tool retrievals (see Fig. 2).

3.2 Intent Extractor

Tool augmented LLM agents often function as chatbots, interacting with users who express their intents in diverse and potentially verbose ways. Users may unconsciously provide extraneous background information before stating their actual tasks, which can confuse downstream tool retrieval when trying to identify the underlying intents (Qian et al., 2024). Additionally, users might express multiple intents in a single conversational query, which current retrieval system may struggle to capture due to the query complexity. To address these, we leverage LLM's reasoning and query understanding capabilities through in-context learning to extract tool-related intents, thereby improving retrieval accuracy. This approach also allows for the effective extraction of multiple intents if the user query contains different requests. We then encode these intents replacing the original user queries into the embedding space during tool retrieval. This technique enables the retrieval system to recommend all the relevant tools for each individual intent (see Fig. 2). The prompt template for extracting user intent using LLMs is available in Appendix A.

3.3 Multi-view Similarity Ranking

As each intent extracted from the user queries could retrieve different relevant tools, we introduce a multi-view similarity ranking method to consider all tool-related intents expressed in the user query. We aggregate similarity scores between each intent and the expanded tool document. By incorporating multiple perspectives within the embedding space, it provides a robust measure of relevance between expanded tool documents and user queries.

We first aggregate the embedding values from multiple copies of the expanded tool document with synthetic queries to represent each tool document in the embedding space. Instead of grouping the entire tools retrieved by all the intents described in the user query before ranking, we rank the tools individually within each intent and retrieve the top tool from each intent until the specified number of candidates have been retrieved. To achieve this, we design an ordering function to consider both the rank of the retrieved tool within each intent and the similarity score value between the tool and the intent. The proposed formulation allows us to capture the relevance of each intent to different aspects of the tool document, as represented by the generated queries.

The multi-view similarity ranking algorithm is explained using an example in Fig. 3. The final retrieved 2 tools from the query include the top tool `book_flight` retrieved from the intent "book a flight from New York City to San Francisco this weekend" and the top tool `find_restaurant` retrieved from the intent "find highly rated restaurants in downtown San Francisco". The detailed implementation is described in Appendix C.

4 Experimental Settings

4.1 Benchmark Datasets

A variety of benchmark datasets containing tools across different domains have been proposed to assess tool-augmented LLMs. These include APIBench (Patil et al., 2023), API-Bank (Li et al., 2023), ToolBench (Xu et al., 2023), ToolAlpaca (Tang et al., 2023), ToolBench (Qin et al., 2023), ToolE (Huang et al., 2023) and ToolQA (Zhuang et al., 2023). To evaluate Re-Invoke's tool retrieval performance, we select ToolBench (Qin et al., 2023) and ToolE (Huang et al., 2023) datasets, as both datasets provide ground truth query and tool document pairs that reflect real-world scenarios. We use the same ToolBench dataset to evaluate the end to end performance when integrating the LLM agent with the proposed Re-Invoke retriever. Detailed data statistics on the benchmark datasets can be found in Appendix D.

Following the approach in Qin et al. (2023), we use nDCG@k metric[1] to evaluate retrieval performance on the benchmark datasets. We report nDCG@5 in the following sections and the detailed retrieval metrics including recall@k[2] can be found in Appendix E. For end-to-end performance evaluation, we use pass rate following the same evaluation protocol proposed in Qin et al. (2023).

4.2 Unsupervised Retrieval Baselines

As the proposed method is training free, we establish the following baselines to benchmark Re-Invoke's unsupervised tool retrieval performance:
(a) **Sparse retrieval using BM25**: We directly calculate relevance between the user query and the tool documents using BM25. We use the default normalization parameter $k = 1.5$ for term frequency and offset parameter $b = 0.75$ for document length normalization.
(b) **Dense retrieval using text embedding**: We encode both query and entire tool documents using Google Vertex AI's `textembedding-gecko@003` model[3] and compute the cosine similarity on the embedding values.
(c) **HyDE as a dense retrieval method**: Following Gao et al. (2023), we use LLMs to generate a hypothetical tool document for each user query. We then calculate document-document similarity using embeddings to retrieve the real tool document. In our experiment, we use Google Vertex AI's `text-bison@001` model[4] to generate the hypothetical tool document. The instruction template can be found in Fig. 6 in Appendix A. Both hypothetical and real tool documents are encoded using the Vertex AI text embedding API.

AnyTool (Du et al., 2024) proposes hierarchical agents by leveraging LLM as a tool retriever for large-scale API calls, but the "hierarchy of the tools" structure may not be available in general tool use cases. It also results in significantly higher latency and cost when the number of tools scale up. Therefore, this method is excluded in our baselines.

4.3 Re-Invoke

We use Google Vertex AI's `text-bison@001` model (with 0.7 temperature) in the query generator to generate 10 diverse synthetic queries per tool document. Other parameters remain default. We explored various concatenation methods (prepending, appending, repetition) and found they yield similar retrieval metrics. Therefore, we append each generated query to the original tool document in the format: `"Documentation: <tool document> Query: <predicted query>"` to create different copies of the expanded tool document. For evaluation, we vary the number of synthetic queries (see ablation study in Sec. 6.1).

[1] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ndcg_score.html
[2] https://www.tensorflow.org/ranking/api_docs/python/tfr/keras/metrics/RecallMetric
[3] https://cloud.google.com/vertex-ai/docs/generative-ai/embeddings/get-text-embeddings
[4] https://cloud.google.com/vertex-ai/docs/generative-ai/model-reference/text

Retrieval	Method	Backbone LLM	ToolBench			ToolE	
			I1	I2	I3	single-tool	multi-tool
Sparse	BM25	-	0.3475	0.2077	0.3375	0.3735	0.2635
	HyDE w/ Vertex AI	`text-bison@001`	0.3084	0.1627	0.2594	0.3770	0.1954
	Re-Invoke w/ BM25 (ours)	`text-bison@001`	0.5519	0.3968	**0.4990**	0.5971	0.5637
	Re-Invoke w/ BM25 (ours)	`gpt-3.5-turbo`	**0.6013**	**0.4230**	0.4959	**0.6300**	**0.5883**
	Re-Invoke w/ BM25 (ours)	`Mistral-7B`	0.5768	0.3964	0.4770	0.6134	0.5373
Dense	Vertex AI	-	0.5962	0.3880	0.4633	0.6522	0.5296
	HyDE w/ Vertex AI	`text-bison@001`	0.4336	0.2221	0.2996	0.6558	0.4910
	Re-Invoke w/ Vertex AI (ours)	`text-bison@001`	0.6110	**0.5379**	**0.5955**	**0.7821**	**0.7231**
	Re-Invoke w/ Vertex AI (ours)	`gpt-3.5-turbo`	0.6090	0.5068	0.5719	0.7705	0.6957
	Re-Invoke w/ Vertex AI (ours)	`Mistral-7B`	**0.6150**	0.5128	0.5771	0.7770	0.6959

Table 1: nDCG@5 metrics on ToolBench I1, I2, I3 and ToolE single-tool, multi-tool datasets using both sparse and retrieval methods. In the sparse retrieval method, we apply BM25 retrieval and HyDE retrieval based on BM25 as two baselines. `text-bison@001` is used to generate hypothesis documents in the HyDE method. We integrate Re-Invoke with BM25 embedding using `text-bison@001`, `gpt-3.5-turbo` and `Mistral-7B-Instruct-v0.3` as three different backbone LLMs. In the dense retrieval method, we apply Vertex AI text embedding retrieval and HyDE retrieval based on Vertex AI text embedding as two baselines. `text-bison@001` is used to generate hypothesis documents in the HyDE method. We also integrate Re-Invoke with the Vertex AI text embedding using `text-bison@001`, `gpt-3.5-turbo` and `Mistral-7B-Instruct-v0.3` as three different backbone LLMs. The highest nDCG@5 metric is marked in bold.

We use the same Google Vertex AI's LLM model in the intent extractor to synthesize the intents from the user queries. We then extract dense embedding vectors from both augmented tool documents and extracted intents using Google Vertex AI's `textembedding-gecko@003` model. We also apply Re-Invoke using BM25 embedding vectors as a sparse retrieval method.

In our Re-Invoke design, we average the embedding values from multiple copies of the expanded tool document as a representation of the tool document. We then compute the embedding similarity score between each extracted intent and the expanded tool documents, and rank the tool documents with the ordering function described in Sec. 3.3. We compare our designed aggregation function with others in Sec. 6.1.

5 Experimental Results

5.1 Baseline Retrieval Performance

As shown in Table 1, semantic retrieval using Vertex AI text embedding significantly outperforms the sparse retrieval BM25 across all five benchmark datasets. This aligns with the findings in Patil et al. (2023) and Qin et al. (2023). Even without specific pre-training on tool retrieval tasks, the existing enterprise text embedding API can effectively represent the semantic relationship between user queries and relevant tool documents.

Compared to its dense retrieval counterpart, HyDE retrieval using Vertex AI text embedding performs less favorably. This suggests that the HyDE approach introduces a concept drift between the actual and hypothetical tool documents. The metric degradation can likely be attributed to information loss within the hypothetical documents.

5.2 Retrieval Performance of Re-Invoke

Re-Invoke consistently outperforms both sparse and dense retrieval baselines across all benchmark datasets, as shown in Table 1 (See Table 6 in Appendix for complete results). When combined with BM25 sparse embeddings, nDCG@5 is significantly increased. Similarly, Re-Invoke with Vertex AI text embedding yields significant performance gains. This improvement stems from the proposed LLM-powered tool document enrichment and user intent extraction.

The application of Re-Invoke significantly improves both sparse and dense retrieval performance although we observe that applying Re-Invoke on top of the sparse retrieval method still underperforms the dense retrieval counterparts. To further analyze the impact of Re-Invoke on improving retrievals, we examine specific user queries and compare the retrieved tools between the baseline and Re-Invoke (see Appendix F).

We also replicate our experiment using OpenAI's `gpt-3.5-turbo` model[5] and Mistral AI's

[5]https://platform.openai.com/docs/models/gpt-3-5-turbo

Tool Retriever	I1 (%)			I2 (%)		I3 (%)	Average (%)
	Instruction	Tool	Category	Instruction	Category	Instruction	
None	39.70	44.72	47.50	64.50	55.33	61.00	52.13
ToolLLM's	47.50	42.00	53.00	62.50	56.78	54.00	52.63
Re-Invoke (ours)	**48.00**	**49.75**	**53.03**	**65.33**	**58.29**	**62.00**	**56.07**

Table 2: End-to-end performance on the ToolBench datasets. We follow Qin et al. (2023) to use pass rate as the evaluation metric. We integrate ToolLLaMA with DFSDT as the agent, using a set of reference tools without a retriever, ToolLLM's retriever, and Re-Invoke retriever. The highest performance metric is marked in bold.

`Mistral-7B-Instruct-v0.3` model (Jiang et al., 2023) as the backbone LLMs. The same backbone LLM is used in both the query generator and intent extractor. Applying the same settings including prompt and decoding parameters as those described in `text-bison@001`, Re-Invoke achieves a similar trend across all benchmark datasets (see Table 1). This demonstrates Re-Invoke's compatibility with various foundation models to improve the baseline retrieval methods.

5.3 End-to-End Performance Evaluation

We employ the proposed Re-Invoke as the tool retriever and the ToolLLaMA with Depth-First Search-Based Decision Tree (DFSDT) approach as the agent. Comprehensive implementation details can be found in ToolLLM (Qin et al., 2023). We adopt the pass rate metric proposed in ToolLLM (Qin et al., 2023) for evaluation metrics. Pass rate calculates the percentage of instructions successfully completed within limited budgets. We evaluate on six subsets of the ToolBench benchmark dataset: *I1-Instruction*, *I1-Category*, *I1-Tool*, *I2-Instruction*, *I2-Category* and *I3-Instruction*, using OpenAI's `gpt-3.5-turbo` model as an evaluator.

We compare the agent performance with different tool retriever settings: using a set of reference tools without retrievers, ToolLLM's API retriever (Qin et al., 2023) trained using the labeled query-tool pairs, and our Re-Invoke retriever without any training data. The reference set of tools are provided in the ToolBench dataset, but they might not be the ground-truth tools as same task could be solved with a different set of tools. All pass-rate evaluation results are reproduced. Table 2 demonstrates that our unsupervised Re-Invoke retriever outperforms both baselines with the set of reference tools and a trained tool retriever across all the benchmark datasets. This aligns with the finding in ToolLLM (Qin et al., 2023) that a tool retriever can expand the search space to find more appropriate tools for a given task. Therefore, using tools retrieved by Re-Invoke can improve the agent performance by suggesting more relevant tools given the task even compared to using the reference toolset. These evaluation results provide evidence that our Re-Invoke retriever can effectively retrieve relevant tools from a vast pool (16,000+ APIs) and it leads to more reliable downstream agent behaviors on the tool use. Importantly, Re-Invoke is completely unsupervised, eliminating the needs for training.

6 Discussions

6.1 Ablation Studies

Re-Invoke component evaluation. In this study, we evaluate the tool retrieval performance on each individual Re-Invoke components: query generator and intent extractor using Vertex AI text embedding API. The results in Table 3(A) provide the evidence that both designed components contribute positively to final retrieval metrics in Re-Invoke. Specifically, we observe consistent retrieval performance improvement across all the benchmark datasets when integrating the query generator with the baseline retrieval method. Similar improvement is also demonstrated when applying intent extractor with the baseline. When integrating both query generator and intent extractor, Re-Invoke achieves the highest retrieval metrics. Note that no improvement is observed on ToolBench I1 dataset when applying intent extractor mainly because ToolBench I1 dataset consists of APIs under the same tool, and each individual intent retrieves overlapped set of APIs. We discuss the scenarios when each component performs better in Sec. 6.3.

Query generator evaluation. We investigate the retrieval performance with different document augmentation settings using the query generator alone including (1) whether to append the document with the synthetic query, (2) number of synthetic queries and (3) aggregation function. Table 3(B,C,D)

Method	ToolBench			ToolE	
	I1	I2	I3	single-tool	multi-tool
(A) INCLUDING CRITICAL COMPONENTS IN RE-INVOKE					
Baseline	0.5962	0.3880	0.4633	0.6522	0.5296
+ Query generator (Sec. 3.1)	**0.6286**	0.4135	0.4906	0.7813	0.6906
+ Intent extractor (Sec. 3.2)	0.5910	0.5157	0.5843	0.6756	0.6258
+ Query generator & Intent extractor (Re-Invoke)	0.6110	**0.5379**	**0.5955**	**0.7821**	**0.7231**
(B) USING GENERATED QUERIES ONLY IN QUERY GENERATOR					
Synthetic query only	0.4924	0.3050	0.4121	0.7535	0.6814
Appending the synthetic query to the document	**0.6286**	**0.4135**	**0.4906**	**0.7813**	**0.6906**
(C) VARYING THE NUMBER OF SYNTHETIC QUERIES IN QUERY GENERATOR					
1 synthetic query	0.5962	0.3741	0.4543	0.7388	0.6503
5 synthetic queries	0.6242	0.4091	0.4882	0.7777	0.6724
10 synthetic queries	**0.6286**	**0.4135**	**0.4906**	**0.7813**	**0.6906**
(D) VARYING THE AGGREGATION FUNCTION IN QUERY GENERATOR (MAX VS MEAN)					
Maximum similarity score	0.6104	0.3867	0.4760	0.7716	0.6333
Mean similarity score	**0.6286**	**0.4135**	**0.4906**	**0.7813**	**0.6906**

Table 3: nDCG@5 metrics of ablation studies on ToolBench I1, I2, I3 datasets and ToolE single-tool, multi-tool datasets. We evaluate the impact of each critical component in Re-Invoke: the query generator and the intent extractor. Within query generator component, we further compare nDCG@5 metrics across different number of synthetic queries, different aggregation functions to aggregate the relevance scores, and appending the synthetic queries to the tool documents or not. The highest retrieval metric is marked in bold.

Metric	ToolBench			ToolE
	I1	I2	I3	single-& multi-tool
recall@5	0.7787	0.7665	0.9043	0.9131
recall@10	0.8402	0.8311	0.9464	0.9462

Table 4: Round-trip consistency recall metrics on synthetic queries in the query generator.

clearly validate that our design choices in the query generator: appending the synthetic query to the tool document with 10 synthetic queries and aggregating the similarity scores on the augmented tool documents with mean function, outperforms the alternatives. We observe that replacing the tool document with the synthetic queries could lead to potential information loss during the retrieval stage and augmenting the tool document is preferred. Mean similarity score is more robust when considering diverse synthetic queries in the query generator. Increasing the number of diverse synthetic queries improves the retrieval performance and demonstrates the effectiveness in enriching the tool documents with diverse synthetic queries.

6.2 Round-Trip Consistency Evaluation

We define the synthetic query quality using the round-trip consistency criteria (Alberti et al., 2019), *i.e.*, the synthetic queries should retrieve the same tool documents used to generate them. Specifically, we compute recall@k as a round-trip consistency metric to quantify if the tool document used to generate the synthetic query are among the top-k documents after retrieval using the synthetic query (see Table 4). The relatively high recall@10 metric across all the benchmark datasets suggests query generator's effectiveness to distinguish highly similar tool documents during retrieval. The round-trip consistency recall metric is lower on ToolBench I1 and I2 datasets, mainly caused by the larger toolset size compared to ToolBench I3 and ToolE datasets.

6.3 Re-Invoke Performance Analysis

We analyze the Re-Invoke performance under different scenarios from the results in Table 3(A). For relatively short tool documents with minimal human-readable descriptions, tool document expansion alone can significantly boost retrieval performance through generative relevance feedback. The documents in the ToolE datasets only include the tool name and descriptions and we can see that *query generator* alone achieves larger performance gains. However, if a tool document lacks API and parameter descriptions, LLMs may struggle to accurately infer usage, relying solely on names. This can lead to generated queries that do not reflect real-world tool usage scenarios. In contrast, for complex user queries with extensive background context or requiring multiple tools simultaneously, *intent extractor* becomes crucial for improving tool retrieval performance. This component ensures individual tool-related contexts are extracted effectively. ToolBench I2 and I3 datasets both contain the queries that need to be handled by calling the APIs from multiple tools and categories on RapidAPI hub, *intent extractor* component alone achieves more significant retrieval performance gains.

6.4 Latency and Computational Cost Analysis

The query generator in Re-Invoke creates the synthetic queries offline when the tool documents are ingested, resulting no additional latency during serving. The number of LLM calls during tool document indexing is proportional to the number of synthetic queries per document and the total number of documents. On the other hand, intent extractor employs an LLM to extract the user intents from the user query during online inference, which incurs an extra LLM call with associated latency and computational cost.

To mitigate the latency and cost increase, techniques such as knowledge distillation from larger to smaller LLMs, or quantization, offer promising avenues for reducing both latency and computational overhead.

7 Conclusion

In this work, we present Re-Invoke, a fully unsupervised tool retrieval approach designed to scale LLM tool learning to large toolsets. We leverage LLMs to enhance tool document context with diverse synthetic queries, extract essential tool-related intents into executable requests through intent extraction. Re-Invoke offers a fresh perspective on scalable tool retrieval, prioritizing context enhancement and intent understanding without any training data.

8 Limitations

Synthetic query diversity and quality. Re-Invoke achieves query diversity through simple LLM response sampling with a zero-shot prompt. To further enhance the quality and reduce concept drift between synthetic and real-world queries, more sophisticated query generation methods could be explored. This might include techniques such as controlled prompting, iterative refinement, or utilizing external knowledge bases.

Intent extractor. Re-Invoke relies on in-context learning and LLM's internal knowledge to extract tool related intents. Future work could include using the downstream agent's execution results as a feedback to refine intent extraction.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

Abhijit Anand, Vinay Setty, Avishek Anand, et al. 2023. Context aware query rewriting for text rankers using llm. *arXiv preprint arXiv:2308.16753*.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2023. Large language models as tool makers. *arXiv preprint arXiv:2305.17126*.

Yung-Sung Chuang, Wei Fang, Shang-Wen Li, Wen-tau Yih, and James Glass. 2023. Expand, rerank, and retrieve: Query reranking for open-domain question answering. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12131–12147, Toronto, Canada. Association for Computational Linguistics.

Yu Du, Fangyun Wei, and Hongyang Zhang. 2024. Anytool: Self-reflective, hierarchical agents for large-scale api calls. *arXiv preprint arXiv:2402.04253*.

Jiazhan Feng, Chongyang Tao, Xiubo Geng, Tao Shen, Can Xu, Guodong Long, Dongyan Zhao, and Daxin Jiang. 2023. Knowledge refinement via interaction between search engines and large language models. *arXiv preprint arXiv:2305.07402*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

Shen Gao, Zhengliang Shi, Minghang Zhu, Bowen Fang, Xin Xin, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. 2024. Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18030–18038.

Google. 2023a. Bard can now connect to your google apps and services.

Google. 2023b. Introducing gemini: our largest and most capable ai model.

Google. 2024. Gemini api.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. In *Advances in Neural Information Processing Systems*, volume 36, pages 45870–45894. Curran Associates, Inc.

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*.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, et al. 2023. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*.

Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Yilun Kong, Jingqing Ruan, Yihong Chen, Bin Zhang, Tianpeng Bao, Shiwei Shi, Guoqing Du, Xiaoru Hu, Hangyu Mao, Ziyue Li, et al. 2023. Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world systems. *arXiv preprint arXiv:2311.11315*.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. Paq: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115.

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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, Singapore. Association for Computational Linguistics.

Jerry Liu. 2022. LlamaIndex.

Linqing Liu, Minghan Li, Jimmy Lin, Sebastian Riedel, and Pontus Stenetorp. 2022. Query expansion using contextual clue sampling with language models. *arXiv preprint arXiv:2210.07093*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

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. In *Advances in Neural Information Processing Systems*, volume 36, pages 43447–43478. Curran Associates, Inc.

Guangyuan Ma, Xing Wu, Peng Wang, Zijia Lin, and Songlin Hu. 2023a. Pre-training with large language model-based document expansion for dense passage retrieval. *arXiv preprint arXiv:2308.08285*.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023b. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315, Singapore. Association for Computational Linguistics.

Iain Mackie, Shubham Chatterjee, and Jeffrey Dalton. 2023. Generative relevance feedback with large language models. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2026–2031.

Kelong Mao, Zhicheng Dou, Fengran Mo, Jiewen Hou, Haonan Chen, and Hongjin Qian. 2023. Large language models know your contextual search intent: A prompting framework for conversational search. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1211–1225, Singapore. Association for Computational Linguistics.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*.

OpenAI. 2023a. Chatgpt plugins.

OpenAI. 2023b. Gpt-4.

OpenAI. 2023c. Gpt-4 technical report.

Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*.

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, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Tell me more! towards implicit user intention understanding of language model driven agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1088–1113, Bangkok, Thailand. Association for Computational Linguistics.

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*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551. Curran Associates, Inc.

Tao Shen, Guodong Long, Xiubo Geng, Chongyang Tao, Yibin Lei, Tianyi Zhou, Michael Blumenstein, and Daxin Jiang. 2024. Retrieval-augmented retrieval: Large language models are strong zero-shot retriever. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 15933–15946, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*.

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*.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9414–9423, Singapore. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

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*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Fanghua Ye, Meng Fang, Shenghui Li, and Emine Yilmaz. 2023. Enhancing conversational search: Large language model-aided informative query rewriting. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5985–6006.

Shi Yu, Jiahua Liu, Jingqin Yang, Chenyan Xiong, Paul Bennett, Jianfeng Gao, and Zhiyuan Liu. 2020. Few-shot generative conversational query rewriting. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 1933–1936.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Ren Kan, Dongsheng Li, and Deqing Yang. 2024. Easytool: Enhancing llm-based agents with concise tool instruction. *arXiv preprint arXiv:2401.06201*.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2299–2314, Mexico City, Mexico. Association for Computational Linguistics.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. In *Advances in Neural Information Processing Systems*, volume 36, pages 50117–50143. Curran Associates, Inc.

Appendix

A Prompt Templates

In this section, we list all the prompt templates used in our experiments. Fig. 4 is the prompt to generate a synthetic query from the tool document. Fig. 5 shows the prompt to extract the user intents. Fig. 6 describes the prompt to generate hypothetical tool document given a user query used in the HyDE retrieval baseline.

```
Suppose you are an assistant and you have access to the following API to answer user's queries.
You are provided with a tool and its available API function including the description and
parameters.

Your task is to generate a possible user query that can be handled by the API.

You must include the input parameters required in the API call. Please be creative and generate
random but specific information for the required parameters.
Now you are given the API documentation below:

<tool document>

Please generate a user query that you will need to call this tool. Note the generated query should
be complex enough to describe the scenarios that you will need to call the provided API to address
them.

The relevant query is:
```

Figure 4: Prompt template to generate the synthetic queries from the tool document.

```
**Instructions**
Suppose you are a query analyzer and your task is to extract the underlying user intents from the
input query. You should preserve all the underlying user request and the extracted user intents
should be easily understood without extra context information.
You should carefully read the given user query to understand its different intents. Then identify
what are the specific intents. Each individual intent should be separated by a newline.

Here are some examples of how you should solve the task.

**Example**
Query: I'm planning to travel to Paris next weekend to visit my family, could you help me book a
round trip flight ticket? I want to fly in economy class.
Intent:
book a round-trip flight ticket in economy class to Paris next weekend

Query: I'm a potential buyer looking for a condominium in the city of Miami. I am specifically
interested in properties that have a minimum of two bathrooms. It should have walkable distance to
the grocery stores.
Intent:
buy a real estate in Miami with a minimum of two bathrooms and walkable distance to the grocery
stores

Query: I want to learn Spanish by talking to the native speakers at any time. Additionally, can you
recommend some interesting books, preferably fictions, so that I can learn by reading? Also include
the websites that I can buy them.
Intent:
learn Spanish by talking to the native speakers
recommend fictions to learn Spanish by reading
suggest the websites to buy Sanish fictions

**Begin!**
Query: <user query>
Intent:
```

Figure 5: Prompt template to extract the underlying intents from user queries.

```
Suppose you are an assistant and you have access to the API documentation to answer user's queries.
Please generate an API documentation in the JSON format that can be called to handle this query.
The API documentation should be general enough to handle the cases beyond the provided queries.
Please provide detailed descriptions on the parameters.

**Examples**
Query: I'm planning to travel to Paris next weekend to visit my family, could you help me book a
round trip flight ticket? I want to fly in economy class.
The API documentation is:
{
    "api_name": "flights",
    "api_description": "Search the flight ticket on a specific travel date."
    "required_parameters": [
        {
            "name": "departure_date",
            "type" DATETIME,
            "description": "The departure date for the flight."
        },
        {
            "name": "from",
            "type" STRING,
            "description": "The city where the flight departs."
        },
        {
            "name": "to",
            "type" STRING,
            "description": "The city where the flight arrives."
        },
        {
            "name": "fare_class",
            "type": STRING,
            "description": "The fare class for the flight, economy, business or first."
        }
    ],
    "optional_parameters": [
        {
            "name": "return_date",
            "type": DATETIME,
            "description": "The return date for the flight."
        }
    ]
}

**Begin!**
Query: <user query>
The API documentation is:
```

Figure 6: One-shot prompt template to generate hypothetical tool document given a user query.

B Example Synthetic Queries

We show 10 different generated user queries from the documentation newsSearch API in the ToolBench dataset in Fig. 7.


```
{"category_name": "Data", "tool_name": "Web Search", "api_name": "newsSearch", "api_description":
"Get news articles relevant for a given query.", "required_parameters": [{"name": "pageSize",
"type": "NUMBER", "description": "The number of items per page. The maximum value is 50.",
"default": "10"}, {"name": "autoCorrect", "type": "BOOLEAN", "description": "Automatically correct
spelling.", "default": true}, {"name": "q", "type": "STRING", "description": "The user's search
query string.", "default": "taylor swift"}, {"name": "pageNumber", "type": "NUMBER", "description":
"The page to view.", "default": "1"}], "optional_parameters": [{"name"": "toPublishedDate", "type":
"STRING", "description": "The published date and time for the newest article allowed. For example:
*2015-05-16T05:50:06.* See  [https://www.c-sharpcorner.com/blogs/date-and-time-format-in-c-sharp
-programming1 ](url)for more possible DateTime formats.", "default": "null"}, {"name":
"safeSearch", "type": "BOOLEAN", "description": "A filter used to filter results for adult
content.", "default": false}, {"name": "fromPublishedDate", "type": "STRING", "description": "The
published date and time for the oldest article allowed. For example:  *2015-05-16T05:50:06.* See
[https://www.c-sharpcorner.com/blogs/date-and-time-format-in-c-sharp-programming1 ](url)for more
possible DateTime formats. "", "default": "null"}, {"name": "withThumbnail"", "type": "BOOLEAN",
"description": "Show results with image thumbnails.", "default": false}], "method": "GET"}
```


```
Show me news articles about the latest coronavirus outbreak in the United States. I would like to
see the articles from the past week, and I would like them to be safe for work.

I would like to see a list of news articles about the latest developments in the field of
artificial intelligence. Please show me the results from the past year and make sure that they are
safe for work.

I'd like to get a list of news articles about the latest developments in the field of artificial
intelligence. Please show me articles from the past 24 hours, and include images with the results.

I would like to get a list of news articles about the latest developments in artificial
intelligence. Please show me the results in chronological order, with the most recent articles
first. I would also like to see images of the articles.

I want to get the news articles about the latest news about the new album by Taylor Swift. I want
the results to be from the past month and I want them to be safe for work.

I want to know the latest news about the war in Ukraine. Please show me the results from the past
24 hours, and make sure that they are safe for work.

I want to find news articles about the latest developments in the field of artificial intelligence.
Please show me articles from the past year that are safe for work and have image thumbnails.

I would like to find news articles about the latest developments in AI. Please show me the results
from the past month and make sure to include images in the results.

I would like to know about the latest news articles on the topic of artificial intelligence.
Please show me the results in chronological order, with the newest articles first. I would also
like to see thumbnails of the articles.

I would like to find news articles about the latest developments in artificial intelligence.
Please show me the results from the past week and make sure they are safe for work.
```

Figure 7: The document on the example newsSearch tool and 10 different synthetic queries that are relevant to the provided tool.

C Multi-view Similarity Ranking Algorithm Implementations

In this section, we describe the multi-view similarity ranking algorithm (Sec. 3.3) implementations. As each tool document is concatenated with a synthetic query to create m copies, we iterate each augmented tool document and compute the average embedding on each copy of the same tool document. We iterate each user intent from a user query and each tool document to compute the embedding similarity score between the individual intent and tool document. Within each intent, we also compute the reversed ranking order of the tools using the similarity score. To allow each intent to be considered during retrieval, we use a tuple to represent the ranking score to include both the reversed ranking order and similarity score value: we compare the reversed ranking order followed by the similarity score value if the reversed ranking order is the same. We then group all the intents and retrieve the top k documents based on the ranking score. The pseudo-code is available in Algorithm 2, with a detailed working example in Fig. 8.

Algorithm 2: Pseudo-code of Re-Invoke's ranking method

Data: n extracted intents $q_1, q_2, ..., q_n$ from query Q, List of tool documents D with each document d concatenated with m synthetic queries and each concatenated document is denoted as d_1, d_2,..., d_m, Text embedding model f_{enc}

Result: a retrieval score to rank the documents given a user query

Function $\texttt{rank}(q_{1...n}, d_{1...m}, f_{enc})$**:**
  for $d \in D$ **do**
    $E_d \leftarrow \frac{1}{m}\sum_{i=1}^{m}(f_{\text{enc}}(d_i))$;
  end
  for $i = 1, .., n$ **do**
    for $d \in D$ **do**
      $\hat{s}(q_i, d) \leftarrow f_{\text{enc}}(q_i) \cdot E_d$;
      $\text{rank}(q_i, d) \leftarrow \hat{s}(q_i, d).\texttt{rank}(\texttt{reversed} = \texttt{True}, \texttt{axis} = 1)$;
      $r(q_i, d) \leftarrow (\text{rank}(q_i, d), \hat{s}(q_i, d))$;
    end
  end
  $r(Q, D) \leftarrow \texttt{max}_{i=1,2,...,n} r(q_i, D)$;
return $r(Q, D)$

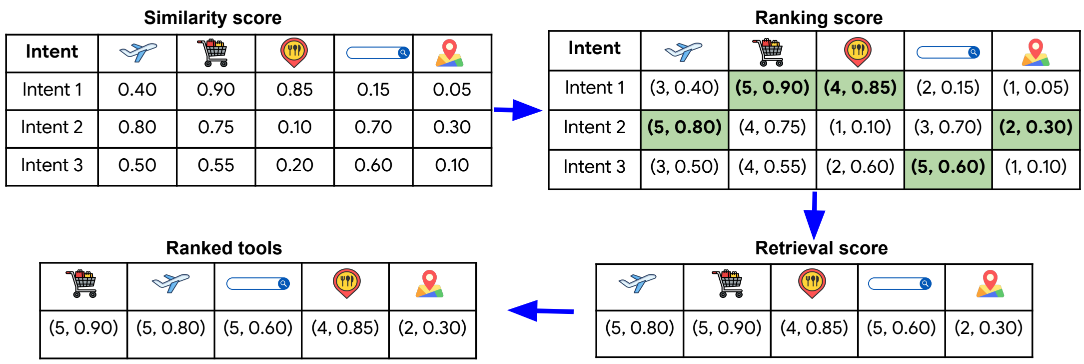

Figure 8: An example of the multi-view similarity ranking algorithm. From the intent-tool similarity score, we define the ranking score as a tuple of the reversed ranking order (*i.e.*, lowest similarity score will have a reversed ranking order of 1) for each tool within the same intent and similarity score. We then find the maximum ranking score across multiple intents for each tool document to compute the retrieval score. The retrieval score will be used to retrieve and rank the top tools given the user query.

D Data statistics

The statistics for the benchmark datasets of ToolBench I1, I2, I3 and ToolE single-tool and multi-tool are shown in Table 5.

Dataset name	**Number of queries**	**Number of tools**	**Number of labeled pairs**
ToolBench I1	87,419	10,439	424,169
ToolBench I2	84,815	13,142	220,832
ToolBench I3	25,251	1,605	72,324
ToolE single-tool	20,550	199	20,550
ToolE multi-tool	497	199	994

Table 5: Data statistics on ToolBench and ToolE benchmark datasets including number of queries, number of tool documents and number of labeled pairs.

E Retrieval performance evaluation

The complete retrieval metrics including nDCG@1, nDCG@5, recall@1 and recall@5 on all the benchmark datasets of ToolBench I1, I2, I3 and ToolE single-tool and multi-tool are shown in Table 6.

Dataset	**Retrieval**	**Method**	**LLM**	**nDCG**		**recall**	
				@1	**@5**	**@1**	**@5**
ToolBench I1	Sparse	BM25	-	0.4108	0.3588	0.1281	0.3135
		HyDE w/ BM25	`text-bison@001`	0.3449	0.3084	0.1073	0.2729
		Re-Invoke w/ BM25 (ours)	`text-bison@001`	0.6338	0.5519	0.2000	0.4767
		Re-Invoke w/ BM25 (ours)	`gpt-3.5-turbo`	**0.6809**	**0.6013**	**0.2167**	**0.5225**
		Re-Invoke w/ BM25 (ours)	`Mistral-7B`	0.6588	0.5768	0.2089	0.4993
	Dense	Vertex AI	-	0.6461	0.5962	0.2069	0.5278
		HyDE w/ Vertex AI	`text-bison@001`	0.4700	0.4336	0.1508	0.3892
		Re-Invoke w/ Vertex AI (ours)	`text-bison@001`	**0.6947**	0.6110	**0.2231**	0.5392
		Re-Invoke w/ Vertex AI (ours)	`gpt-3.5-turbo`	0.6779	0.6090	0.2209	0.5413
		Re-Invoke w/ Vertex AI (ours)	`Mistral-7B`	0.6847	**0.6150**	0.2207	**0.5441**
ToolBench I2	Sparse	BM25	-	0.2543	0.2168	0.1091	0.2201
		HyDE w/ BM25	`text-bison@001`	0.1940	0.1627	0.0839	0.1641
		Re-Invoke w/ BM25 (ours)	`text-bison@001`	0.4722	0.3968	0.2007	0.3980
		Re-Invoke w/ BM25 (ours)	`gpt-3.5-turbo`	**0.4982**	**0.4230**	**0.2117**	**0.4271**
		Re-Invoke w/ BM25 (ours)	`Mistral-7B`	0.4691	0.3964	0.1991	0.3990
	Dense	Vertex AI	-	0.4451	0.388	0.1902	0.3976
		HyDE w/ Vertex AI	`text-bison@001`	0.2661	0.2221	0.1144	0.2235
		Re-Invoke w/ Vertex AI (ours)	`text-bison@001`	**0.5456**	**0.5379**	**0.2315**	**0.5642**
		Re-Invoke w/ Vertex AI (ours)	`gpt-3.5-turbo`	0.5021	0.5068	0.2134	0.5373
		Re-Invoke w/ Vertex AI (ours)	`Mistral-7B`	0.5133	0.5128	0.2174	0.5426
ToolBench I3	Sparse	BM25	-	0.4075	0.3452	0.1601	0.3480
		HyDE w/ BM25	`text-bison@001`	0.2965	0.2594	0.1162	0.2655
		Re-Invoke w/ BM25 (ours)	`text-bison@001`	**0.5768**	**0.4990**	**0.2253**	**0.5052**
		Re-Invoke w/ BM25 (ours)	`gpt-3.5-turbo`	0.5660	0.4959	0.2213	0.5052
		Re-Invoke w/ BM25 (ours)	`Mistral-7B`	0.5432	0.4770	0.2124	0.4868
	Dense	Vertex AI	-	0.5165	0.4633	0.2029	0.4793
		HyDE w/ Vertex AI	`text-bison@001`	0.3475	0.2996	0.1376	0.3048
		Re-Invoke w/ Vertex AI (ours)	`text-bison@001`	**0.5965**	**0.5955**	**0.2327**	**0.6242**
		Re-Invoke w/ Vertex AI (ours)	`gpt-3.5-turbo`	0.5764	0.5719	0.2248	0.6016
		Re-Invoke w/ Vertex AI (ours)	`Mistral-7B`	0.5833	0.5771	0.2271	0.6047
ToolE single-tool	Sparse	BM25	-	0.2716	0.3732	0.2716	0.4618
		HyDE w/ BM25	`text-bison@001`	0.3121	0.3770	0.3121	0.4337
		Re-Invoke w/ BM25 (ours)	`text-bison@001`	0.4928	0.5971	0.4927	0.6866
		Re-Invoke w/ BM25 (ours)	`gpt-3.5-turbo`	**0.5255**	**0.6300**	**0.5255**	**0.7193**
		Re-Invoke w/ BM25 (ours)	`Mistral-7B`	0.5021	0.6134	0.5021	0.7093
	Dense	Vertex AI	-	0.5265	0.6522	0.5265	0.7574
		HyDE w/ Vertex AI	`text-bison@001`	0.5488	0.6558	0.5488	0.7448
		Re-Invoke w/ Vertex AI (ours)	`text-bison@001`	**0.6716**	**0.7821**	**0.6715**	**0.8707**
		Re-Invoke w/ Vertex AI (ours)	`gpt-3.5-turbo`	0.6551	0.7705	0.6551	0.8635
		Re-Invoke w/ Vertex AI (ours)	`Mistral-7B`	0.6600	0.7770	0.6600	0.8714
ToolE multi-tool	Sparse	BM25	-	0.1841	0.2627	0.0926	0.3350
		HyDE w/ BM25	`text-bison@001`	0.2414	0.1954	0.1207	0.1942
		Re-Invoke w/ BM25 (ours)	`text-bison@001`	0.5392	0.5637	0.2696	0.6408
		Re-Invoke w/ BM25 (ours)	`gpt-3.5-turbo`	**0.5634**	**0.5883**	**0.2817**	**0.6610**
		Re-Invoke w/ BM25 (ours)	`Mistral-7B`	0.4909	0.5373	0.2455	0.6157
	Dense	Vertex AI	-	0.4286	0.5296	0.2143	0.6258
		HyDE w/ Vertex AI	`text-bison@001`	0.5614	0.4910	0.2807	0.5050
		Re-Invoke w/ Vertex AI (ours)	`text-bison@001`	**0.6660**	**0.7231**	**0.3330**	**0.8008**
		Re-Invoke w/ Vertex AI (ours)	`gpt-3.5-turbo`	0.6499	0.6957	0.3249	0.7797
		Re-Invoke w/ Vertex AI (ours)	`Mistral-7B`	0.6076	0.6959	0.3038	0.7968

Table 6: Retrieval metrics (nDCG@1, nDCG@5, recall@1 and recall@5) on ToolBench I1, I2, I3 and ToolE single-tool and multi-tool datasets with different approaches including baselines and Re-Invoke using BM25 and Vertex AI text embedding. We observe the similar tool retrieval performance trend with different retrieval metrics. The highest metric is marked in bold.

F Case Study on Retrieved Tools

In this section, we showcase that tool retrieval can benefit from Re-Invoke using the ToolE dataset as demonstration examples. A few user queries are cherrypicked from the ToolE single-tool and multi-tool datasets that the baseline Vertex AI retriever retrieved the wrong tools while the Re-Invoke recommended the relevant tools. We investigate query generator and intent extractor components separately.

Table 7 shows the example queries that the correct tools are retrieved with the query generator component. It can be clearly seen that Re-Invoke's query generator component can better distinguish among similar tools to determine which tool is more relevant to user's request. For example, when the user is asking for the weather forecast for a location, the Vertex AI baseline retriever retrieves the very specific `airqualityforeast` tool while Re-Invoke retrieves the correct `WeatherTool` tool, which is more tailed to answer user's queries.

Table 8 lists the correct tools retrieved with the intent extractor component. Similarly, Re-Invoke's intent extractor effectively understands the user intents to recommend the most relevant tools to user's specific request. For example, when the user is asking for recommendations on online courses on machine learning and needs the access to relevant PDFs or URLs, Re-Invoke's intent extractor identifies two intents "recommend a course on machine learning" and "have access to relevant PDFs or URLs for further reading" and successfully retrieves the correct tools `CourseTool` and `PDF&URLTool` from each intent. However, the baseline retrieval method retrieves `CourseTool` and `search` tools instead.

We have also observed that Re-Invoke can still lead to wrong retrievals, especially when the tools are very similar, e.g., `HousePurchasingTool` and `HouseRentingToo`, `FinanceTool` and `CompanyInfoTool`. Please see the examples in Table 9. When the user is explicitly looking to buy a condominium in the query, Re-Invoke retrieves the wrong `HouseRentingTool`. We believe those errors can be reduced by designing a more sophisticated approach to generate more explicit synthetic queries that can be used to distinguish among confusing tool documents.

User query	Are there any strategy games, specifically turn-based and real-time strategy games, available for Nintendo Switch that offer a variety of gameplay options and customizable gameplay mechanics?
Baseline retrievals	{"name": "Chess", "description": "Unleash your inner chess master with this interactive chess experience! You can play against a novice or a grandmas-ter!"}
Re-Invoke retrievals	{ "name": "GameTool", "description":"Get game-related information and recommend games."}
User query	What are some popular attractions in London that I shouldn't miss? I'm specifically interested in historical landmarks, famous museums, and iconic landmarks that are unique to the city.
Baseline retrievals	{"name":"themeparkhipster", "description": "Find theme park waiting times around the world."}
Re-Invoke retrievals	{ "name": "TripAdviceTool", "description": "A comprehensive travel assistant that makes travel planning more vivid and practical. It offers tourism activities, accommodation and attraction recommendations, aiming to provide users with a more enjoyable and enriching travel experience through technology."}
User query	Can you help me find research papers on a specific topic?
Baseline retrievals	{"name": "ResearchHelper", "description": "Tool that offers additional functions beyond searching academic papers, such as generating mind maps, answering user questions and storing them in specific formats."}
Re-Invoke retrievals	{ "name": "ResearchFinder", "description": "Tool for searching academic papers."}
User query	Could you please provide me with the highly detailed weather forecast for Tokyo, the capital city of Japan, specifically for the upcoming four days?
Baseline retrievals	{"name": "airqualityforeast", "description": "Planning something outdoors? Get the 2-day air quality forecast for any US zip code."}
Re-Invoke retrievals	{ "name": "WeatherTool", "description": "Provide you with the latest weather information."}
User query	Could you please provide detailed information about the deployed smart contract of a specific ERC20 token on the Ethereum blockchain, including its address, source code, contract functions, and any associated events or transactions?
Baseline retrievals	{"name": "magi_codex", "description": "Ask about Magic: The Gathering cards, rules and interactions."}
Re-Invoke retrievals	{"name": "FinanceTool", "description": "Stay informed with the latest financial updates, real-time insights, and analysis on a wide range of options, stocks, cryptocurrencies, and more."}
User query	What questions can I ask about this YouTube video?
Baseline retrievals	{"name": "video_highlight", "description": "Explore, research, and interact with YouTube videos and personal videos."}
Re-Invoke retrievals	{"name": "VideoSummarizeTool", "description": "Generate summaries from YouTube video links, offer question-answering capabilities, analyze and interpret the content of YouTube videos, and support interactions with online video platforms such as YouTube and Daily Motion."}
User query	Despite submitting my resume to numerous companies and job openings, I have not received any responses or feedback regarding my application.
Baseline retrievals	{"name": "JobTool", "description": "Your Global Career Hub! Find diverse job opportunities, expert interview tips, and resume optimization guidance. Empowering job seekers worldwide on their path to success."}
Re-Invoke retrievals	{"name": "ResumeTool", "description": "Quickly create resumes and receive feedback on your resume."}

Table 7: A list of cherry-picked example queries from the ToolE single-tool dataset, including top 1 tool retrieved by the baseline and Re-Invoke's query generator using the Vertex text embedding API. Re-Invoke's query generator retrieves the correct tools (in green) while the baseline retrieves the wrong tools (in red).

User query	What are some popular investment options with good returns, and can you recommend a playlist to relax while I research them?
Baseline retrievals	{"name": "FinanceTool", "description": "Stay informed with the latest financial updates, real-time insights, and analysis on a wide range of options, stocks, cryptocurrencies, and more."} {"name": "portfoliopilot", "description": "Your AI investing guide: portfolio assessment, recommendations, answers to all finance questions."}
Re-Invoke retrievals	{"name": "FinanceTool", "description": "Stay informed with the latest financial updates, real-time insights, and analysis on a wide range of options, stocks, cryptocurrencies, and more."} {"name": "MusicTool", "description": "Create music playlists, search for music, and check out the latest music trends."}
User query	Could you provide me with news articles on renewable energy sources and any research papers exploring their effectiveness??
Baseline retrievals	{"name": "ResearchHelper", "description": "Tool that offers additional functions beyond searching academic papers, such as generating mind maps, answering user questions and storing them in specific formats."} {"name": "ResearchFinder", "description": "Tool for searching academic papers."}
Re-Invoke retrievals	{"name": "ResearchHelper", "description": "Tool that offers additional functions beyond searching academic papers, such as generating mind maps, answering user questions and storing them in specific formats."} {"name": "NewsTool", "description": "Stay connected to global events with our up-to-date news around the world."}
User query	Can you recommend me a course on machine learning? I want to learn more about the topic, and also have access to relevant PDFs or URLs for further reading.
Baseline retrievals	{"name": "CourseTool", "description": "Unlock a world of knowledge and growth with our comprehensive learning platform, offering a diverse range of courses from renowned providers like Coursera and Upskillr, personalized language learning, professional team information lookup, open course schedule discovery, and top-tier university content."} {"name": "search", "description": "Level up your design skills quickly with a wide range of design courses, interactive workshops and AI-guided mentorship."}
Re-Invoke retrievals	{"name": "PDF&URLTool", "description": "Interact with any PDF files, provide page references for fact-checking, support chatting via Google Drive links to AI-driven PDF summaries and analysis; engage in interactive conversations with websites, access links on the internet to fetch required information, including generating articles and intelligent assistance for interactions with source code."} {"name": "CourseTool", "description": "Unlock a world of knowledge and growth with our comprehensive learning platform, offering a diverse range of courses from renowned providers like Coursera and Upskillr, personalized language learning, professional team information lookup, open course schedule discovery, and top-tier university content."}
User query	Can you recommend any online courses for learning about natural language processing and a GitHub repository with relevant code examples?
Baseline retrievals	{"name": "CourseTool", "description": "Unlock a world of knowledge and growth with our comprehensive learning platform, offering a diverse range of courses from renowned providers like Coursera and Upskillr, personalized language learning, professional team information lookup, open course schedule discovery, and top-tier university content."} {"name": "AI2sql", "description": "Converts a natural language text into an SQL query."}
Re-Invoke retrievals	{"name": "RepoTool", "description": "Discover GitHub projects tailored to your needs, explore their structures with insightful summaries, and get quick coding solutions with curated snippets. Elevate your coding journey with RepoTool, your go-to companion for GitHub project exploration and code mastery."} {"name": "CourseTool", "description": "Unlock a world of knowledge and growth with our comprehensive learning platform, offering a diverse range of courses from renowned providers like Coursera and Upskillr, personalized language learning, professional team information lookup, open course schedule discovery, and top-tier university content."}

Table 8: A list of cherry-picked example queries from the ToolE multi-tool dataset, including 2 tools retrieved by the baseline and Re-Invoke's intent extractor component using the Vertex text embedding API. Re-Invoke's intent extractor identifies the intents (in green and blue) and retrieves the correct tools (in green and blue) while the baseline retrieves the wrong tools (in red).

User query	I'm working on a project about renewable energy sources. Can you help me find scientific publications related to this topic and generate bibtex bibliographies for them?
Baseline retrievals	{"name": "ResearchHelper", "description": "Tool that offers additional functions beyond searching academic papers, such as generating mind maps, answering user questions and storing them in specific formats."}
Re-Invoke retrievals	{ "name": "ResearchFinder", "description": "Tool for searching academic papers."}
User query	I'm a potential buyer looking for a condominium in the city of Miami. I am specifically interested in properties that have a minimum of two bathrooms.
Baseline retrievals	{"name": "HousePurchasingTool", "description": "Tool that provide all sorts of information about house purchasing"}
Re-Invoke retrievals	{ "name": "HouseRentingTool", "description": "Tool that provide all sorts of information about house renting"}
User query	Can you provide me with a list of restaurants in Japan that I can reserve a table at?
Baseline retrievals	{"name": "RestaurantBookingTool", "description": "Tool for booking restaurant"}
Re-Invoke retrievals	{ "name": "TripAdviceTool", "description": "A comprehensive travel assistantn that makes travel planning more vivid and practical. It offers tourism activities, accommodation and attraction recommendations, aiming to provide users with a more enjoyable and enriching travel experience through technology." }
User query	I'm looking for a luxurious hotel in Bali that offers stunning beach views, as I am planning a romantic getaway.
Baseline retrievals	{"name": "TripTool", "description": "Offer discounted hotel and accommodation bookings, along with personalized hotel and product searches, travel planning, image editing, and more, helping users easily plan their trips and find accommodation and transportation options."}
Re-Invoke retrievals	{ "name": "TripAdviceTool", "description": "A comprehensive travel assistantn that makes travel planning more vivid and practical. It offers tourism activities, accommodation and attraction recommendations, aiming to provide users with a more enjoyable and enriching travel experience through technology."}
User query	Users would like to know the exact percentage of the dividend yield for Coca-Cola stock based on its current market price and dividend payouts.
Baseline retrievals	{"name": "FinanceTool", "description": "Stay informed with the latest financial updates, real-time insights, and analysis on a wide range of options, stocks, cryptocurrencies, and more."}
Re-Invoke retrievals	{"name": "CompanyInfoTool", "description": "Obtain relevant information about global companies from databases or knowledge graphs."}
User query	I have a presentation scheduled in the near future, and I am seeking assistance in enhancing its impact. Specifically, I would greatly appreciate it if you could utilize your skills to paraphrase and rephrase significant elements extracted from my research article, thereby transforming them into highly captivating and attention-grabbing points.
Baseline retrievals	{"name": "SummarizeAnything_pr", "description": "Summarize YouTube videos, web pages, and PDF documents by providing a link. This is a free preview."}
Re-Invoke retrievals	{"name": "PolishTool", "description": "Elevate your content with our AI-powered tool, which utilizes advanced rewriting techniques to create more human-like expressions and foster creative inspiration."}

Table 9: A list of cherrypicked example queries from the ToolE single-tool dataset, including top 1 tool retrieved by the baseline and Re-Invoke using the Vertex text embedding API. Baseline retrieves the correct tool (in green), while Re-Invoke retrieves the wrong tool (in red).