Retrieval Models Aren't Tool-Savvy: Benchmarking Tool Retrieval for Large Language Models

Anonymous ACL submission

Abstract

Tool learning aims to augment large language models (LLMs) with diverse tools, enabling them to act as agents for solving practical tasks. Due to the limited context length of tool-using LLMs, adopting information retrieval (IR) models to select useful tools from large toolsets is a critical initial step. However, the performance of IR models in tool retrieval tasks remains underexplored and unclear. Most tool-use benchmarks simplify this step by manually pre-annotating a small set of relevant tools for each task, which is far from the real-world scenarios. In this paper, we propose TOOLRET, a heterogeneous tool retrieval benchmark comprising 7.6k diverse retrieval tasks, and a corpus of 43k tools, collected from existing datasets. We benchmark six types of models on TOOLRET. Surprisingly, even the models with strong performance in conventional IR benchmarks, exhibit poor performance on TOOLRET. This low retrieval quality degrades the task pass rate of tool-use LLMs. As a further step, we contribute a large-scale training dataset with over 200k instances, which substantially optimizes the tool retrieval ability of IR models.[1]

1 Introduction

Large language models (LLMs) have demonstrated remarkable progress across various natural language processing (NLP) tasks, such as text summarization (Chang et al., 2023). However, they suffer from inherent inabilities to interact with the physical world and access vast, up-to-date knowledge (Qin et al., 2024). To alleviate these drawbacks, *tool learning* is proposed to equip LLMs with external tools, augmenting them as agents to manipulate tools for practical task-solving (Qu et al., 2025b; Wang et al., 2024e).

In practical applications, retrieving useful tools from toolsets for LLM agents typically serves as the initial step (Wang et al., 2024c; Xu et al., 2024; Song et al., 2023). This step becomes particularly critical in real-world scenarios where the candidate tools are usually large-scale and many of them are similar in functionality (Qu et al., 2024a). However, most existing work (Guo et al., 2024; Qian et al., 2023) simplifies this retrieval process by manually pre-selecting a small set of 10-20 relevant tools for each evaluation task. For example, the ToolACE (Liu et al., 2024a) and ToolBench (Qin et al., 2023) annotate about 10 tools per task. While recent information retrieval (IR) techniques such as semantic matching (Qu et al., 2024a; Xu et al., 2024), can assist with tool retrieval, they are often trained on ad-hoc tool-use datasets, lacking comprehensive evaluation on diverse scenarios, especially for unseen tasks. To further explore the importance of tool retrieval, we conduct a pilot experiment on ToolBench (Qin et al., 2023). As shown in Figure 1, we observe that (i) the agent's performance substantially drops when replacing the officially annotated toolset with the retrieved tools; and (ii) even strong retrievers like colbertv2 (Santhanam et al., 2021a), struggle to retrieve target tools effectively. These findings highlight the necessity to (i) systematically evaluate IR models on diverse tool retrieval tasks; and (ii) analyze the impact of retrieval on the end-to-end task pass rate.

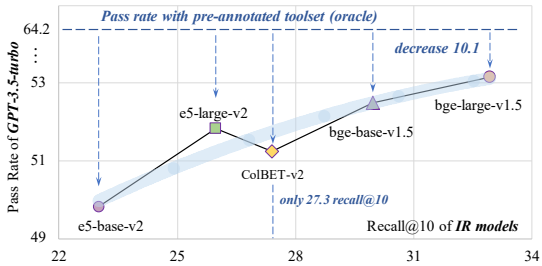

Figure 1: Correlation between the tool retrieval performance (e.g., Recall@10) of IR models and the end-to-end task pass rate of tool-use agents.

In this work, we introduce TOOLRET, the *first*

[1]Code is available on Anonymous GitHub.

large-scale tool retrieval benchmark comprising 7.6k diverse retrieval tasks and a corpus of 43k tools, which comprehensively evaluates IR models across diverse retrieval scenarios. Specifically, we collect query-tool datasets from the following sources: (i) Tool-use agent benchmarks from published research papers in AI conferences, such as ACL and NeurIPS; (ii) Related conference resources such as AppBench in EMNLP and ToolLens in CIKM; and (iii) Other publicly available datasets from the open-source community, e.g., HuggingFace. The collected data is carefully curated to cover a wide range of practical tool requirements, comprising diverse types of tool documentation, domains, and varying query lengths. Then, we standardize the format of all the collected tasks, aligning them with retrieval tasks similar to the format in **MTEB**, where *each retrieval task contains a query and target tools (e.g., labels)*. To support the instructional retrieval (Weller et al., 2024) setting of our benchmark, we also introduce a target-aware strategy to supplement each query with an instruction using the powerful LLMs (i.e., gpt-4o).

We systematically evaluate five types of IR models such as embedding models and LLM re-ranking, under various experimental settings. Our results reveal that even the best model (i.e., `NV-embedd-v1`) that demonstrates strong performance in conventional IR benchmarks, achieves an nDCG@10 of only 33.83 in our benchmark. This highlights the challenges of the tool retrieval tasks. We identify two key factors contributing to this performance gap: (i) Lower term overlap between queries and target tools in tool retrieval tasks, which demands higher representation abilities for IR models to accurately match query intent with the correct tools; and (ii) Task shift from conventional information-seeking tasks (e.g., document retrieval) to tool retrieval, leading to suboptimal performance of IR models that are not explicitly optimized.

To enhance the retrieval performance and enable IR models to augment tool-use agents, we further propose the TOOLRET-train, a large-scale training dataset containing more than 200k retrieval tasks. We extend our data collection process from TOOLRET to include the training set of three mainstream tool-use datasets, including ToolACE (Liu et al., 2024a), APIGen (Liu et al., 2024b) and ToolBench (Qin et al., 2023). To enable the training, we pair each retrieval task with 10 negative tools retrieved by the `NV-embed-v1`. Finally, each training example contains the query, an generated instruction, the target tools, and the negative tools. Results show that the IR models trained over TOOLRET-train, exhibit significant improvements in the retrieval process, leading to a higher end-to-end task pass rate when integrated with tool-use LLMs.

Our contributions are summarized as follows: (i) We introduce TOOLRET, the *first evaluation benchmark* for tool retrieval tasks. (ii) We evaluate the tool retrieval performance of various IR models and analyze the impact of retrieval on the end-to-end task pass rate of tool-use LLMs; and (iii) We contribute to a large-scale training dataset that enhances the performance of IR models, improving their ability to augment tool-use LLMs effectively.

2 Related work

Tool learning with foundation models. Tool learning aims to equip LLMs with tools, such as web API (Song et al., 2023) and python packages (Wang et al., 2024d), expanding their utility (Qin et al., 2023). Existing work teaching LLMs to use tools can be broadly classified into tuning-free (Lu et al., 2023) and tuning-based methods (Gao et al., 2024). The former prepends the description of candidate tools in the LLMs' context, prompting them to select and invoke tools (Huang et al., 2023). The latter enables LLMs to learn the usage of each tool through training on synthetic data (Liu et al., 2024a; Gao et al., 2024). However, both two paradigms struggle when facing the large-scale toolset in practice (Qu et al., 2024b; Liu et al., 2024b). First, real-world toolsets are typically massive, making it less possible to incorporate all tools within the limited context of LLMs. For example, the RapidAPI platform contains more than 52k tools while the PyPI[2] hosts over 600k frequently updated packages. Second, since tools are frequently updated, it is cost-intensive to re-train the LLMs to memorize all tools (Qu et al., 2025a). Although recent studies address this challenge using semantic retrievers (Qin et al., 2023; Wang et al., 2024c), these solutions are typically ad-hoc and lack systematic evaluation across diverse tool retrieval scenarios. To fill this gap, we present the *first comprehensive tool retrieval benchmark* with systematic analysis.

Information retrieval benchmark. Conventional information retrieval (IR) benchmarks are typically designed for information-seeking tasks, such as Nature Question (Kwiat kowski et al., 2019) for question answering and MS-MARCO (Nguyen

[2] https://pypi.org/

et al., 2016) for passage re-ranking. Recent work also explores the IR technique in various downstream tasks, such as table retrieval (Chen et al., 2024b; Zhang and Balog, 2020) and scientific retrieval (Ajith et al., 2024), which substantially augments the downstream models. However, tool retrieval, a crucial step for tool-use agents, remains underexplored. Compared with traditional IR tasks, retrieving useful tools is more challenging since solving a task typically requires the combination of multiple tools (Qu et al., 2024b). Most existing benchmarks simplify this retrieval process by manually annotating a small set of tools that fit the LLMs' context, which is far from reality with a large toolset. In this work, we evaluate IR models on diverse tool retrieval tasks and contribute over 200k training data to facilitate future research.

3 Benchmark construction

3.1 Data collection

To build a comprehensive benchmark for tool retrieval evaluation, we collect data from the following well-known sources: (i) *Tool-use LLM benchmarks*: A wide range of benchmarks published in leading AI conferences such as ACL and NeurIPS; (ii) *Conference Resources*: Datasets from resource tracks in IR and NLP conferences (e.g., CIKM and EMNLP); and (iii) *Other high-quality dataset*: We identify related datasets released on open-source platforms like HuggingFace and their technique reports can be found in public submissions like arXiv. We include them to enrich TOOLRET.

Given the rapid development of benchmarks from these sources, we collect datasets released between the *August 2023 to December 2024* in this version.[3] We download these data from official channels based on their usage requirements and totally collect more than 30 datasets. Since the data sources are diverse and their original formats vary substantially, we perform necessary data cleaning operations such as deduplication and text normalization to ensure consistency and quality.

We observe that most of the collected datasets are originally designed to evaluate the tool-use capability of LLMs, where the LLM is required to correctly call a sequence of target tools given an input query. To facilitate retrieval evaluation in TOOLRET, we align the format of all collected tasks with the well-established IR benchmark like *BEIR* and *MTEB*. Specifically, each task consists of a query as input and target tools as label (*a.k.a*, ground truth), where a tool is identified by a unique identifier and paired with detailed documentation to describe its functionality. Endpoints of the collected datasets and concrete examples of our formatted dataset are provided in Appendix B.

[3]Our team will maintain and update the benchmark.

3.2 Data sampling

After collecting the datasets, we observe data size imbalances across different datasets. Besides, some datasets are extremely large with substantial redundant content, making comprehensive model evaluation both inefficient and unnecessary. Therefore, we streamline them through effective data sampling while maintaining its evaluation integrity.

Task sampling. For each collected dataset, we encode the tasks using the embedding model, i.e., *NV-embedd-v1*, and apply the *K-means* clustering algorithm on the text embeddings. We set the number of clusters to the size of the corresponding toolset and randomly sample one task from each cluster. If the toolset size exceeds the number of queries, we retain all queries. For example, the original ToolEyes (Ye et al., 2024a) dataset contains 500 queries and 95 tools; Thus, we set the cluster number as `min(500, 95) = 95` for clustering.

Toolset sampling. To eliminate redundancy, we manually review the documentation of each raw dataset to identify and merge identical toolsets. For example, since the COLT (Qu et al., 2024a) toolset overlaps with the Toolbench (Qin et al., 2023) , we merge their intersecting tools. Ultimately, we merge all toolsets from the 34 datasets to form the final corpus, resulting in a total of 43k tools. Each tool is assigned a unique identifier.

After sampling, we obtain *7.6k retrieval tasks* and a corpus of *43k tools*.

3.3 Instruction construction

Instructional information retrieval (Sun et al., 2024; Weller et al., 2024) is an active research area, where an additional instruction is paired with the input query to guide IR models in retrieving target information. This instruction-following capability is especially critical in tool retrieval, as IR models are often used to augment LLM agents and receive additional context from the agents beyond the input query. To support this instructional IR scenario, we construct the instructions as part of TOOLRET.

Considering manually writing instructions is cost-intensive and challenging to scale, we introduce a *target-aware* strategy using powerful LLMs

Statistic	
# size of retrieval task	7,615
- # of *web API* retrieval task	4,916
- # of *code function* retrieval task	950
- # of *customized app* retrieval task	1,749
# size of tool	43,215
- # of *web API*	36,978
- # of *code function*	3,794
- # of *customized app*	2,443
avg. query / instruction length (tokens)	46.87 / 43.43
avg. tool documentation length (token)	174.56

Table 1: Basic statistics of our benchmark TOOLRET.

	Ours	NQ	MSMARCO	HotpotQA	MTEB
# Average number of targets for an input query.	2.17	1.00	1.00	2.00	2.57
# ROUGE-L overlap between query and targets.	0.06	0.31	0.34	0.11	0.27

Table 2: Comparison with conventional IR benchmarks.

to automate this process. Specifically, we first invite three human experts with strong NLP and IR backgrounds to manually craft 100 seed instructions. In line with the well-defined format from Asai et al., our instruction outlines the relevance criteria by bridging the query intent and the functionality of the target tools. For example, for the *transcribing the audio to text* task, the instruction is presented as "*retrieve tools that process audio inputs to produce accurate textual transcriptions aligned with the user requirements*". Next, we employ a powerful LLM, i.e., GPT-4o, as an automatic instruction generator and guide it to generate instruction for each task through in-context learning. To enhance the diversity, we randomly sample in-context examples from the pool of both the generated and handcrafted instructions. A detailed pseudo algorithm is provided in Appendix B.

After the above three processes, we obtain TOOLRET, which consists of 7.6k tasks, each paired with an instruction, and a corpus of 43k diverse tools, providing a comprehensive testbed and supporting various evaluation settings.

4 Benchmark statistic

Table 1 provides the basic statics of TOOLRET. We observe that there are three mainstream formats of tool documentation: (i) Code, which is a function-level snippet in programming language; (ii) Web API, which elaborates the tool usage in structured JSON format following the Web OpenAPI specification; (iii) Customized application, which directly describes the tool functionality in free-form nature language. Based on these formats, we categorize

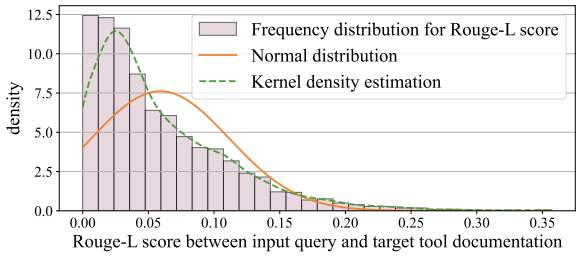

Figure 2: ROUGE-L overlap between the query (input) and the target tools (label).

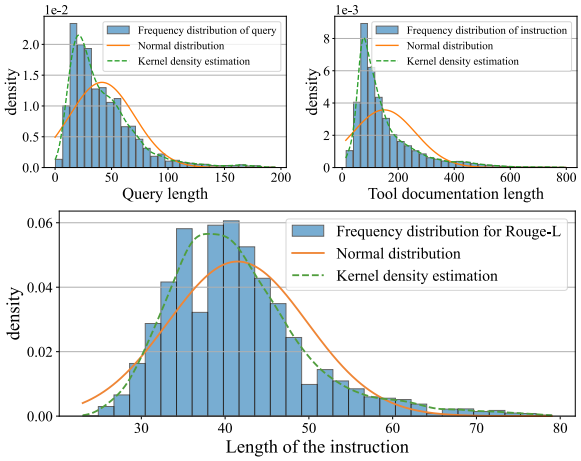

Figure 3: Length distribution of our benchmark.

TOOLRET into three subsets accordingly and divide the TOOLRET into **Code Function**, **Web API**, and **Customized App** subsets. Below, we report a more detailed analysis of TOOLRET.

4.1 Complexity

In tool learning, previous studies have highlighted the necessity of combining multiple tools for task solving (Shi et al., 2024). Thus, we analyze the complexity of our retrieval benchmark from two aspects. **First**, we calculate *the average number of target tools* for each retrieval task and compare it with well-known IR benchmarks such as HotpotQA (Yang et al., 2018) and MTEB (Muenni ghoff et al., 2022). As shown in Table 2, TOOLRET requires models to recall more targets, posing a challenge in comprehensive retrieval. **Second**, we compute the lexical overlap, i.e., ROUGE-L, between the input query and corresponding retrieval targets (tool documentation in TOOLRET and passage in IR benchmarks). We find that this overlap is substantially lower in TOOLRET. It indicates that, for neural IR models, TOOLRET requires more heavily on the semantic representation rather than simple lexical matching. Therefore, the retrieval task in TOOLRET is more challenging.

Quality Review Question	Yes or No %
Whether the instruction is relevant to the original input query?	90.1% / 9.9%
Whether the instruction describes the feature of target tools	92.3% / 8.7%
Whether the instruction comprehensively describe the feature of all target tools	89.2% / 10.8%
Whether the instruction contains hallucination about the target tools or input query?	5.9% / 94.1%

Table 3: The quality review for our generated instructions, which is conducted by five human experts with 0.743 *Kappa* statistics.

4.2 Length statistics

Figure 3 illustrates the length distribution of the query, instruction, and tool documentation in TOOLRET.[4] We find that most queries are concise, typically containing fewer than 60 tokens (about 25 words), which aligns with real-world user behavior, as users tend to input brief queries with minimal effort. Additionally, most tool documentation is under 200 tokens, which is similar to the chunk length in standard IR document retrieval corpus, such as Wikipedia dump (Karpukhin et al., 2020).

4.3 Quality

So far, we have demonstrated the complexity and quantity of our benchmark while the quality of the LLM-generated instructions remains uncertain. To investigate this, we ask 5 human experts to label the quality based on four aspects listed in Table 3. Our evaluation reveals that 89.2% of the generated instructions correctly cover the feature of the target tools and are faithfully grounded on the original queries. For the remaining 10.8% instructions that mismatch the query or the target tools, we ask experts to revise them. This re-check mechanism ensures the high quality of instructions in TOOLRET, making it a reliable evaluation benchmark. To explain more intuitively, we list a number of seed instructions, high-quality and low-quality instructions in Table 8. Annotation guidance is also provided in Appendix B to promote our transparency.

4.4 Instruction diversity

We further analyze how the generated instructions differ from the seed instructions used to prompt the generation. For each generated instruction, we compute its highest ROUGE-L overlap with the 100 seed instructions. We plot the distribution of these ROUGE-L scores in Figure 4. The results

[4]We use the tokenizer from gpt-3.5-turbo in this work.

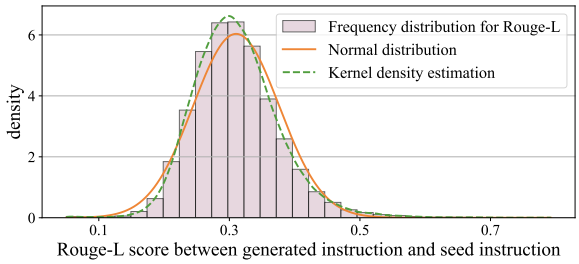

Figure 4: ROUGE-L overlap between the handcrafted seed instructions and model-generated instructions.

indicate a decent number of new instructions are generated, which have low overlap with the seeds.

5 Benchmark evaluation setup

5.1 Evaluation protocol

We use three widely used IR metrics to evaluate the retrieval performance: (i) *NDCG@K* (N@K): evaluates ranking quality based on the relevance of retrieved tools; (ii) *Recall@K* (R@K): evaluates the proportion of target tools successfully retrieved within the top-K results; and (iii) *Precision@K* (P@K): evaluates the accuracy of the retrieved tools within the top-K results. We also use *Completeness@K* (C@K) from COLT (Qu et al., 2024b), which specifically evaluates the retrieval completeness in tool retrieval tasks. The C@K is 1 if all target tools are included in the top-k retrieved tools; otherwise, it is 0.

We mainly evaluate IR models under two settings: (i) *w/o inst.*: The model take the query alone as input; and (ii) *w/ inst.*: The model takes the concatenation of the query and instruction as input to retrieve. This allows us to analyze the impact of instructions on retrieval performance.

5.2 Model to Evaluate

We comprehensively evaluate the following mainstream IR models on our benchmark.

Sparse retrieval. These methods measure the similarity between query and tool documentation based on lexical overlap. We evaluate BM25s (Lù, 2024).

Single-task dense retrieval. These methods use dual-encoder models trained on conventional IR datasets. We evaluate gtr (Ni et al., 2021a), contriever (Izacard et al., 2021a), and colbertv2.0 (Santhanam et al., 2021a), all trained on MS-MARCO (Nguyen et al., 2016). We also evaluate COLT (Qu et al., 2024a), a recently proposed model trained on ad-hoc tool retrieval datasets.

Multi-task embedding Models. These methods

Model	ToolRet-Web				ToolRet-Code				ToolRet-Customized				Average	
	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	C@10
BM25s	18.98	4.64	24.62	15.20	21.20	3.37	28.23	26.96	26.76	5.86	32.39	24.40	22.32	22.19
COLT	15.43	2.63	21.11	20.04	20.69	5.12	28.07	18.53	21.63	4.72	29.19	22.40	19.25	20.32
Colbert	22.40	5.37	27.41	15.45	16.43	2.65	22.54	21.65	19.54	3.65	23.72	18.97	19.46	18.69
contriever-msmarco	21.15	**5.83**	27.19	14.70	14.56	2.40	19.28	17.71	17.72	3.31	22.77	18.31	17.81	16.91
gtr-t5-base	17.36	4.25	24.17	15.95	16.47	2.71	22.27	21.16	23.47	5.09	28.93	22.49	19.10	19.87
gtr-t5-large	**22.45**	5.42	**29.75**	**18.72**	**18.25**	**2.89**	**24.12**	**23.08**	**26.30**	**5.76**	**31.86**	**24.45**	**22.34**	**22.09**
all-MiniLM-L6-v2	11.66	3.07	16.36	10.15	14.44	2.50	19.50	18.11	22.80	5.21	29.10	20.25	16.30	16.17
e5-small-v2	19.89	5.08	26.46	16.26	15.48	2.39	19.26	18.05	24.60	5.56	29.67	20.76	19.99	18.36
e5-base-v2	19.75	5.04	25.89	15.37	14.43	2.47	19.19	18.00	22.68	5.11	29.13	22.25	18.95	18.54
e5-large-v2	18.99	4.90	25.97	16.27	17.09	2.68	21.87	20.70	26.42	**6.07**	32.19	23.17	20.83	20.05
gte-base-en-v1.5	23.55	6.28	32.03	19.15	17.43	2.87	23.71	22.48	21.62	4.76	29.03	23.17	20.86	21.60
gte-large-en-v1.5	22.41	5.91	30.14	18.44	16.66	2.87	23.64	22.39	20.62	5.19	26.75	17.67	19.90	19.50
bge-base-en-v1.5	22.50	6.02	29.96	17.30	17.78	2.92	23.66	22.27	**25.99**	5.71	32.17	24.26	22.09	21.27
bge-large-en-v1.5	**24.45**	**6.66**	**32.93**	**19.30**	**18.90**	**3.12**	**25.76**	**24.47**	25.72	5.54	**32.18**	**24.79**	**23.02**	**22.85**
gte-Qwen2-1.5B-inst.♠	29.17	7.93	38.05	21.49	21.66	3.41	28.89	27.67	36.04	7.89	44.51	35.55	28.96	26.04
e5-mistral-7b♠	26.76	7.25	34.39	21.05	20.01	3.44	28.31	27.10	31.41	6.68	38.47	29.24	26.06	25.80
GritLM-7B♠	25.74	6.85	34.27	21.28	22.02	3.72	30.41	28.87	**42.31**	**8.71**	**49.34**	**38.17**	30.02	29.44
NV-Embed-v1♠	**31.30**	**8.35**	**39.15**	**23.05**	**29.64**	**4.72**	**40.45**	**38.88**	40.54	8.25	45.93	34.44	**33.83**	**32.12**
mxbai-rerank-large-v1	22.99	5.61	30.32	18.38	24.76	3.88	34.86	33.22	26.76	5.91	34.53	26.03	24.84	25.88
monot5-base-msmarco	28.92	7.70	36.44	19.97	21.61	3.62	30.06	27.88	36.22	7.54	45.11	36.41	28.92	28.09
bge-reranker-v2-m3	32.92	8.73	41.88	25.63	24.28	3.80	32.71	30.94	30.51	7.00	36.03	26.74	29.24	27.77
jina-reranker-v2-base	35.38	9.25	44.65	26.98	26.47	4.15	35.20	33.94	38.94	8.14	46.06	35.42	33.60	32.11
bge-reranker-v2-gemma	**36.72**	**9.69**	**45.94**	**27.85**	**29.89**	**4.42**	**38.23**	**36.82**	**39.93**	**9.06**	**49.43**	**37.75**	**35.51**	**34.14**
Mixtral-8x22B	28.21	**8.31**	34.13	**25.42**	27.41	3.14	34.13	36.98	30.76	5.40	34.12	28.65	28.80	30.35
gpt-3.5-turbo-0125	30.29	8.01	**36.00**	24.22	28.69	4.27	36.25	35.64	29.80	6.39	35.01	28.70	29.60	29.52
gpt-3.5-turbo-1106	**31.01**	7.86	35.82	23.76	**28.95**	**4.44**	**38.16**	**38.45**	**32.30**	**6.89**	**38.31**	**30.84**	**30.75**	**31.01**

Table 4: Experiment results in ***w/o inst.*** setting (§ 5), where the model takes the query as input to retrieve. We mark the baselines pre-trained on instructional datasets with ♠. We highlight the best performance in each type of model.

utilize transformer encoders trained on various IR datasets. We evaluate all-MiniLM-L6-v2, gte (Li et al., 2023c), bge (Xiao et al., 2023a) and e5 (Wang et al., 2022), covering a wide range of sizes.

Cross-encoder re-rankers. These models re-rank the initially retrieved documents based on the query-passage relevance. We evaluate: MonoT5 (Nogueira et al., 2020), mxbai-rerank-large, jina-reranker-v2-base, and bge-reranker.

LLM agents. These methods leverage general-purpose LLM agents for re-ranking tasks in a zero-shot setting, simulating the *tool selection* process of tool-use agents. We evaluate the widely used LLM re-ranking framework, i.e., RankGPT (Sun et al., 2023), with various LLMs as backbone.

Initial tools for *LLM agent* and *Re-ranking* baselines are retrieved by `Nv-embedd-v1` model. Details of these baselines are provided in Appendix D.

6 Experiment result

6.1 Tool retrieval performance

Existing retrievers struggle. As shown in Table 9, the tool retrieval tasks in ToolRet raise significant challenges for existing retriever models. Specifically, all retrievers in our experiments achieve less than 35% in Completeness@10 and under 52% in recall@10. Notably, retrieval methods that demonstrate strong performance in conventional information retrieval (IR) tasks, such as ColBERT, even underperform compared to simple lexical-based matching approaches like BM25. Similarly, other embedding-based models, even the NV-Embed-v1 with 7B parameter, achieve less than 45% in completeness@10, exhibiting limitations.

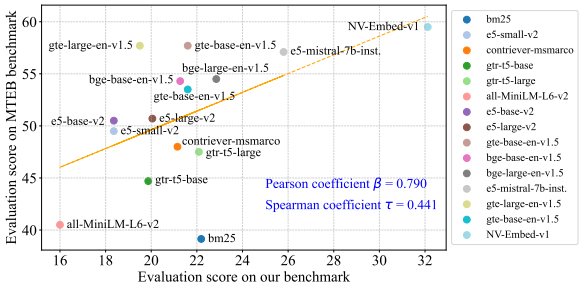

Figure 5: Correlation between the score on our benchmark and MTEB (retrieval subset).

We identify two potential reasons for the above performance degradation: (i) Tool retrieval tasks require intensive reasoning over the input query to align user intentions with candidate tools, as the lexical overlap between the query and targets is low. (ii) There exists a domain shift between the conventional training corpora used for retrieval models and the specific tool retrieval tasks, which current models are not explicitly optimized for.

Re-ranking technique has limited improvement. As shown in Table 9, commonly used re-ranking

Model	ToolRet-Web				ToolRet-Code				ToolRet-Customized				Average	
	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	C@10
BM25s	26.33	6.10	34.22	22.79	41.90	6.20	56.49	55.39	41.16	8.39	48.60	38.90	36.46	39.03
COLT	28.91	4.61	40.64	38.83	20.06	4.71	27.78	18.84	31.29	6.05	42.19	34.01	26.75	30.56
Colbert	16.67	3.12	21.14	14.94	30.35	4.37	41.38	40.28	24.35	4.56	30.97	24.87	23.79	26.70
contriever-msmarco	23.48	**5.29**	30.21	19.69	31.61	4.84	43.01	41.74	21.93	3.85	27.28	23.04	25.67	28.16
gtr-t5-base	20.38	4.49	27.53	19.24	33.59	4.90	43.18	41.88	41.84	7.66	48.35	39.28	31.94	33.46
gtr-t5-large	**24.37**	5.27	**31.64**	**21.26**	**36.76**	**5.33**	**47.42**	**45.92**	**42.04**	**8.48**	**50.84**	**40.00**	**34.39**	**35.73**
all-MiniLM-L6-v2	12.77	3.26	19.38	13.33	31.59	5.06	43.86	42.25	32.24	7.14	43.55	32.34	25.53	29.31
e5-small-v2	26.42	6.20	34.44	21.39	32.36	4.84	42.38	41.11	34.62	6.90	42.29	32.58	31.14	31.69
e5-base-v2	24.71	5.78	33.45	21.94	31.40	5.01	42.83	41.38	38.06	7.54	46.84	36.43	31.39	33.25
e5-large-v2	23.62	5.52	32.19	21.80	34.27	5.05	44.42	43.19	43.32	8.51	52.30	41.42	33.73	35.47
gte-base-en-v1.5	**30.75**	7.00	**39.44**	**25.88**	**41.68**	**6.20**	**53.96**	**51.64**	37.95	6.96	46.57	38.10	36.79	38.54
gte-large-en-v1.5	28.06	6.55	36.32	22.57	35.77	5.75	49.56	47.71	37.27	7.88	47.98	35.84	33.70	35.37
bge-base-en-v1.5	25.95	6.16	35.12	23.40	35.15	5.22	45.74	44.32	43.20	**8.82**	**53.54**	**42.29**	34.77	36.67
bge-large-en-v1.5	30.03	**7.01**	39.28	25.63	41.53	6.00	52.76	51.18	**43.90**	8.31	51.79	42.24	**38.49**	**39.68**
e5-mistral-7b♠	31.07	7.65	41.30	27.04	44.97	6.66	58.95	56.79	40.88	7.91	49.35	38.35	38.97	40.73
NV-Embed-v1♠	31.51	7.74	40.52	26.74	**47.92**	7.10	**62.07**	**59.60**	48.70	10.07	57.69	43.88	42.71	43.41
gte-Qwen2-1.5B-inst.♠	**37.53**	9.31	**48.31**	**30.95**	47.38	**7.29**	61.12	59.55	**52.98**	**10.63**	**59.47**	**45.68**	**45.96**	**45.39**
GritLM-7B♠	36.58	**9.34**	46.01	27.65	41.26	6.17	53.81	52.07	45.55	9.74	54.01	41.40	41.13	40.37
mxbai-rerank-large-v1	17.53	4.05	25.82	17.95	33.86	5.05	47.84	46.47	26.83	6.71	37.61	28.60	26.08	31.01
monoT5-base-msmarco	23.33	5.88	30.70	18.13	31.39	5.27	45.18	42.51	37.77	6.76	46.63	39.70	30.83	33.45
bge-reranker-v2-m3	34.83	8.54	45.23	31.73	50.86	7.64	67.26	64.78	42.35	9.52	53.75	39.90	42.68	45.47
jina-reranker-v2-base	**42.35**	**10.11**	**51.21**	**34.23**	53.21	7.66	66.03	63.94	45.94	10.36	57.96	45.41	47.17	47.86
bge-reranker-v2-gemma	34.73	8.09	45.08	32.29	**55.85**	**8.22**	**70.53**	**68.76**	**51.97**	**11.04**	**61.20**	**45.65**	**47.52**	**48.90**
Mixtral-8x22B	35.31	7.56	38.63	34.60	33.27	5.77	39.60	38.53	34.40	6.44	39.72	38.20	34.33	37.11
gpt-3.5-turbo-0125	37.22	8.97	40.82	35.22	35.42	6.22	41.16	42.64	37.29	**8.24**	41.34	**39.70**	29.60	29.52
gpt-3.5-turbo-1106	**38.31**	**9.02**	**41.29**	**35.76**	**38.69**	**7.27**	**42.57**	**42.81**	**39.30**	7.89	**43.31**	37.31	**38.77**	**38.63**

Table 5: Experiment results in ***w/ inst.*** setting (§ 5), where the model takes the query and instruction as input to retrieval. ♠ indicates the model is pre-trained on instructional datasets. We highlight the best performance.

methods provide limited and even negative improvements for the tool retrieval task. When using MonoT5 to re-rank the tools retrieved by NV-Embed-v1, the average NDCG@10 decreases from 33.83 to 28.92. A similar trend is observed with the mxbai-rerank. Other advanced models such as bge-ranker-v2-gemma only have 4.7% improvement in the Completeness@10 metric. These results further indicate the challenging nature of tool retrieval.

6.2 Substantial gains from instruction

Besides the evaluation results under *w/o inst* setting in Table 4, we also present the results under *w/ inst* setting in Table 5. We observe that all the IR model achieves better performance when an additional instruction is paired with the query as input. Notably, the instruction-tuned embedding model like `NV-embed-v1` or `e5-mistral` has the most obvious improvement, which potentially benefits from its powerful instruction-following capability. These results illustrate the advantages of the instruction and instruction tuning in tool retrieval tasks.

6.3 Compare with conventional IR tasks

To further explore the complexity of tool retrieval tasks, we compare the models' performance on ToolRet and conventional IR task benchmark, i.e., MTEB, showing their relationship in Figure 5. First, we can see that the two benchmarks share a similar trend (Pearson's coefficient $\beta = 0.790$), but the score in ToolRet is lower. This indicates that the task in ToolRet has a correlation with conventional IR tasks but is more challenging. Second, we also observe that conventional IR models trained with relevance-oriented criteria such as contriever perform poorly on ToolRet, which indicates that ToolRet requires more target-aware reasoning ability. This is also illustrated in § 4.1.

7 Retrieval affect downstream task

In this section, we qualitatively analyze the impact of retrieval performance on downstream tool-use agents. We conduct end-to-end evaluations on ToolBench (Qin et al., 2023) dataset using the official *Pass Rate* metric that evaluates whether the model successfully calls target tools to complete a task.

7.1 Poor retrieval leads to poor tool-use agents

For each task in ToolBench, we replace the officially pre-annotated toolset (oracle) with tools retrieved by IR models. As shown in Figure 6, the tool-use LLMs, when equipped with the retrieved tools, exhibit substantially lower performance compared to their oracle counterparts. For example, in ToolBench-G1, GPT-3.5 achieves a pass rate of 50.60 using tools retrieved by `bge-large`, decreasing by 11.40. This indicates that tool retrieval is a crucial step to build better tool-use LLMs and

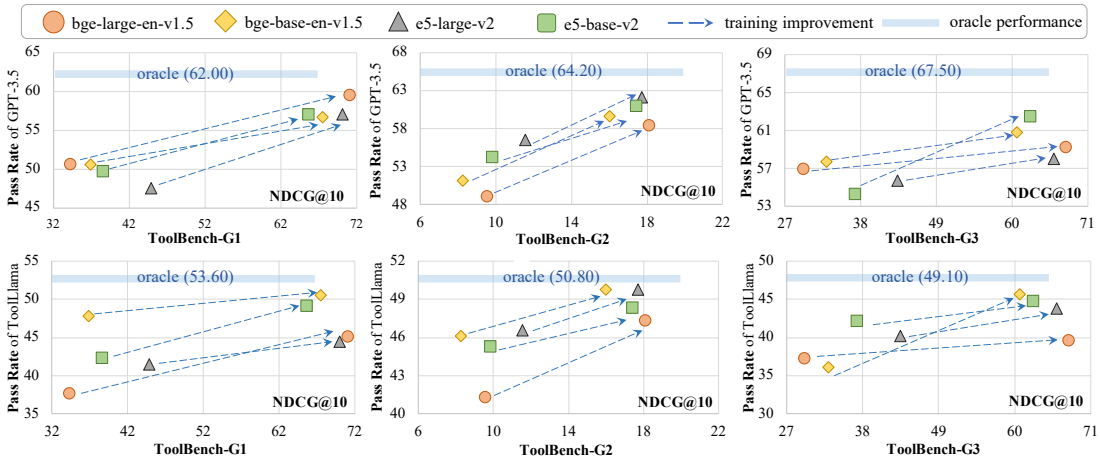

Figure 6: The **horizontal axis** indicates the retrieval performance of IR models, both before and after training. The **vertical axis** corresponds to the end-to-end pass rate of tool-use LLMs using the tools retrieved by these IR models.

improve their task-solving performance.

7.2 Towards better retrieval

The analysis in § 6.2 highlights the advantage of instruction-tuning in improving tool retrieval. However, to the best of our knowledge, there is no large-scale instructional IR dataset for tool retrieval tasks. We propose the ToolRet-train to fill this gap.

Large-scale training data We extend the data collection process from ToolRet to include the training sets of three mainstream tool-use datasets: ToolACE (Liu et al., 2024a), ToolBench (Qin et al., 2023) and APIGen (Liu et al., 2024b). Ultimately, we collect over 200k training instances, each comprising a query q and a set of target tools $\mathcal{T}$. We also pair each query q with an instruction $\mathcal{I}$ using our target-aware strategy (See Appendix D).

Learning objective To train a IR model (denoted as θ), we first use it to retrieved top-K negative tools, denoted as $\widehat{\mathcal{T}} = \{\hat{t}_j \mid j \in [K], \hat{t}_j \notin \mathcal{T}\}$. The model θ is then optimized by maximizing the *log-likelihood of the target tools*. The loss function $\mathcal{L}$ is formulated as:

$$-\frac{1}{|\mathcal{T}|}\sum_{t_i \in \mathcal{T}} \log \frac{e^{\text{sim}(\mathcal{I}\oplus q, t_i)}}{e^{\text{sim}(\mathcal{I}\oplus q, t_i)} + \sum_{\hat{t}_j \in \widehat{\mathcal{T}}} e^{\text{sim}(\mathcal{I}\oplus q, \hat{t}_j)}}.$$

The $\mathcal{I} \oplus q$ indicates concatenation of instruction and query with a special token. During the training, we set the K to 10 and the learning rate to 5e-5.

Improvement from retrieval. As shown in Figure 6, all IR models trained on ToolRet-train achieve substantial improvement in NDCG@10 metric. We further evaluate the task pass rate of two tool-use LLMs: GPT-3.5 and ToolLlama (Qin et al., 2023). When equipped with the improved IR models, both LLMs exhibit substantial gains in pass rate, confirming the critical role of retrieval in downstream tasks. As part of future work, we suggest adapting the IR models to better augment the tool-use LLMs, which offers a efficient plug-and-play solution compared with training LLMs.

We further conduct an ablation study by removing the instruction $\mathcal{I}$ from the loss function $\mathcal{L}$. The results show that this variant shows improvements compared with the non-tuned counterparts, but underperforms compared with their instruction-tuned counterparts (See Appendix D). These validate the effectiveness of our instructional training data in enhancing tool retrieval performance.

8 Conclusion

In this work, we introduce ToolRet, the first diverse tool retrieval benchmark comprising 7.6k queries, each paired with an instruction, and a corpus of 43k tools. ToolRet is a heterogeneous benchmark, constructed by aggregating existing tool-use datasets and aligning them into a unified format, similar to conventional IR benchmarks such as MTEB. We evaluate state-of-the-art IR models on ToolRet and uncover a surprising finding: even models with strong performance on conventional IR benchmarks struggle in tool retrieval. This low retrieval quality significantly degrades the end-to-end task pass rate of tool-use LLMs. Inspired by this, we further propose ToolRet-train, a large-scale training set containing over 200k retrieval tasks. Results show that IR models trained on ToolRet-train exhibit substantial improvement and also enhance the pass rate of tool-use LLMs by 10%-20%. In the future, we plan to extend the ToolRet into multimodal scenarios.

Limitation

The limitations of this work include the lack of exploration in multilingual retrieval settings. Currently, our benchmark is confined to the English language and focuses exclusively on text retrieval. To address this limitation, we plan to expand our research to encompass multilingual information retrieval (IR) scenarios in the future. Additionally, another limitation lies in the insufficient investigation of prompt sensitivity. Given that large language models (LLMs) are highly sensitive to prompt wording, we aim to annotate a broader range of instructions in the future to examine how variations in prompt phrasing influence LLM performance.

Building upon TOOLRET, we suggest the following directions for **future work**: (i) Investigating sensitivity to instructions: Conduct a comprehensive study on how LLM performance varies with different prompt formulations and instruction styles. (ii) Enhancing IR models for improved retrieval accuracy: Further optimize IR models to achieve higher retrieval precision, leveraging these improvements to augment tool-use LLMs and, consequently, enhance end-to-end task performance.

Ethics Statement

We acknowledge the importance of the ACM Code of Ethics and fully agree with it. We ensure that this work is compatible with the provided code in terms of publicly accessible datasets and models. Risks and harms associated with large language models include the generation of harmful, offensive, or biased content. The new benchmark is composed of various previous datasets and is therefore licensed under their respective data licenses.

In this research, we prioritize reproducibility by not only utilizing state-of-the-art commercial LLMs but also experimenting extensively with open-source LLMs. Throughout the study, we have strictly followed ethical standards to maintain the integrity and validity of our work. All tools and resources used in this research were obtained from publicly available platforms, ensuring transparency and reproducibility in our experimental procedures. Furthermore, we have made every effort to ensure that our research does not harm individuals or groups, nor does it involve any form of deception or potential misuse of information.

References

Anirudh Ajith, Mengzhou Xia, Alexis Chevalier, Tanya Goyal, Danqi Chen, and Tianyu Gao. 2024. Litsearch: A retrieval benchmark for scientific literature search. *arXiv preprint arXiv:2407.18940*.

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. 2022. Task-aware retrieval with instructions. *arXiv preprint arXiv:2211.09260*.

Emily M Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Si-An Chen, Lesly Miculicich, Julian Martin Eisenschlos, Zifeng Wang, Zilong Wang, Yanfei Chen, Yasuhisa Fujii, Hsuan-Tien Lin, Chen-Yu Lee, and Tomas Pfister. 2024b. Tablerag: Million-token table understanding with language models. *arXiv preprint arXiv:2410.04739*.

Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, et al. 2023. T-eval: Evaluating the tool utilization capability step by step. *arXiv preprint arXiv:2312.14033*.

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: AAAI*.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. 2021. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. *arXiv preprint arXiv:2403.07714*.

Shijue Huang, Wanjun Zhong, Jianqiao Lu, Qi Zhu, Jiahui Gao, Weiwen Liu, Yutai Hou, Xingshan Zeng, Yasheng Wang, Lifeng Shang, et al. 2024. Planning,

creation, usage: Benchmarking llms for comprehensive tool utilization in real-world complex scenarios. *arXiv preprint arXiv:2401.17167*.

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*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021a. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021b. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. 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 2020, Online, November 16-20, 2020*. Association for Computational Linguistics.

Tom Kwiat kowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. 2023a. Making large language models a better foundation for dense retrieval. *Preprint*, arXiv:2312.15503.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023b. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023c. Towards general text embeddings with multi-stage contrastive learning. *ArXiv*, abs/2308.03281.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, et al. 2024a. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, et al. 2024b. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *arXiv preprint arXiv:2406.18518*.

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 *Neural Information Processing Systems: NeurIPS*.

Xing Han Lù. 2024. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring. *ArXiv*, abs/2407.03618.

Zixian Ma, Weikai Huang, Jieyu Zhang, Tanmay Gupta, and Ranjay Krishna. 2024. m & m's: A benchmark to evaluate tool-use for m ulti-step m ulti-modal tasks. In *European Conference on Computer Vision*, pages 18–34. Springer.

Niklas Muenni ghoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. *Preprint*, arXiv:2402.09906.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773. CEUR-WS.org.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. 2021a. Large dual encoders are generalizable retrievers. *ArXiv*, abs/2112.07899.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. 2021b. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. *arXiv preprint arXiv:2003.06713*.

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, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2023. Toolink: Linking toolkit creation and using through chain-of-solving on open-source model. *arXiv preprint arXiv:2310.05155*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, et al. 2024. Tool learning with foundation models. *ACM Computing Surveys*, 57(4):1–40.

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*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Xu Jun, and Ji-Rong Wen. 2025a. From exploration to mastery: Enabling llms to master tools via self-driven interactions. In *The Thirteenth International Conference on Learning Representations*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2024a. Colt: Towards completeness-oriented tool retrieval for large language models. In *CIKM*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2024b. Towards completeness-oriented tool retrieval for large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 1930–1940.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025b. Tool learning with large language models: A survey. *Frontiers of Computer Science*.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J Maddison, and Tatsunori Hashimoto. 2023. Identifying the risks of lm agents with an lm-emulated sandbox. *arXiv preprint arXiv:2309.15817*.

Keshav Santhanam, O. Khattab, Jon Saad-Falcon, Christopher Potts, and Matei A. Zaharia. 2021a. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *North American Chapter of the Association for Computational Linguistics*.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021b. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488*.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. 2023. Taskbench: Benchmarking large language models for task automation. *arXiv preprint arXiv:2311.18760*.

Zhengliang Shi, Shen Gao, Xiuyi Chen, Yue Feng, Lingyong Yan, Haibo Shi, Dawei Yin, Zhumin Chen, Suzan Verberne, and Zhaochun Ren. 2024. Chain of tools: Large language model is an automatic multi-tool learner. *arXiv preprint arXiv:2405.16533*.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, et al. 2023. Restgpt: Connecting large language models with real-world restful apis. *arXiv preprint arXiv:2306.06624*.

Weiwei Sun, Zhengliang Shi, Jiulong Wu, Lingyong Yan, Xinyu Ma, Yiding Liu, Min Cao, Dawei Yin, and Zhaochun Ren. 2024. Mair: A massive benchmark for evaluating instructed retrieval. In *EMNLP*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. *arXiv preprint arXiv:2304.09542*.

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

Hongru Wang, Rui Wang, Boyang Xue, Heming Xia, Jingtao Cao, Zeming Liu, Jeff Z Pan, and Kam-Fai Wong. 2024a. Appbench: Planning of multiple apis from various apps for complex user instruction. *arXiv preprint arXiv:2410.19743*.

Jize Wang, Zerun Ma, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. 2024b. Gta: a benchmark for general tool agents. *arXiv preprint arXiv:2407.08713*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. 2024c. Toolgen: Unified tool retrieval and calling via generation. *arXiv preprint arXiv:2410.03439*.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024d. Executable code actions elicit better llm agents. *arXiv preprint arXiv:2402.01030*.

Zhiruo Wang, Zhoujun Cheng, Hao Zhu, Daniel Fried, and Graham Neubig. 2024e. What are tools anyway? a survey from the language model perspective. *arXiv preprint arXiv:2403.15452*.

Orion Weller, Benjamin Chang, Sean MacAvaney, Kyle Lo, Arman Cohan, Benjamin Van Durme, Dawn Lawrie, and Luca Soldaini. 2024. Followir: Evaluating and teaching information retrieval models to follow instructions. *arXiv preprint arXiv:2403.15246*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023a. C-pack: Packaged resources to advance general chinese embedding. *ArXiv*, abs/2309.07597.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023b. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Qiancheng Xu, Yongqi Li, Heming Xia, and Wenjie Li. 2024. Enhancing tool retrieval with iterative feedback from large language models. *arXiv preprint arXiv:2406.17465*.

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*.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2024. Gpt4tools: Teaching large language model to use tools via self-instruction. *Advances in Neural Information Processing Systems*, 36.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Junjie Ye, Guanyu Li, Songyang Gao, Caishuang Huang, Yilong Wu, Sixian Li, Xiaoran Fan, Shihan Dou, Qi Zhang, Tao Gui, et al. 2024a. Tooleyes: Fine-grained evaluation for tool learning capabilities of large language models in real-world scenarios. *arXiv preprint arXiv:2401.00741*.

Junjie Ye, Yilong Wu, Songyang Gao, Caishuang Huang, Sixian Li, Guanyu Li, Xiaoran Fan, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024b. Rotbench: a multi-level benchmark for evaluating the robustness of large language models in tool learning. *arXiv preprint arXiv:2401.08326*.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R Fung, Hao Peng, and Heng Ji. 2023. Craft: Customizing llms by creating and retrieving from specialized toolsets. *arXiv preprint arXiv:2309.17428*.

Shuo Zhang and Krisztian Balog. 2020. Web table extraction, retrieval, and augmentation: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(2):1–35.

Yinger Zhang, Hui Cai, Xeirui Song, Yicheng Chen, Rui Sun, and Jing Zheng. 2023. Reverse chain: A generic-rule for llms to master multi-api planning. *arXiv preprint arXiv:2310.04474*.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. 2024. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv preprint arXiv:2406.15877*.

A Data Card

Following previous work (Bender and Friedman, 2018; Gebru et al., 2021; Zhuo et al., 2024), we provide the datacard for TOOLRET, where we tend to summarize and centralize all information that might be relevant for the benchmark analysis.

(i) *The purpose of this benchmark*: This benchmark is proposed to comprehensively evaluate the information retrieval (IR) models on tool retrieval tasks. On top of TOOLRET, we find that existing IR models, despite achieving strong performance in conventional IR benchmarks such as MTEB and BEIR, still suffer from substantial challenges in tool retrieval tasks. The poor retrieval quality further degrades the end-to-end task pass rate of tool-use LLMs. Thus, we believe that the TOOLRET reveals the importance of tool retrieval in building better tool-use LLMs, and can be used as a comprehensive and fair benchmark in facilitating the development of tool retrieval models.

(ii) *How will the dataset be distributed (e.g., Tarball on Website or Github)?* The proposed benchmark TOOLRET will be released to the public, and hosted on GitHub and Hugging Face. The TOOLRET will be managed and maintained by our research team.

(iii) *Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)?* Yes. If we include more tasks or find any errors, we will correct the dataset hosted on Hugging Face and GitHub and update the results in the leaderboard accordingly. It will be updated on our website.

(iv) *Will the training dataset* TOOLRET*-train will be released publicly.* Yes, the proposed training dataset TOOLRET-train will be released to the public, and hosted on GitHub and Hugging Face.

B Details of Benchmark

B.1 Dataset collections

TOOLRET is a heterogeneous benchmark that integrates a wide range of well-established tool-use datasets and aligns them into a unified format, similar to standard information retrieval (IR) benchmarks such as **BEIR** and **MTEB**, to facilitate tool retrieval evaluation. In tool learning, we observe that previous work primarily focuses on three mainstream types of tools:

(i) *Web APIs*: These tools are encapsulated in the OpenAPI format (standard JSON documentation) and can be directly invoked via HTTP requests. Web APIs are typically used to access, manipulate (e.g., add, delete, edit, or query), or retrieve private data or information from specialized databases, covering a wide range of domains such as movies, music, and sports.

(ii) *Code Functions*: These tools are represented by source code containing function signatures and implementation details. Code functions primarily focus on low-level computations or atomic operations, such as tensor calculations, calling Hugging Face models, or utilizing PyTorch libraries.

(iii) *Customized Apps*: These tools are paired with free-form natural language descriptions. They are typically user-oriented or personalized, enabling tasks such as sending emails or other custom applications.

These tool types differ in functionality and documentation format, reflecting diverse scenarios for tool-use LLMs. For IR models, retrieving different types of tools may present varying levels of difficulty. Therefore, we categorize the collected datasets into these three types based on their paired toolset formats, resulting in three subsets of TOOLRET: TOOLRET-web, TOOLRET-code, and TOOLRET-customized. During evaluation, we report the performance of IR models on each subset to provide a fine-grained analysis. Below, we list the datasets included in each subset and provide detailed descriptions.

B.2 TOOLRET-Web

The TOOLRET-Web subset is constructed by integrating the following datasets, which contain tools in the form of Web APIs:

- **AutoTools-Food** (Shi et al., 2024): Contains APIs related to food recipes, where LLMs must retrieve specific food-related tools to answer user queries.
- **RestGPT-TMDB** (Song et al., 2023) and **AutoTools-Movie** (Shi et al., 2024): Includes web APIs from the **TMDB** platform, a movie database. Evaluation tasks require LLMs to retrieve tools to find relevant information about movies or celebrities and extract key evidence to answer given queries.
- **AutoTools-Weather** (Shi et al., 2024): Features web APIs from a weather database. LLMs must invoke these APIs and gather responses to answer weather-related queries.
- **RestGPT-Spotify** and **AutoTools-Music** (Shi et al., 2024): Contains web APIs from a music platform. Evaluation tasks require LLMs to retrieve tools for searching songs or albums based on user queries.
- **ToolBench** (Qin et al., 2023): Comprises over **16,000** web APIs crawled from **RapidAPI**. Queries are generated by LLMs, with ground truth tools labeled for each query.
- **ToolLens** (Qu et al., 2024a): A subset of **ToolBench**, where queries are annotated to evaluate tool functionality.
- **APIbank** (Li et al., 2023b): Contains web APIs for daily personalized applications, such as alarm booking and database login.
- **MetaTool (ToolE)** (Huang et al., 2023): Designed to evaluate whether LLMs are aware of tool usage and can correctly select tools.
- **Mnms** (Ma et al., 2024): Evaluates LLM-based agents' tool-use abilities for **multi-step, multi-modal** tasks involving tools that process visual information. Since TOOLRET focuses on text-based IR models, images are represented using their URLs.
- **Reverse-Chain** (Zhang et al., 2023): Contains diverse **multi-step tasks** requiring LLMs to invoke relevant tools sequentially.
- **ToolEyes** (Ye et al., 2024a): Includes tools across various domains, such as **advice, entertainment, and art**, providing a broad evaluation of tool-use LLMs in practical scenarios.
- **UltraTool** (Huang et al., 2024): A benchmark designed to improve and evaluate LLMs' tool utilization abilities in **real-world scenarios**, focusing on the entire process of planning, creating, and applying tools in complex tasks.
- **T-Eval** (Chen et al., 2023): A fine-grained benchmark assessing tool-use LLMs across multiple evaluation aspects, including **instruction following, planning, reasoning, retrieval, understanding, and review**.

B.3 TOOLRET-Code

The TOOLRET-code subset is constructed by integrating the following datasets, which contain tools in the form of code functions:

- **Gorilla-PyTorch** (Patil et al., 2023): Contains various **PyTorch functions** (code snippets) as tools, evaluating LLMs' ability to correctly combine PyTorch functions for solving deep learning tasks. The functions in this dataset are collected from the **Python Torch** package.

Dataset	Endpoint	Query size	Tool size	Task type
Web APIs				
GTA (Wang et al., 2024b)	https://github.com/open-compass/GTA	14	14	
Gorilla (Patil et al., 2023)	https://github.com/ShishirPatil/gorilla	598	1,005	
- *gorilla-pytorch* subset	https://github.com/ShishirPatil/gorilla/tree/main/data	43	43	
- *gorilla-tensor* subset	https://github.com/ShishirPatil/gorilla/tree/main/data	55	55	
- *gorilla-huggingface* subset	https://github.com/ShishirPatil/gorilla/tree/main/data	500	907	
CRAFT (Yuan et al., 2023)	https://github.com/lifan-yuan/CRAFT	654	985	
- *craft-Tabmwp* subset	https://github.com/lifan-yuan/CRAFT/tree/main/tab_and_math/TabMWP	174	180	
- *craft-Vqa* subset	https://github.com/lifan-yuan/CRAFT/tree/main/vqa	200	525	
- *craft-algebra* subset	https://github.com/lifan-yuan/CRAFT/tree/main/tab_and_math/MATH	280	280	
AutoTools (Shi et al., 2024)	https://github.com/mangopy/AutoTools	159	159	
- *AutoTools-Food* subset	https://github.com/mangopy/AutoTools/tree/main/data	22	22	
- *AutoTools-Weather* subset	https://github.com/mangopy/AutoTools/tree/main/data	11	11	
- *AutoTools-Movie* subset	https://github.com/mangopy/AutoTools/tree/main/data	54	54	
- *AutoTools-music* subset	https://github.com/mangopy/AutoTools/tree/main/data	72	72	
APIGen (Liu et al., 2024b)	https://huggingface.co/datasets/Salesforce/xlam-function-calling-60k	1,000	3,605	
APIbank (Li et al., 2023b)	https://github.com/AlibabaResearch/DAMO-ConvAI	101	101	
Appbench (Wang et al., 2024a)	https://github.com/ruleGreen/AppBench	32	32	
Mms (Ma et al., 2024)	https://github.com/RAIVNLab/mnms	33	33	
Metatool (*a.k.a.*, ToolE) (Huang et al., 2023)	https://github.com/HowieHwong/MetaTool	200	200	
Reverse Chain (Zhang et al., 2023)	https://github.com/ASK-03/Reverse-Chain	200	783	
RestGPT (Song et al., 2023)	https://github.com/Yifan-Song793/RestGPT	94	94	
- *RestGPT-TMDB* subset	https://github.com/Yifan-Song793/RestGPT/datasets/tmdb.json	54	54	
- *RestGPT-Spotify* subset	https://github.com/Yifan-Song793/RestGPT/datasets/spotify.json	40	40	
Toolbench (Qin et al., 2023)	https://github.com/OpenBMB/ToolBench	1,100	16,000	
- *G1-instruction* subset	https://drive.google.com/drive/folders/1yBUQ732mPu-KclJnuQELEhtKakdXFc3J	200	16,072	
- *G1-Tool* subset	https://drive.google.com/drive/folders/1yBUQ732mPu-KclJnuQELEhtKakdXFc3J	200	16,072	
- *G1-category* subset	https://drive.google.com/drive/folders/1yBUQ732mPu-KclJnuQELEhtKakdXFc3J	200	16,072	
- *G2-instruction* subset	https://drive.google.com/drive/folders/1yBUQ732mPu-KclJnuQELEhtKakdXFc3J	200	16,072	
- *G2-instruction* subset	https://drive.google.com/drive/folders/1yBUQ732mPu-KclJnuQELEhtKakdXFc3J	200	16,072	
- *G3-instruction* subset	https://drive.google.com/drive/folders/1yBUQ732mPu-KclJnuQELEhtKakdXFc3J	100	16,072	
ToolLens (Qu et al., 2024a)	https://github.com/quchangle1/COLT	314	314	
Tooleyes (Ye et al., 2024a)	https://github.com/Junjie-Ye/ToolEyes	95	95	
ToolACE (Liu et al., 2024a)	https://huggingface.co/datasets/Team-ACE/ToolACE/	1,000	16,072	
GPT4tools (Yang et al., 2024)	https://github.com/AILab-CVC/GPT4Tools	32	32	
Rotbench (Ye et al., 2024b)	https://github.com/Junjie-Ye/RoTBench	550	919	
T-eval (Chen et al., 2023)	https://github.com/open-compass/T-Eval	100	100	
- *T-eval-step level* subset	https://huggingface.co/datasets/lovesnowbest/T-Eval	50	50	
- *T-eval-dialogue level* subset	https://huggingface.co/datasets/lovesnowbest/T-Eval	50	50	
Taskbench (Shen et al., 2023)	https://github.com/microsoft/JARVIS	103	103	
- *TaskBench-multimedia* subset	https://github.com/microsoft/JARVIS/taskbench/multimedia	40	40	
- *TaskBench-daily* subset	https://github.com/microsoft/JARVIS/taskbench/dailylifeapis	40	40	
- *TaskBench-DL* subset	https://github.com/microsoft/JARVIS/taskbench/huggingface	23	23	
ToolAlpaca (Tang et al., 2023)	https://github.com/tangqiaoyu/ToolAlpaca	94	1,937	
Toolbench-sam (Xu et al., 2023)	https://github.com/sambanova/toolbench	197	197	
ToolEmu (Ruan et al., 2023)	https://github.com/ryoungj/ToolEmu	38	38	
TooLink (Qian et al., 2023)	https://github.com/qiancheng0/Toolink	497	1,804	
UltraTool (Huang et al., 2024)	https://github.com/JoeYing1019/UltraTool	500	1,171	

Table 6: The detailed statistics about the each collected dataset in TOOLRET. We highlight that the subsets of ToolBench share a same toolsets containing 16,000+ tools. Besides, the TOOLRET combine and deduplicate the toolsets from the above datasets to build the final tool retrieval corpus.

- **Gorilla-Tensor** (Patil et al., 2023): Includes **TensorFlow functions** as tools, collected from **TensorFlow Hub**, to assess LLMs' tool selection capabilities in deep learning scenarios.
- **Gorilla-HuggingFace** (Patil et al., 2023): Treats specific **downstream models** from the **Hugging Face** platform as tools. This dataset evaluates LLMs' performance in correctly calling Hugging Face models based on user queries.
- **CRAFT-TabMWP** (Yuan et al., 2023): Evaluates LLMs' ability to use **functions for table processing**. The functions in this dataset are first generated by **GPT-4** and subsequently verified.
- **CRAFT-VQA** (Yuan et al., 2023): Provides evaluation cases for **visual question answering (VQA)**, where LLMs must call image processing functions such as **image capture** and **object detection**.
- **CRAFT-Math-Algebra** (Yuan et al., 2023): Assesses LLMs' ability to invoke **algebra functions** for solving complex mathematical problems.

B.4 TOOLRET-Customized

Besides Web APIs and code functions, we also collect datasets that contain customized apps. Unlike Web APIs and code functions, customized apps are described using free-form natural language documentation

Algorithm 1: The pseudo algorithm for our target-aware strategy in automatically constructing instructions for evaluation tasks.

Input: A set of N seed instructions $S = \{s_i \mid i \in [N]\}$ manually crafted by human experts; A powerful LLM $\mathcal{M}$ (e.g., GPT-4o); Collected tasks $\mathcal{T} = \{t_i | i \in [|\mathcal{T}|]\}$

Initialize an instruction pool $I \leftarrow S$;

for $i \in |\mathcal{T}|$ **do**

  Sample k examples $\{s'_1, s'_2, ..., s'_k\}$ from I;

  `// Generate a new instruction` s_i `using` $\mathcal{M}$ `through in-context learning:`

  $s_i \leftarrow \mathcal{M}(\text{prompt with } \{s'_1, s'_2, ..., s'_k\})$;

  `//Append new instruction to pool:`

  $\mathcal{I} = \mathcal{I} \cup \{s_i\}$;

Apply heuristic filtering to remove low-quality instructions from I;

Output: A set of high-quality instructions $\mathcal{I} = \{s_1, s_2, ..., s_{|\mathcal{T}|}\}$

rather than structured formats. Specifically, we include the following datasets: **ToolACE** (Liu et al., 2024a), **GPT4Tools** (Yang et al., 2024), **TaskBench** (Shen et al., 2023), **ToolAlpaca**, **ToolBench-sam** (Xu et al., 2023), **ToolEmu** (Ruan et al., 2023), and **TooLink** (Qian et al., 2023).

B.5 Task format

The final benchmark, TOOLRET, integrates the above datasets and reformats all test cases into a unified format, similar to conventional IR benchmarks such as **BEIR** and **MTEB**, to evaluate IR models in tool retrieval tasks. Each reformatted task consists of: an input query, an instruction, and the corresponding target tools (e.g., labels). Each tool is assigned a unique identifier and is paired with detailed documentation describing its functionality. Below, we present a concrete example from TOOLRET.

```
# An example of an evaluation task in our proposed benchmark

- Query: I need to find a grocery store near 123 Main Street, Downtown District that
 has a good selection of limes for my Easter celebration.
- ID: toolLens_query_7
- Target tools (labels): toolLens_tool_20, toolLens_tool_50, toolLens_tool_2
- Instruction: Given a `local grocery search` task, retrieve tools that can locate
grocery stores based on the user's specified location and criteria, such as the
availability of specific items like limes, to meet the query's requirements.


# Examples of tool documentation

- toolLens_tool_20: {"category_name": "Food", "required_parameters": [{"name": "
ingredient", "type": "STRING", "description": "", "default": "strawberry"}], "
optional_parameters": [], "method": "GET", "template_response": {"name": "str", "
ingredients": ["list of str with length 9"], "instructions": ["list of str with
length 7"]}, "name": "pastry/ingredient", "description": "This API endpoint allows
users to retrieve a random pastry recipe that contains a specific ingredient. Users
can make a GET request to the endpoint with the name of the ingredient as a query
parameter, and the API will return a JSON response with the given recipe, including
the name, list of ingredients, and instructions."}

- toolLens_tool_50: {"category_name": "Health_and_Fitness", "required_parameters":
[], "optional_parameters": [{"name": "limit", "type": "NUMBER", "description": "
limit the length of response", "default": "10"}], "method": "GET", "
template_response": {"count": "int", "food": [{"_id": "str", "food_name": "str", "
quantity": "str", "calories": "int", "uri": "str", "type": "str", "type_uri": "str",
 "core": "str", "core_uri": "str", "food_nutrition": [{"nutrient_name": "str", "
value": "float", "unit": "str", "_list_length": 3}], "_list_length": 10}]}, "name":
"View All Food Items", "description": "The request allows clients to retrieve a
comprehensive list of all available food items.\n\nAPI request sent to [https://
indnutrientsapi.tech/food](https://indnutrientsapi.tech/food)"}

- toolLens_tool_2: {"category_name": "Food", "required_parameters": [{"name": "
```

grocery", "type": "string", "description": "", "default": ""}], "optional_parameters": [], "method": "GET", "template_response": {"message": "str"}, "name": "Search a Grocery", "description": "Search a specific grocery"}

B.6 Details of instruction construction

In TOOLRET, each task is paired with an instruction using a target-aware strategy, where GPT-4o acts as an automatic expert through in-context learning. Specifically, we follow these steps. We first invite three human experts with strong backgrounds in NLP and IR to manually craft seed instructions. These expert-crafted instructions form an initial example pool. For each task, we randomly sample a set of instructions from this pool as in-context learning examples. Using these examples, GPT-4o generates a new instruction tailored to the given task. The newly generated instruction is then appended back to the instruction pool to enhance instruction diversity. The detailed pseudo algorithm is provided in Alg. 1.

Below, we provide a concrete example of the GPT-4o prompt used in our instruction construction process. The example of seed instructions and the generated instructions is provided in Table 8.

```
# The prompt for GPT-4o.

Given a query, you need to design an instruction about 20 words that clearly
indicates this is a task to retrieve tools capable of solving the query based on its
 content. The instruction should emphasize the task requirements and target outcomes
 of the query while incorporating the functional characteristics of the tools to
help the system accurately match the appropriate tools.

Below, I have provided the target tools (i.e., the labels for the query). Please
analyze the key aspects of the query and the tool descriptions. Your instruction
should implicitly highlight the task requirements and the characteristics of the
target tools relevant to the query.

Here is an output template that your should follow. Please note that the instruction
 should be concise.

Query: I would like to generate a video presenting a text-based discussion on the
topic of 'The Benefits of Exercise'
Labels: [1] {'id': 'taskbench_data_huggingface_tool_5', 'doc': {'input-type': ['text
'], 'output-type': ['text'], 'name': 'Text Generation', 'description': 'Generating
text is the task of producing new text. These models can, for example, fill in
incomplete text or paraphrase.'}}
Instruction: Given a `text-to-video` task, retrieve tools that process text inputs
to generate coherent textual outputs aligned with the query's topic and requirements
.

Query: I have an audio file 'example.wav' which is difficult to understand. I would
like you to help me transcribe the audio to text
Labels: [1] {'id': 'taskbench_data_huggingface_tool_19', 'doc': {'input-type': ['
audio'], 'output-type': ['text'], 'name': 'Automatic Speech Recognition', '
description': 'Automatic Speech Recognition (ASR), also known as Speech to Text (STT
), is the task of transcribing a given audio to text. It has many applications, such
 as voice user interfaces.'}, 'relevance': 1}
Instruction: Given a `audio transcription` task, retrieve tools that process audio
inputs to produce accurate textual transcriptions aligned with the query's
requirements.

Query: Conduct a two-sample independent t-test with two samples, sample1=[1, 2, 3,
4, 5] and sample2=[6, 7, 8, 9, 10], and a significance level of 0.05.
Labels: [1] {'id': 'tool_id_693', 'doc': {'name': 'independent_samples_t_test', '
description': 'Conducts a two-sample independent t-test and returns the t-statistic,
 p-value, and conclusion.', 'parameters': {'sample1': {'description': 'The first
sample of observations.', 'type': 'List[float]', 'default': 0.05}, 'sample2': {'
description': 'The second sample of observations.', 'type': 'List[float]', 'default
': 0.05}, 'alpha': {'description': 'The significance level of the test. Defaults to
0.05.', 'type': 'float, optional'}}}, 'relevance': 1}
Instruction: Given a `significance test` task, retrieve tools that perform
statistical tests, specifically a two-sample independent t-test, by processing
numerical inputs and returning the t-statistic, p-value.
```

```
Query: Can I get a list of all boards and their attributes on page number two with a
 page size of seven?
Labels: [1] {'id': 'ToolEyes_tool_34', 'doc': {'name': 'get_boards', 'description':
'A list of all boards and their attributes.', 'parameters': {'type': 'object', '
properties': {'page': {'type': 'string', 'description': 'Get the items on a specific
 page. 0(default) is the first page.'}, 'page_size': {'type': 'string', 'description
': 'Get the number of boards on a specific page. Default: 5.'}}, 'required': []}}}
Instruction: Given a `pagination query` task, retrieve tools that can list boards
and their attributes by processing parameters such as page number and page size to
return the requested information.
```

B.7 Human annotation

To ensure the quality of the generated instructions, we conduct human annotation to review them based on four key aspects listed in Table 3. For instructions deemed low-quality, human annotators manually revise them to improve accuracy and clarity. For a clear illustration, we provide concrete examples of handcrafted instructions, high-quality generated instructions, and low-quality generated instructions in Table 8. Below, we also provide the detailed human annotation guidelines used in our review process for reproducibility and transparency.

```
# Human guidance for instruction quality annotation

We ask you to evaluate the quality of the generated instructions based on the
following four aspects. Please carefully assess each instruction and provide your
judgment:

## Aspects for annotation

1. Hallucination Check: Does the instruction contain any incorrect or fabricated
information about the target tools or the input query?  (Are there any details in
the instruction that do not align with the actual features of the target tools or
the content of the input query?)

2. Comprehensiveness of Tool Features: Does the instruction fully and accurately
describe the features of all target tools mentioned in the query? (Are there any
important features of the target tools that are missing or inadequately described in
 the instruction?)

3. Accuracy of Tool Feature Description: Does the instruction correctly describe the
 features of the target tools? (Key question to ask: Are the descriptions of the
target tools technically accurate and consistent with their actual functionality?)

4. Relevance to Input Query: Is the instruction directly relevant to the original
input query? (Key question to ask: Does the instruction address the specific needs
or context provided in the input query, or does it deviate from the query's intent?)

## Detailed annotation Process

For each instruction, evaluate it based on the four aspects above.
1. If the instruction meets all criteria (no hallucination, comprehensive, accurate,
 and relevant), mark it as correct.
2. If the instruction fails to meet any of the criteria, mark it as incorrect and
provide a brief explanation of the issue (e.g., "contains hallucination," "missing
key tool features," or "irrelevant to query").

For incorrect instructions, ``revise`` them to ensure they meet all quality criteria
. The goal of this annotation process is to ensure that all instructions in our
benchmark are of high quality and faithfully grounded in the original queries and
target tools.
```

C Large-scaling training dataset: ToolRet-train

We extend the data collection process from ToolRet to incorporate the training sets of three mainstream tool-use datasets: ToolACE (Liu et al., 2024a), ToolBench (Qin et al., 2023), and APIGen (Liu et al.,

2024b). These datasets are selected for their diversity in task types, tool categories, and query complexity, ensuring a comprehensive representation of real-world tool-use scenarios.

After collecting and preprocessing the data, we ultimately collect over 200k training instances. Each instance consists of a query and a corresponding set of target tools. Next, we further pair each query with an instruction using our target-aware strategy (§ 3.3). This strategy generates instructions that explicitly guide the retrieval process by incorporating task-specific context and tool functionality descriptions. We report the basic statistics of TOOLRET-train in Table 7. We also show a training example of our TOOLRET-train.

```
# Query: Is 'https://www.apple.com' available in the Wayback Machine on September 9,
 2015?

# Instruction: Given a `URL availability` task, retrieve tools that check if a given
 URL is archived and accessible on a specific date in the Wayback Machine.

# Target tools (labels): ['{'name': 'availability', 'description': 'Checks if a
given URL is archived and currently accessible in the Wayback Machine.', 'parameters
': {'url': {'description': 'The URL to check for availability in the Wayback Machine
.', 'type': 'str', 'de...}}}']

# Negative tools: [
    {'name': 'top_grossing_mac_apps', 'description': 'Fetches a list of the top-
    grossing Mac apps from the App Store.', 'parameters': {'category': {'description
    ': "The category ID for the apps to be fetched. Defaults to '6016' (general
    category).", 'type': 'str', 'default': '6016'}, 'country': {'descript...},
    {'name': 'top_paid_mac_apps', 'description': 'Retrieves a list of the top paid
    Mac  apps from the App Store.', 'parameters': {'category': {'description'...},
    ...
    {'name': 'exact_url_non_english', 'description': 'Retrieves the backlinks of a
    specific non-English URL using the RapidAPI service...}
]
```

Statistic	
# size of retrieval task	205,826
# Average token length of the input query	52.87
# Average token length of the paired iinstruction	46.72
# Average token length of the tool documentation	163.52
# Number of negative tools per input query	5
# Number of target tools (labels) per input query	2.31

Table 7: Basic statistics of the collected large-scaling training set TOOLRET-train. We use the tokenizer from gpt-3.5-turbo in this work.

D More experiment details

D.1 Baselines

We comprehensively evaluate the following mainstream retrieval models on our benchmark, including:

- **Sparse Retrieval**. These methods measure the similarity between tasks and tool documentation based on lexical overlap. We evaluate BM25s (Lù, 2024).

- **Single-task dense retrieval**. These methods employ dual-encoder architecture models trained on conventional IR datasets. We evaluate gtr (Ni et al., 2021a), contriever (Izacard et al., 2021a), and colbertv2.0 (Santhanam et al., 2021a), all trained on MS-MARCO (Nguyen et al., 2016) with relevance criteria. We also evaluate the COLT (Qu et al., 2024a) which is a recently proposed model trained on an ad-hoc tool retrieval dataset.

Example of our seed instructions (handcrafted instruction
Query: I would like to generate a video presenting a text-based discussion on the topic of 'The Benefits of Exercise'. # Instruction: Given a "text-to-video" task, retrieve tools that process text inputs to generate coherent textual outputs aligned with the query's topic and requirements.
Query: I have an audio file 'example.wav' which is difficult to understand. I would like you to help me transcribe the audio to text. # Instruction: Given a "audio transcription" task, retrieve tools that process audio inputs to produce accurate textual transcriptions aligned with the query's requirements.
Query: Conduct a two-sample independent t-test with two samples, sample1=[1, 2, 3, 4, 5] and sample2=[6, 7, 8, 9, 10], and a significance level of 0.05. # Instruction: Given a "significance test" task, retrieve tools that perform statistical tests, specifically a two-sample independent t-test, by processing numerical inputs and returning the t-statistic, p-value.
Query: Can you cancel a timer for my smart device? # Instruction: Given a "timer cancellation" task, retrieve tools that handle smart device operations by processing device ID and switch time inputs to cancel a scheduled action and return the status of the operation.
Query: Find cruise tickets from Fontana to Santa Rosa on date 2023-07-04. # Instruction: Given a "ticket booking" task, retrieve tools that support booking cruise tickets by processing travel details such as departure location, destination, date, and time.
Example of our generated instructions (high-quality instructions)
Query: Suppose that $f(x) = 4x + 5$. What is $f^{-1}(f^{-1}(9))$? # Instruction: Given a "inverse function calculation" task, retrieve tools that calculate the value of the repeated inverse for a linear function by processing coefficients, constants, and target values to determine the result.
Query: Identify an function that can classify images and works with spiking neural networks. # Instruction: Given an "image classification" task, retrieve tools that execute image classification by using spiking neural network models and processing image inputs.
Query: I need to find a grocery store near 123 Main Street, Downtown District that has a good selection of limes for my Easter celebration. # Instruction": "Given a "local grocery search" task, retrieve tools that can locate grocery stores based on the user's specified location and criteria, such as the availability of specific items like limes, to meet the query's requirements.
Query: Please help me find a recipe with no more than 40 grams of carbohydrates per gram and at least 5 grams of protein per gram. # Instruction: "Given a "nutritional recipe search" task, retrieve tools that can find recipes based on specific nutritional criteria such as carbohydrate and protein content.
Query: What is 64277 times 38142? # Instruction: Given a "multiplication" task, retrieve tools that compute the product of two numbers by processing numerical inputs and returning the result.
Query: Can I get a list of all boards and their attributes on page number two with a page size of seven? # Instruction: Given a "pagination query" task, retrieve tools that can list boards and their attributes by processing parameters such as page number and page size to return the requested information.
Example of our generated instructions (low-quality instructions)
Query: I would like to generate a video presenting a text-based discussion on the topic of 'The Benefits of Exercise' # Instruction: Given a text-to-video task, please retrieve relevant tools to generate video about exercise. **// Too general to cover the key features.** # Revised version: Given a 'text-to-video' task, retrieve tools related to general video scripting, exercise video libraries or tools process text data. **// More specific to the query and target tools.**
Query: Can you assist me in finding a 1-bedroom townhouse or condo in Little Rock with max rent 1541000? I want it on the sixth floor with 7 balconies. # Instruction: Please retrieve tools find 1-bedroom or condo in Little Rock. **// Too general and miss the point of *floor*** # Revised version: Given a property search task, retrieve tools that can find rental properties based on location, property type, rent budget, and specific features or requirements. **// Cover all key points and related to tools functionality.**

Table 8: Example of the instruction in TOOLRET. We show the handcrafted instruction by human experts, the high-quality instruction and low-quality instruction generated by GPT-4o, respectively. We also show the revised version of the low-quality instruction paired with the reason for revision.

Name	Public Link of Endpoint
Conventional sparse and dense models	
BM25S (Lù, 2024)	https://github.com/xhluca/bm25s
contriever (Izacard et al., 2021b)	https://huggingface.co/facebook/contriever-msmarco
ColBERTv2 (Santhanam et al., 2021b)	https://github.com/stanford-futuredata/ColBERT
gtr-t5-base (Ni et al., 2021b)	https://huggingface.co/sentence-transformers/gtr-t5-base
gtr-t5-large (Ni et al., 2021b)	https://huggingface.co/sentence-transformers/gtr-t5-large
Multi-task embedding models	
all-MiniLM-L6-v2	https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
e5-small-v2 (Wang et al., 2022)	https://huggingface.co/intfloat/e5-small-v2
e5-base-v2 (Wang et al., 2022)	https://huggingface.co/intfloat/e5-base-v2
e5-large-v2 (Wang et al., 2022)	https://huggingface.co/intfloat/e5-large-v2
bge-base-en-v1.5 (Xiao et al., 2023b)	https://huggingface.co/BAAI/bge-base-en-v1.5
bge-large-en-v1.5 (Xiao et al., 2023b)	https://huggingface.co/BAAI/bge-large-en-v1.5
gte-Qwen2-1.5B-inst. (Li et al., 2023c)	https://huggingface.co/Alibaba-NLP/gte-Qwen2-1.5B-instruct
e5-mistral-7b (Wang et al., 2023)	https://huggingface.co/intfloat/e5-mistral-7b-instruct
GritLM-7B (Muennighoff et al., 2024)	https://huggingface.co/GritLM/GritLM-7B
NV-Embed-v1 (Lee et al., 2024)	https://huggingface.co/nvidia/NV-Embed-v1
Cross-encoder re-ranking	
mxbai-rerank-large-v1	https://huggingface.co/mixedbread-ai/mxbai-rerank-large-v1
monot5-base (Nogueira et al., 2020)	https://huggingface.co/castorini/monot5-base-med-msmarco
bge-reranker-v2-m3 (Li et al., 2023a; Chen et al., 2024a)	https://huggingface.co/BAAI/bge-reranker-v2-m3
jina-reranker-v2	https://huggingface.co/jinaai/jina-reranker-v2-base-multilingual
bge-reranker-v2-gemma (Li et al., 2023a; Chen et al., 2024a)	https://huggingface.co/BAAI/bge-reranker-v2-gemma
LLM agent	
RankGPT	https://github.com/sunnweiwei/RankGPT
- Mixtral-8x22B	https://huggingface.co/mistralai/Mixtral-8x22B-Instruct-v0.1
- GPT-3.5-turbo-1106	https://openai.com/chatgpt/overview/
- GPT-3.5-turbo-0125	https://openai.com/chatgpt/overview/

Table 9: The public link or endpoint of the baselines in our experiments.

- **Multi-task Embedding Models**. These methods utilize transformer encoders trained on various annotated IR datasets. We evaluate gte (Li et al., 2023c), bge (Xiao et al., 2023a), and e5 (Wang et al., 2022), covering a wide range of parameter sizes. Additionally, we evaluate all-MiniLM-L6-v2[5] from the Sentence Transformers platform.

- **Cross-encoder Re-rankers**. These models re-rank the initially retrieved documents based on the query-passage relevance using bidirectional or unidirectional transformers. We evaluate MonoT5-Base and three re-rankers trained on diverse tasks: (i) mxbai-rerank-large-v1[6], (ii) jina-reranker-v2-base[7], and (iii) BGE-reranker.

- **LLM Agents**. These methods leverage general-purpose LLM agents for re-ranking tasks in a zero-shot setting, simulating the tool selection process of tool-use agents. We evaluate the widely used LLM re-ranking framework, i.e., RankGPT (Sun et al., 2023), with various LLMs as backbone.

We highlight that the initial tools for LLM agent and Re-ranking baselines are retrieved by `NV-embedd-v1 model`. Details about these baselines are provided in Table 9.

D.2 Compare with conventional IR tasks

To further investigate the complexity of tool retrieval tasks, we conducted a comparative analysis of model performance between our proposed benchmark (ToolRet) and the conventional Information Retrieval (IR) task benchmark, specifically the Massive Text Embedding Benchmark (MTEB). The relationship between these two benchmarks is visually presented in Figure 7. Our analysis reveals two significant findings.

First , we observe a strong positive correlation between the two benchmarks, as evidenced by a Pearson's correlation coefficient of $\beta = 0.790$, indicating a similar performance trend across models. However, we observe that the absolute performance scores in ToolRet are consistently lower than those in the

[5] huggingface.co/all-MiniLM-L6-v2
[6] huggingface.co/mxbai-rerank-large-v1
[7] huggingface.co/jina-reranker-v2-base-multilingual

Model	REST API				Code Function				Customized tool				Avg.	
	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	C@10
Conventional sparse and dense models														
BM25s	**62.09**	**15.68**	**72.98**	**58.06**	**56.98**	**8.24**	**73.95**	**72.81**	68.48	**14.78**	**80.51**	69.55	**62.51**	**66.81**
ColBERT	56.73	14.59	67.86	47.87	53.56	7.82	71.66	70.09	64.49	13.05	76.35	64.85	58.26	60.94
contriever-msmarco	57.81	15.33	70.77	50.87	49.66	7.29	65.99	64.56	**68.86**	14.67	83.05	73.45	58.77	62.96
gtr-t5-base	57.06	14.54	68.26	49.75	49.38	7.29	65.76	64.09	70.48	14.29	81.60	70.96	58.97	61.60
gtr-t5-large	60.32	15.27	72.05	54.03	52.79	7.41	67.28	65.79	72.03	14.69	84.39	**73.95**	61.72	64.59
Embedding models														
all-MiniLM-L6-v2	53.92	14.47	66.77	48.64	50.14	7.58	68.39	66.74	68.31	14.20	81.57	71.14	57.46	62.17
e5-small-v2	61.95	15.84	72.91	53.34	51.45	7.76	68.16	65.46	69.13	14.24	79.69	68.33	60.85	62.38
e5-base-v2	62.90	15.90	73.98	53.83	55.81	8.44	74.17	72.53	69.96	14.98	**83.63**	73.94	62.89	**66.77**
e5-large-v2	61.72	15.90	73.27	52.84	56.21	8.42	75.25	**73.14**	69.88	**15.01**	81.13	**71.30**	62.60	65.76
gte-base-en-v1.5	64.35	**16.55**	75.80	57.38	**59.18**	**8.77**	**76.95**	**74.45**	71.79	14.53	81.90	70.07	65.11	67.30
gte-large-en-v1.5	60.67	15.46	72.30	52.41	54.11	8.22	73.35	71.37	68.59	14.36	80.41	69.82	61.12	64.53
bge-base-en-v1.5	65.05	16.37	75.72	57.30	54.55	7.72	69.22	67.48	71.21	14.71	83.13	72.53	63.60	65.77
bge-large-en-v1.5	**66.25**	16.48	**75.84**	**57.75**	58.61	8.41	74.91	72.74	**71.19**	14.20	80.44	69.27	**65.35**	66.59
gte-Qwen2-1.5B-inst.	67.57	16.93	78.14	60.81	58.12	8.51	75.41	73.39	71.73	15.34	83.03	73.39	65.81	69.19
e5-mistral-7b	**69.51**	**17.37**	**79.34**	**62.48**	58.15	8.37	75.12	72.79	72.52	14.68	81.79	71.49	66.73	68.92
GritLM-7B	69.43	17.25	78.97	61.67	62.78	9.22	78.74	77.59	**76.04**	15.44	85.55	74.35	**69.42**	71.21
NV-Embed-v1	66.04	16.88	77.19	59.06	**63.46**	**9.40**	**81.79**	**79.82**	75.39	**15.75**	**88.48**	**78.37**	68.30	**72.42**
Cross-encoder re-ranking models														
mxbai-rerank-large-v1	57.48	14.60	68.65	49.54	50.37	7.75	69.59	67.88	62.24	13.32	73.26	61.24	56.70	59.55
monot5-base-msmarco	54.57	14.23	64.38	46.12	50.00	8.05	68.76	66.80	64.50	13.28	75.80	67.84	56.36	60.25
bge-reranker-v2-m3	70.42	17.75	80.33	65.49	64.22	9.37	80.60	79.48	75.70	16.15	88.87	78.65	70.11	74.54
bge-reranker-v2-gemma	**75.67**	**18.63**	**84.07**	**71.00**	**69.59**	**9.78**	**84.18**	**83.47**	**77.17**	**16.55**	**88.34**	**79.80**	**74.14**	**78.09**

Table 10: Results of control experiment where each IR models is evaluated from the toolset of each integrated dataset in ***w/ inst.*** setting.

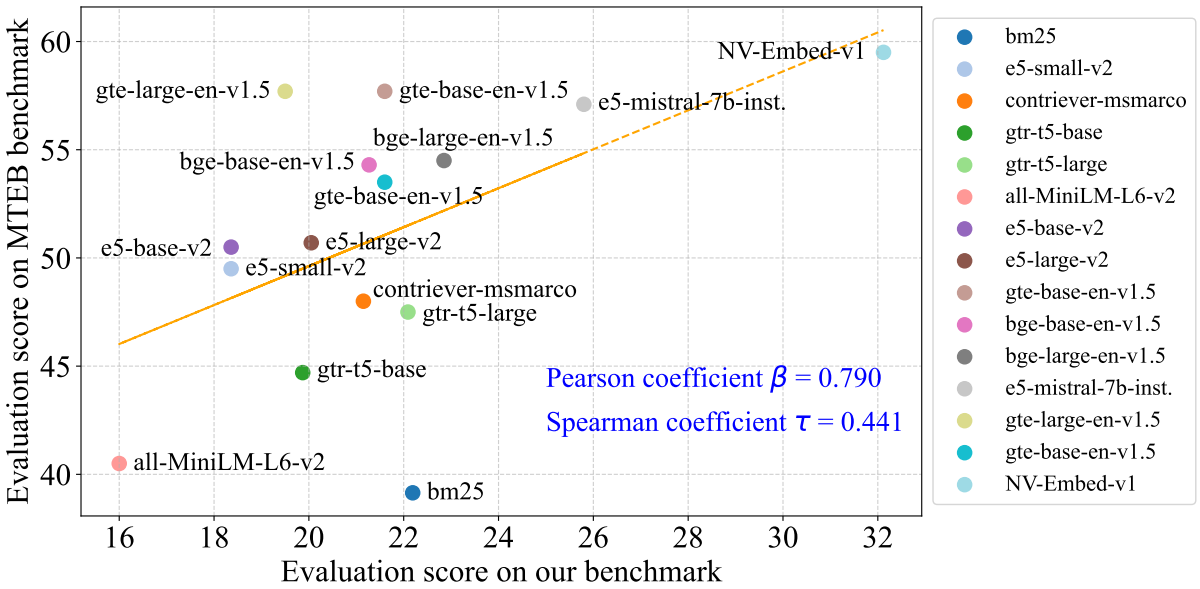

Figure 7: Correlation between the score on our benchmark and MTEB (retrieval subset).

conventional IR benchmark. This discrepancy suggests that while our benchmark shares fundamental characteristics with conventional IR tasks, it presents additional challenges that make it more demanding for existing models.

Second , our experimental results demonstrate that state-of-the-art IR models, particularly those trained with relevance-oriented optimization criteria (e.g., Contriever), exhibit substantially degraded performance on ToolRet. This performance gap underscores the necessity for target-aware reasoning capabilities in our benchmark, which goes beyond traditional relevance matching. The unique challenges of ToolRet are further elaborated in § 4.1, where we identify two key distinguishing factors: (1) the presence of

Model	ToolRet-Web				ToolRet-Code				ToolRet-Customized				Avg.	
	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	C@10
Conventional sparse and dense models														
BM25S	51.79	13.78	63.35	45.46	**38.74**	**5.87**	52.65	51.39	59.72	**13.55**	**71.83**	60.38	50.08	52.41
ColBERT	51.70	13.56	61.92	41.01	38.60	6.05	**55.07**	**54.05**	53.91	12.10	66.85	55.29	48.07	50.11
contriever-msmarco	53.23	14.55	65.96	46.59	35.97	5.79	52.32	50.84	**56.94**	13.21	72.11	61.17	48.71	52.87
gtr-t5-base	51.65	14.13	64.55	44.56	33.98	5.51	48.88	47.77	54.28	13.24	70.95	60.82	46.64	51.05
gtr-t5-large	**56.62**	**15.06**	**69.77**	**50.26**	37.40	5.85	52.41	51.27	56.24	13.31	71.25	**61.40**	**50.09**	**54.31**
Embedding models														
all-MiniLM-L6-v2	48.49	13.54	62.35	43.63	34.40	5.62	50.15	48.71	58.08	13.22	72.72	62.17	46.99	51.50
e5-small-v2	54.40	14.60	66.47	46.76	35.18	5.67	50.70	49.19	56.85	13.45	73.43	60.86	48.81	52.27
e5-base-v2	55.42	14.92	67.90	48.10	38.35	6.23	56.08	54.90	59.96	14.18	76.18	66.55	51.24	56.52
e5-large-v2	54.32	14.81	67.69	47.89	40.24	6.23	56.33	54.85	59.40	13.79	72.96	60.37	51.32	54.37
gte-base-en-v1.5	56.48	**15.40**	**70.07**	**50.96**	39.46	**6.27**	**56.58**	**55.24**	**64.00**	**14.23**	**78.03**	**66.93**	53.31	**57.71**
gte-large-en-v1.5	55.39	14.89	68.33	48.98	38.23	6.22	56.16	54.95	57.88	13.86	75.12	65.15	50.50	56.36
bge-base-en-v1.5	56.17	14.86	68.30	49.05	38.71	6.05	54.35	53.08	59.40	13.93	75.38	64.81	51.43	55.65
bge-large-en-v1.5	**58.20**	15.33	69.83	50.20	**40.39**	6.13	55.03	53.67	61.40	13.83	77.27	66.71	**53.33**	56.86
gte-Qwen2-1.5B-inst.♠	60.28	15.64	72.60	53.82	44.06	6.56	59.29	57.78	65.57	14.79	**80.33**	**70.59**	56.64	60.73
e5-mistral-7b♠	60.78	15.93	73.12	55.60	44.20	6.77	61.18	59.79	60.56	13.80	74.47	63.38	55.18	59.59
GritLM-7B♠	**62.54**	**16.07**	73.82	54.46	46.80	6.98	62.89	61.35	**67.61**	**14.93**	80.04	68.41	58.98	61.41
NV-Embed-v1♠	61.76	16.02	**73.86**	**55.96**	**50.38**	**7.54**	**67.93**	**66.03**	67.01	14.61	79.70	69.74	**59.72**	**63.91**
Cross-encoder re-ranking models														
mxbai-rerank-large-v1	56.45	14.51	67.51	49.02	42.55	6.28	57.85	55.96	54.63	13.03	72.68	60.20	51.21	55.06
monot5-base-msmarco	56.10	14.82	65.29	47.12	41.05	6.31	57.20	55.14	64.60	13.91	75.79	65.76	53.92	56.01
bge-reranker-v2-m3	61.78	15.94	72.35	55.61	45.15	6.74	61.03	59.56	62.45	14.86	79.65	68.68	56.46	61.28
jina-reranker-v2-base	**65.86**	**17.14**	**77.54**	**62.33**	47.23	7.01	63.50	62.24	**69.10**	15.38	81.29	69.63	60.73	64.73
bge-reranker-v2-gemma	65.80	16.87	76.85	61.40	**52.49**	**7.60**	**68.14**	**65.94**	67.87	**15.63**	**81.98**	**72.24**	**62.05**	**66.53**

Table 11: Results of control experiment where each IR models is evaluated from the toolset of each integrated dataset in ***w/o inst.*** setting.

multiple potential target tools for each query, and (2) significantly lower term overlap between input queries and relevant tools compared to conventional IR scenarios. These characteristics collectively contribute to a more complex retrieval environment that requires advanced reasoning and understanding capabilities from retrieval models.

D.3 Results of controlled experiment

Since ToolRet integrates multiple datasets, we also conduct controlled experiments where IR models retrieve tools exclusively within the toolset of each individual dataset instead of the overall tool corpus. Table 11 presents the results under the setting that the IR models only take the query to retrieve, i.e., the *w/o inst* setting. Table 10 presents the results under the setting that the IR models take the query and additional instruction to retrieve, i.e., the *w/ inst* setting.

D.4 Results of in-subset retrieval

ToolRet contains three subsets, including ToolRet-web, ToolRet-code and ToolRet-customized. The tool in each subset diverges by its documentation format, domain, and functionality. For a comprehensive evaluation, we also conduct an in-subset retrieval experiment, where IR models retrieve tools exclusively within the toolset of each subset instead of the overall tool corpus. Table 13 presents the results under the setting that the IR models only take the query to retrieve, i.e., the *w/o inst* setting. Table 12 presents the results under the setting that the IR models take the query and additional instruction to retrieve, i.e., the *w/ inst* setting.

D.5 Results of trained IR mdels

Experimental results on ToolRet reveal that even IR models with strong performance on conventional IR benchmarks such as MTEB and BEIR struggle significantly in tool retrieval tasks. A key factor contributing to this performance degradation is the lack of a large-scale training dataset specifically tailored for tool retrieval. To address this gap, we introduce ToolRet-train, a diverse training dataset comprising more than 200k tool retrieval tasks. Each example in ToolRet-train consists of an input query, an instruction generated using our target-aware strategy, the corresponding target tools, and a set of

Model	ToolRet-Web				ToolRet-Code				ToolRet-Customized				Average	
	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	C@10
Sparse and dense models														
bm25	49.01	7.17	64.96	63.68	28.92	6.78	37.09	24.44	51.28	10.66	60.70	48.40	43.07	45.51
COLT	36.58	5.74	50.84	49.04	21.98	5.12	29.68	20.03	46.02	9.12	58.02	45.27	34.86	38.12
Colbert	43.80	6.40	58.02	56.28	16.60	3.05	20.85	14.95	31.18	5.86	39.40	32.10	30.53	34.44
contriever-msmarco	35.78	5.31	47.17	46.07	25.19	5.67	31.95	20.56	44.37	9.35	57.53	46.80	35.11	37.81
gtr-t5-base	37.45	5.50	48.78	47.44	22.54	4.93	29.64	20.60	51.02	10.28	61.08	49.06	37.00	39.03
gtr-t5-large	42.14	5.98	53.59	52.19	26.60	5.71	33.68	22.38	53.95	11.17	66.08	52.21	40.90	42.26
Embedding models														
all-MiniLM-L6-v2	36.93	5.65	49.54	47.92	15.89	3.89	22.68	15.24	43.09	9.29	56.84	43.96	31.97	35.71
e5-small-v2	38.22	5.55	49.34	48.07	28.97	6.81	36.96	23.45	47.59	9.78	58.19	45.31	38.26	38.94
e5-base-v2	40.69	6.51	55.38	54.01	28.43	6.59	37.14	23.67	47.89	9.48	59.01	46.91	39.00	41.53
e5-large-v2	40.14	5.87	52.23	50.80	26.88	6.16	35.65	24.31	51.40	10.65	61.45	48.28	39.47	41.13
gte-base-en-v1.5	48.25	7.16	61.96	59.17	33.28	7.57	41.99	27.20	50.33	9.70	62.05	50.09	43.95	45.49
gte-large-en-v1.5♠	40.48	6.46	56.34	54.52	30.58	7.00	38.79	24.78	49.24	9.98	59.13	47.20	40.10	42.17
bge-base-en-v1.5	43.74	6.43	57.33	55.85	29.83	6.96	38.86	25.67	52.41	10.75	63.84	51.19	41.99	44.24
bge-large-en-v1.5	44.07	6.44	56.78	55.22	33.88	7.90	43.11	28.62	53.48	10.53	63.66	52.00	43.81	45.28
gte-Qwen2-1.5B-inst.♠	47.29	7.10	61.89	59.98	39.30	9.90	48.58	29.51	55.56	11.67	65.55	51.75	47.38	47.08
e5-mistral-7b♠	48.76	7.16	63.57	61.40	33.06	8.14	43.56	28.49	57.16	11.43	67.28	52.62	46.32	47.51
GritLM-7B	53.69	8.13	68.70	67.26	41.59	10.06	51.69	33.87	60.14	11.63	68.93	54.60	51.81	51.91
NV-Embed-v1♠	51.95	7.62	66.58	64.21	34.42	8.40	43.66	29.65	57.93	12.34	71.14	57.47	48.10	50.44
Cross-encoder re-ranking models														
mxbai-rerank-large-v1	33.39	5.09	46.75	45.11	24.90	5.95	32.20	19.52	35.62	7.35	43.96	34.14	31.30	32.92
monot5-base-msmarco	29.95	5.20	42.47	40.36	30.20	7.84	37.94	21.43	46.99	9.26	57.10	46.48	35.71	36.09
bge-reranker-v2-m3	56.49	8.42	72.26	70.67	38.09	9.25	48.14	33.48	54.66	12.46	70.79	56.17	49.75	53.44
jina-reranker-v2-base	35.24	5.48	47.70	46.31	36.23	9.21	45.59	29.26	54.12	11.92	65.09	51.18	41.86	42.25
bge-reranker-v2-gemma	62.84	8.87	76.24	74.86	37.13	8.62	47.23	33.45	64.33	13.72	77.37	62.08	54.76	56.79

Table 12: Experiments are conducted under the ***w/ inst.*** setting, with retrieval performed within each subset individually.

negative tools. IR models are trained to distinguish target tools from negative tools (§ 7). We evaluate these trained IR models on ToolRet and present the results in Table 14.

D.6 Improved IR enhances tool-use LLMs

We further investigate the impact of improved IR models on the end-to-end performance of tool-use LLMs. Specifically, we evaluate tool-use LLMs on the ToolBench (Qin et al., 2023) dataset using the official *Pass Rate* metric, which measures whether the model successfully invokes the correct tools to complete a given task.

For each task in ToolBench, we replace the pre-annotated toolset (oracle) with tools retrieved by IR models from ToolRet' tool corpus, which contains 43,000 tools. Since ToolRet integrates the ToolBench dataset, we can compute NDCG@10 for this retrieval step. For a comprehensive evaluation, we assess two widely used tool-use LLMs, including GPT-3.5 and ToolLLaMA (Qin et al., 2023).

Table 15 presents the retrieval NDCG@10 scores alongside the corresponding pass rates on ToolBench.[8] Our results demonstrate that LLM agents equipped with improved IR models achieve substantial gains in pass rate, highlighting the critical role of accurate tool retrieval in downstream task performance. Furthermore, Figure 6 visually illustrates a positive correlation between improved IR performance and higher task pass rate, suggesting that better retrieval directly leads to improved downstream outcomes.

Based on this analysis, we propose that future work could explore the following two directions: (i) Further optimize IR models to enhance tool retrieval performance; or (ii) Adapt IR models by incorporating feedback from end-to-end task performance, allowing them to better support tool-use LLMs. These approaches provide a more efficient plug-and-play solution compared to fine-tuning LLMs, enabling flexible integration into diverse tool-use systems.

[8]ToolBench consists of three subsets: ToolBench-G1, ToolBench-G2, and ToolBench-G3.

Model	ToolRet-Web				ToolRet-Code				ToolRet-Customized				Average	
	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	C@10
Sparse and dense models														
bm25	26.47	4.04	34.51	33.22	20.61	5.06	26.35	15.87	38.48	8.51	48.24	37.09	28.52	28.73
COLT	22.23	3.75	31.66	30.15	21.65	5.36	29.15	19.12	36.12	7.99	47.63	37.64	26.67	28.97
Colbert	23.58	3.84	32.64	31.19	23.50	5.70	28.82	16.03	33.71	6.64	40.85	32.01	26.93	26.41
contriever-msmarco	19.11	3.16	26.52	24.97	21.84	5.93	27.83	15.19	35.01	7.57	44.07	34.71	25.32	24.96
gtr-t5-base	21.19	3.48	29.05	27.90	18.18	4.45	25.09	16.32	35.95	8.10	45.77	36.50	25.11	26.91
gtr-t5-large	24.48	3.91	33.55	32.44	23.49	5.64	30.79	19.31	38.88	8.89	49.55	38.64	28.95	30.13
Embedding models														
all-MiniLM-L6-v2	18.07	3.10	25.13	23.75	13.71	3.49	18.73	11.78	31.61	7.12	40.66	30.03	21.13	21.85
e5-small-v2	20.10	3.14	26.47	25.12	21.02	5.33	27.70	16.88	32.58	7.35	39.85	30.33	24.57	24.11
e5-base-v2	20.96	3.54	29.43	28.23	21.25	5.43	27.59	16.42	32.90	7.33	43.10	33.51	25.04	26.05
e5-large-v2	22.93	3.49	29.53	28.09	20.16	5.19	27.26	16.68	39.45	8.81	49.07	38.19	27.51	27.65
gte-base-en-v1.5	24.50	3.98	33.85	32.48	24.32	6.46	33.01	19.91	37.86	8.10	49.44	38.94	28.89	30.44
gte-large-en-v1.5♠	23.00	3.90	33.07	31.93	23.84	6.19	31.38	19.15	35.34	8.24	45.43	35.28	27.39	28.79
bge-base-en-v1.5	23.44	3.86	32.79	31.47	24.17	6.37	31.80	18.68	36.59	8.47	47.44	36.87	28.07	29.01
bge-large-en-v1.5	23.12	3.76	31.93	30.67	27.14	7.24	35.77	21.16	35.51	7.82	45.71	36.39	28.59	29.40
gte-Qwen2-1.5B-inst.♠	28.99	4.57	39.58	38.16	31.85	8.37	39.95	22.69	45.47	10.02	55.68	43.85	35.44	34.90
e5-mistral-7b	25.38	4.13	34.81	33.47	28.22	7.60	35.99	22.28	42.31	9.03	51.34	40.24	31.97	32.00
GritLM-7B♠	29.67	4.82	41.30	39.77	30.86	8.18	39.99	25.25	48.92	10.31	59.01	46.55	36.48	37.19
NV-Embed-v1♠	35.50	5.54	48.36	46.72	33.08	8.77	41.08	24.83	51.28	11.13	61.49	48.05	39.95	39.87
Cross-encoder re-ranking models														
mxbai-rerank-large-v1	31.75	4.79	43.82	42.20	24.84	5.96	32.13	19.53	35.65	7.31	43.38	33.43	30.75	31.72
monot5-base-msmarco	26.91	4.32	37.16	35.03	30.43	8.11	38.04	20.45	46.32	9.15	56.53	46.36	34.56	33.95
bge-reranker-v2-m3	31.25	4.92	42.97	41.34	34.04	8.86	42.85	26.46	43.81	10.38	53.28	41.84	36.36	36.54
jina-reranker-v2-base	33.31	5.06	44.20	42.93	36.79	9.49	46.20	28.55	53.18	11.56	63.22	50.70	41.09	40.72
bge-reranker-v2-gemma	38.59	5.67	50.14	48.67	38.08	10.05	47.65	28.98	50.39	11.54	61.69	49.48	42.36	42.38

Table 13: Experiments are conducted under the ***w/ inst.*** setting, with retrieval performed within each subset individually.

Model	ToolRet-Web				ToolRet-Code				ToolRet-Customized				Avg.	
	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	P@10	R@10	C@10	N@10	C@10
bge-large-en-v1.5‡	25.07	3.94	33.52	31.85	32.05	7.98	41.03	26.11	41.72	8.80	49.26	37.99	32.95	31.98
bge-large-en-v1.5‡	23.41	3.22	32.02	31.97	30.74	6.31	36.03	24.31	37.31	7.30	46.31	34.21	30.49	30.16
bge-large-en-v1.5	18.90	3.13	25.80	24.50	24.49	6.67	32.95	19.30	25.72	5.54	32.18	24.79	23.03	22.86
bge-base-en-v1.5‡	20.35	3.41	28.19	26.77	30.69	7.70	39.16	24.95	37.01	8.53	47.21	36.40	29.35	29.37
bge-base-en-v1.5†	19.05	2.98	25.13	23.70	24.90	6.41	35.44	22.50	33.51	6.40	44.20	35.71	25.82	27.30
bge-base-en-v1.5	17.76	2.91	23.59	22.20	22.44	6.03	29.96	17.29	25.98	5.71	32.17	24.26	22.06	21.25
e5-large-v2‡	23.15	3.74	31.41	29.94	33.05	7.79	40.42	26.97	34.33	6.60	42.08	34.26	30.18	30.39
e5-large-v2†	21.33	2.94	28.34	25.13	30.45	6.45	38.20	26.70	30.13	5.60	39.82	32.33	27.30	28.05
e5-large-v2	17.03	2.67	21.77	20.63	18.94	4.90	25.95	16.26	26.37	6.07	32.19	23.17	20.78	20.02
e5-base-v2‡	19.97	3.33	27.67	26.36	26.45	5.92	32.71	22.24	31.03	6.06	38.42	30.92	25.81	26.51
e5-base-v2†	15.44	2.78	25.37	23.61	24.50	5.20	30.12	19.37	28.03	5.47	40.21	31.92	22.66	24.97
e5-base-v2	14.42	2.46	19.18	18.00	19.80	5.04	25.89	15.37	22.69	5.11	29.13	22.25	18.97	18.54

Table 14: Experimental results of IR models before and after training on our datasets. Models trained with the concatenation of instruction and query are denoted by ‡. In contrast, the variants trained solely on the query as input are marked with † (See the ablation study in § 7 for details).

Dataset	TOOLRET	ToolBench-G1		ToolBench-G2		ToolBench-G3	
	NDCG@10	NDCG@10	Pass Rate	NDCG@10	Pass Rate	NDCG@10	Pass Rate
gpt-3.5-turbo as tool-use LLM							
oracle	-	-	62.00	-	57.20	-	67.40
bge-large-en-v1.5	23.03	34.29	50.60	9.48	49.00	29.69	56.90
bge-large-en-v1.5 ♠	$32.95_{\uparrow 43.07\%}$	$71.11_{\uparrow 107.38\%}$	$59.50_{\uparrow 17.59\%}$	$18.11_{\uparrow 91.03\%}$	$58.40_{\uparrow 19.18\%}$	$67.87_{\uparrow 128.60\%}$	$59.20_{\uparrow 4.04\%}$
bge-base-en-v1.5	22.06	36.89	50.60	9.28	51.20	33.02	57.70
bge-base-en-v1.5♠	$29.35_{\uparrow 33.05\%}$	$67.52_{\uparrow 83.03\%}$	$56.60_{\uparrow 11.86\%}$	$16.01_{\uparrow 72.52\%}$	$59.60_{\uparrow 16.41\%}$	$60.75_{\uparrow 83.98\%}$	$60.80_{\uparrow 5.37\%}$
e5-large-v2	20.78	44.91	47.50	11.57	56.50	43.43	55.70
e5-large-v2♠	$30.18_{\uparrow 45.24\%}$	$70.08_{\uparrow 56.05\%}$	$57.00_{\uparrow 20.0\%}$	$17.71_{\uparrow 53.07\%}$	$62.10_{\uparrow 9.91\%}$	$66.09_{\uparrow 52.18\%}$	$58.00_{\uparrow 3.99\%}$
e5-base-v2	18.97	38.66	49.60	9.87	54.10	37.35	54.20
e5-base-v2♠	$25.81_{\uparrow 36.06\%}$	$65.79_{\uparrow 70.18\%}$	$56.90_{\uparrow 14.72\%}$	$17.45_{\uparrow 76.80\%}$	$60.80_{\uparrow 12.38\%}$	$62.74_{\uparrow 67.98\%}$	$62.40_{\uparrow 15.13\%}$
ToolLlama as tool-use LLM							
oracle	-	-	53.6	-	50.8	-	49.1
bge-large-en-v1.5	23.03	34.29	37.60	9.48	41.30	29.69	37.20
bge-large-en-v1.5♠	$32.95_{\uparrow 43.07\%}$	$71.11_{\uparrow 107.38\%}$	$45.10_{\uparrow 19.95\%}$	$18.11_{\uparrow 91.03\%}$	$47.30_{\uparrow 14.53\%}$	$67.87_{\uparrow 128.60\%}$	$39.60_{\uparrow 6.45\%}$
bge-base-en-v1.5	22.06	36.89	47.80	9.28	46.10	33.02	36.10
bge-base-en-v1.5♠	$29.35_{\uparrow 33.05\%}$	$67.52_{\uparrow 83.03\%}$	$50.60_{\uparrow 5.86\%}$	$16.01_{\uparrow 72.52\%}$	$49.80_{\uparrow 8.03\%}$	$60.75_{\uparrow 83.98\%}$	$45.70_{\uparrow 26.60\%}$
e5-large-v2	20.78	44.91	41.50	11.57	46.60	43.43	40.20
e5-large-v2♠	$30.18_{\uparrow 45.24\%}$	$70.08_{\uparrow 56.05\%}$	$44.50_{\uparrow 7.23\%}$	$17.71_{\uparrow 53.07\%}$	$49.80_{\uparrow 4.72\%}$	$66.09_{\uparrow 52.18\%}$	$43.80_{\uparrow 4.50\%}$
e5-base-v2	18.97	38.66	42.20	9.87	45.20	37.35	42.00
e5-base-v2♠	$25.81_{\uparrow 36.06\%}$	$65.79_{\uparrow 70.18\%}$	$49.10_{\uparrow 16.35\%}$	$17.45_{\uparrow 76.80\%}$	$48.30_{\uparrow 6.86\%}$	$62.74_{\uparrow 67.98\%}$	$44.70_{\uparrow 6.60\%}$

Table 15: Experiment results of IR models before and after training. We also show the end-to-end task pass rate of tool-use LLMs when equipped with the tools retrieved by the IR models on ToolBench dataset.