⚓ PORTS: Preference-Optimized Retrievers for Tool Selection with Large Language Models

Lorenzo Molfetta Giacomo Frisoni Nicolò Monaldini Gianluca Moro
Department of Computer Science and Engineering, University of Bologna
{lorenzo.molfetta, giacomo.frisoni, gianluca.moro}@unibo.it,
nicolo.monaldini@studio.unibo.it

Abstract

Integrating external tools with Large Language Models (LLMs) has emerged as a promising paradigm for accomplishing complex tasks. Since LLMs still struggle to effectively manage large tool collections, researchers have begun exploring retrieval-based methods to preselect the most relevant options, addressing input length and latency constraints. However, existing retrievers are often misaligned with tool-calling LLMs due to their separate training processes. This paper presents PORTS, a novel odds ratio preference optimization method for training retrievers aimed at tool selection. Using a perplexity-inspired preference signal from a frozen LLM, our approach fine-tunes a retriever to find helpful tools by optimizing the correlation between the selection probabilities and the downstream performances while jointly enforcing a contrastive semantic loss between documentation strings. The versatility of PORTS and its ability to significantly improve tool selection accuracy are demonstrated through extensive experiments on six datasets, two encoder models, and three LLMs with diverse prior knowledge. With low computational demands, our alignment process facilitates generalization to new queries and tools, proving valuable for practical applications with evolving toolsets.[1]

1 Introduction

"The right tool for the right job."—Proverb

Equipping Large Language Models (LLMs) with the capability to dynamically interact with external tools[2] has garnered significant research attention. This integration not only improves the problem-solving potential of LLMs, but also dramatically expands their functional scope (Yao et al., 2022; Lazaridou et al., 2022). When presented with a user query, tool-augmented LLMs can determine when and how to utilize specific tools to generate more accurate and informative responses. For example, tools can enable LLMs to use a calculator, set calendar events, and access real-time weather information. As the field continues to evolve, LLMs with tools are expected to play a pivotal role in shaping the future of Natural Language Processing (NLP) (Qu et al., 2024b).

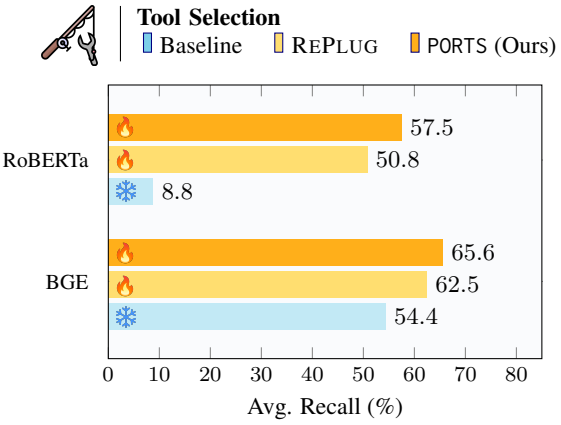


Figure 1: **Results overview.** Comparison between frozen, REPLUG-tuned, and PORTS-tuned retrievers. Scores are averaged across Recall@{1,2,3} for three LLMs (if trained) and six datasets (test set).

Fine-tuning LLMs with tool usage examples is expensive and confines the acquired knowledge to a predefined set of tools (Qiao et al., 2023; Yang et al., 2023). The in-context learning paradigm alleviates these issues, but the limitations in input length and noise for lengthy prompts make it impractical to manage many descriptions or demonstrations directly (Liu et al., 2024; Qu et al., 2024a), introducing efficiency and accuracy challenges in tool-selection tasks, mainly when precise parameter specification and schema typing are paramount. Furthermore, when faced with hundreds of tool docstrings in the prompt, the language model alone

[1]Code, models, and datasets are publicly available at https://github.com/disi-unibo-nlp/ports

[2]Consistent with Qu et al. (2024b), we argue that all external means of augmenting LLMs should be classified as tools. Accordingly, we regard individual APIs as separate tools.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 10019–10042
November 4-9, 2025

struggles to identify the most suitable one, often resulting in suboptimal performance (Qu et al., 2024b), increased computational needs, and high costs. Recently, the focus has shifted towards the use of retrievers to effectively support LLMs in the tool selection process (Qin et al., 2024; Gao et al., 2024; Anantha et al., 2023). Retrieval-enhanced pipelines filter the top-K most suitable tools for a given query, aiming to reduce noise and enhance the LLM's ability to select the right tool and configure the necessary parameters for calling.

Although several publications have explored retrievers for tool selection, optimization of the retrieval component itself has received little consideration. Clustering approaches have proven successful in multi-dimensional representation spaces (Lodi et al., 2010), but current methods predominantly employ non-parametric indexing techniques (Patil et al., 2023) or standard encoder models trained with supervised signals (Qin et al., 2024). Whereas these methodologies can be effective in isolation, they often falter when integrated into a broader pipeline, primarily due to misalignment between the training criteria used for the retrieval and generation modules. A significant challenge arises when tools with descriptions similar to the user query are ultimately irrelevant or potentially misleading for the LLM (Yu et al., 2023; Shi et al., 2023). Furthermore, tools can exhibit subtle differences, such as variations in the names, numbers, and types of input parameters, which complicate effective selection–an issue that is increasingly prevalent given the rapid proliferation of publicly available tools and Model Context Protocol servers. In such scenarios, conditioning the encoder on the LLM output may provide additional training signals that benefit the selection process. However, most existing retriever adaptation techniques require the LLM to be trained from scratch (Izacard et al., 2023; Lin et al., 2024; Cheng et al., 2023), which can be prohibitively costly or unfeasible with closed-source solutions characterized by no access to internal representations. Recent research has explored an alternative approach: training encoder models with the support of LLMs, using them as supervision signals to better align representations with task-specific objectives (Bolya et al., 2025). We focus on this emerging class of methods, investigating how LLMs can be effectively leveraged and optimized to guide encoder alignment through feedback incorporation for tool retrieval.

In this paper, we propose a new method to train preference-optimized retrievers for tool selection (PORTS), aligning them with the needs of the LLM responsible for tool usage. Our training scheme adapts a pre-trained encoder model with supervision signals from a black-box LLM, preferring retrieving tool docstrings that stimulate the downstream LLM to use the right tool. Introducing a novel contrastive preference loss enables a more accurate selection process in application domains where multiple tools might adequately serve the task and yield coherent yet inaccurate results. We conduct experiments on six public datasets. The results are analyzed with various classes of encoders and LLMs. To gauge generalizability, tests are carried out with in-domain and out-of-domain tools. We conclude that PORTS can effectively increase the tool selection performance of the baseline retriever, with low computational overhead. Figure 1 shows the averaged metric gains of our alignment method in retrieving tools seen during training.

2 Related Work

Tool Learning Recent studies in language modeling have explored the use of non-differentiable tools to supplement the knowledge stored in the model weights, offloading tasks to external modules. They broadly fall into two categories. *Tuning-based methods* train models to use one or a few tools in specific domains. Example works include TALM (Parisi et al., 2022), Toolformer (Schick et al., 2023), ToolLLaMA (Qin et al., 2024), Gorilla (Patil et al., 2023), ToolkenGPT (Hao et al., 2023), and Granite (Abdelaziz et al., 2024). These models are trained on datasets where the input text is augmented with tool calls. During inference, when such invocations are identified, the decoding process is paused, the corresponding tool is executed, and the result is incorporated before resuming text generation. Specialized tool-calling models mostly rely on synthetic instruction-tuning data generated from proprietary models such as GPT-4 (OpenAI, 2023). Among the few exceptions intended for commercial applications is the NexusRaven series (Nexusflow.ai, 2023). However, LLM fine-tuning is only applicable to open-source models and is generally hindered by expensive data collection and computing infrastructure, as well as poor flexibility in accommodating emergent or updated tools. Conversely, *tuning-free methods* are compatible with all LLMs and capitalize on in-context learning abilities, showing tool descriptions

and demonstrations directly in the prompt (Shen et al., 2023; Song et al., 2023; Yao et al., 2023). As the tool arsenal grows (e.g., 16.000+ (Qin et al., 2024)), a retriever becomes essential. Retrieval-based and fine-tuning methods can be combined to achieve better performance (Gao et al., 2024).

Tool Retrieval Merging Retrieval-Augmented Generation (RAG) and tool calling enables LLMs to evaluate a small subset of retrieved tools and select the most suitable for response formulation. Tool retrieval approaches can be classified into two main types: term-based and semantic-based. Term-based techniques, exemplified by TF-IDF (Jones, 2004) and BM25 (Robertson and Zaragoza, 2009), rely on exact term matching and utilize sparse representations for both tool docstrings and queries. For instance, Gorilla (Patil et al., 2023) implements tool retrieval by combining BM25 with GPT-Index. In contrast, semantic-based methods utilize neural networks to learn the relationships between queries and tool descriptions. CRAFT (Yuan et al., 2024), for example, instructs LLMs to generate fictitious tool descriptions conditioned on the input queries, then uses pre-trained SimCSE for similarity computation. Few studies focus on training the retriever itself, while iterative refinement with cosine similarity has proven effective for adapting representations (Domeniconi et al., 2014c,b, 2015). TPTU (Kong et al., 2023), ToolLLaMA (Qin et al., 2024) and Confucius (Gao et al., 2024) fine-tune a SentenceBERT model using contrastive learning objectives. ProTIP (Anantha et al., 2023) fine-tunes BERT-base with a contrastive loss optimized for progressive tool retrieval. COLT (Qu et al., 2024a) models collaborative relationships among multiple tools using graphs and implements tool retrieval through cross-view graph contrastive learning. ToolRerank (Zheng et al., 2024) addresses the re-ranking stage of tool retrieval, proposing an adaptive and hierarchy-aware method. However, these studies do not consider LLM preferences in specializing the retriever.

3 Methodology

In this section, we introduce PORTS and elaborate on its theoretical motivations, design, and training losses. Figure 2 illustrates our architecture.

3.1 Preliminary

RAG is a widely used framework for augmenting LLMs with external knowledge sources. While models like RETRO (Borgeaud et al., 2022) and FiD (Izacard and Grave, 2021) achieve significant improvements through separate training of retrieval and generation components, end-to-end training approaches offer potential for enhanced relevance, coherence, and contextual awareness (Guu et al., 2020; Li et al., 2024). However, end-to-end training poses challenges, including high computational demands, complex data handling, and the difficulty of maintaining dynamic search indexes with accurate, up-to-date embeddings. Techniques like batch negative sampling (Karpukhin et al., 2020) have improved efficiency, but require careful selection of diverse negatives. Adapting RAG systems to new domains often necessitates simultaneous retraining of both the retriever and generator, underscoring the importance of efficient data management (Siriwardhana et al., 2023). To address these challenges, methods such as REPLUG LSR (Shi et al., 2024) use frozen language models as references to optimize retrieval without costly full-model fine-tuning. We argue that reducing output uncertainties alone is insufficient for aligning with users' goals when selecting tools for tasks. Retrieval models must effectively navigate reference corpora by distinguishing between semantically similar but irrelevant options. Inspired by techniques from LLM fine-tuning, such as RLHF (Christiano et al., 2017), DPO (Rafailov et al., 2023), and ORPO (Hong et al., 2024), retrieval models can leverage comparative loss to prioritize relevant information, mirroring approaches like triplet learning.

3.2 Task Definition

Given an input query q, we aim to pair it with a candidate tool t_i from a predefined set $\mathbf{T} = \{t_1, t_2, \ldots, t_{|\mathbf{T}|}\}$ by maximizing their semantic alignment. PORTS is trained to prioritize the retrieval of tools' docstrings d_{t_i} that most enhance tool calling accuracy when prompted to the LLM.

3.3 PORTS

Our approach optimizes a retrieval model $\mathcal{R}$ through a dual training strategy accounting for query-docstring semantic similarity and tool support in correct answer prediction. We shape probabilities over available data by enforcing preferences on the top-K tools. An LLM serves as an indirect ranking agent, aligning selections with tool usage patterns, resulting in a context-aware algorithm tailored to downstream tasks.

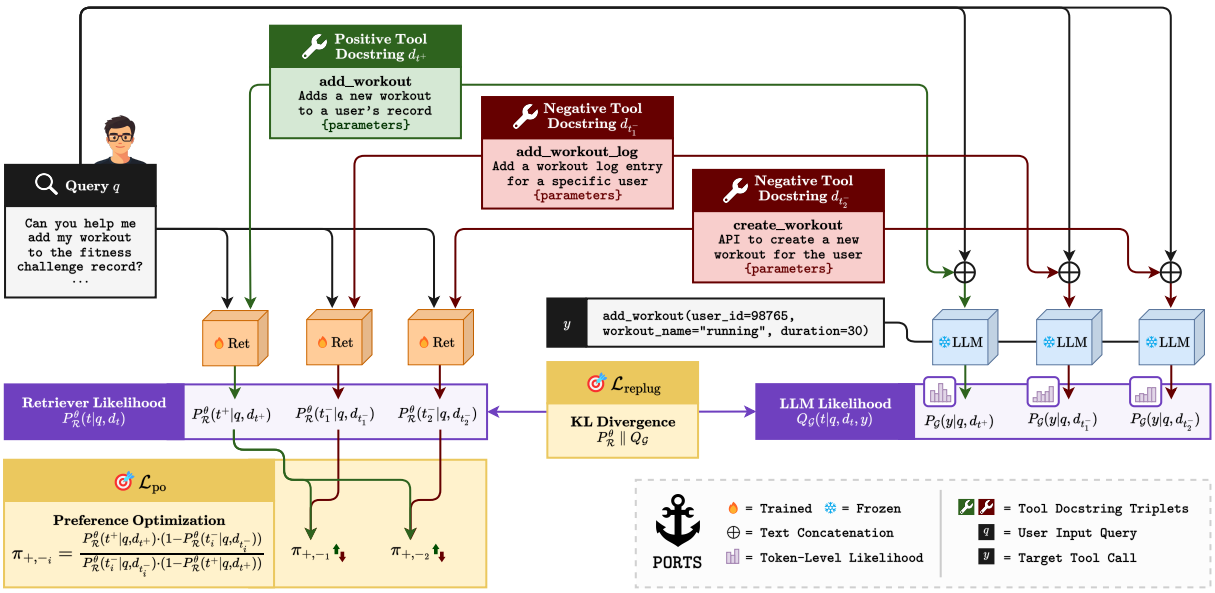


Figure 2: **PORTS training process.** Simplified illustration of the PORTS' training recipe with one positive and two negative tool docstrings from APIBench. Input tool documentation triplets are encoded independently and prompted separately to the frozen LLM. The retriever is fine-tuned to align tool selection probabilities with the correct answer likelihood while maximizing the ratio between the odds of selecting the right tool and the wrong ones.

Goal-Directed Retrieval We formalize the retrieval process using an encoder $\mathcal{E}$ and a generative LLM $\mathcal{G}$. Each instance in the dataset $\mathcal{D}$ comprises a user query q, the target tool call y, the tool required t to solve the request (positive), and a set of n tools irrelevant to the task (negatives). Each tool t_i is associated with a description d_{t_i}, which includes its general characteristics, objectives, and parameters. PORTS first computes the alignment between q and the tool docstrings d_t using a cosine similarity function $sim(q, d_t) = \mathcal{E}_\theta(q) \cdot \mathcal{E}_\theta(d_t)$, where $\mathcal{E} : \mathbb{Z}^l \to \mathbb{R}^d$ denotes the retriever's encoding function that maps input sequences of l tokens to a d-dimensional vector space, parameterized by weights θ. These similarities are then normalized and converted into retrieval probabilities using a softmax function with scaling factor γ:

$$P_{\mathcal{R}}^{\theta}(t|q,d_t) = \frac{\exp\left(\frac{sim(q,d_t)}{\gamma}\right)}{\sum_{t_i \in \mathcal{T}} \exp\left(\frac{sim(q,d_{t_i})}{\gamma}\right)} \quad (1)$$

The retrieval distribution over the corpus of tools in Eq. 1 is approximated by marginalizing over restricted triplet sets $\mathcal{T} = (t_i^+, t_{i,1}^-, \ldots, t_{i,n}^-) \subseteq \mathbf{T}$ for efficiency. As in REPLUG, we prompt each retrieved tool docstring independently with the query and then conduct K separate inferences. This enables a direct and noiseless correlation between tool selection quality and confidence in the generated output. The output probability distribution $Q_{\mathcal{G}}$ of the LLM, reflecting confidence in the final prediction, is computed as in Eq. 2. It applies a softmax function, parameterized by temperature β, to the average log-likelihood $P_{\mathcal{G}}$ of producing the correct tool call y.

$$Q_{\mathcal{G}}(t|q,d_t,y) = \frac{\exp\left(\frac{P_{\mathcal{G}}(y|q,d_t)}{\beta}\right)}{\sum_{t_i \in \mathcal{T}} \exp\left(\frac{P_{\mathcal{G}}(y|q,d_{t_i})}{\beta}\right)}$$
$$P_{\mathcal{G}}(y|q,d_t) = \frac{1}{l}\log \prod_{i=1}^{l} P_{\mathcal{G}}(y_i|q,d_t,y_{<i}) \quad (2)$$

The retriever is trained by optimizing the Kullback-Leibler divergence between the $Q_{\mathcal{G}}(t|q,y)$ and $P_{\mathcal{R}}^{\theta}(t|q)$ distributions:

$$\mathcal{L}_{\text{replug}} = \mathop{\mathbb{E}}_{\mathcal{T},q,y\sim\mathcal{D}} \sum_{t_i \in \mathcal{T}} \text{KL}\left(P_{\mathcal{R}}^{\theta}(t_i|q,d_{t_i}) \parallel Q_{\mathcal{G}}(t_i|q,d_{t_i},y)\right) \quad (3)$$

During training, the model's perplexity—which expresses the confidence in the prediction of the correct tool call—is leveraged by $\mathcal{L}_{\text{replug}}$ in Eq. 3 to encourage the reshape of the retrieval distribution. As a result, the encoder model $\mathcal{E}_\theta$ learns to assign lower ranks to tools that increase the probability of generating incorrect responses.

Preference-Aligned Retrieval We introduce a contrastive loss signal that enforces a policy to favor selecting correct tools over incorrect ones (Eq. 4).

$$\pi_\theta(t|q, d_t) = \frac{P^\theta_\mathcal{R}(t|q, d_t)}{1 - P^\theta_\mathcal{R}(t|q, d_t)} \tag{4}$$

For each positive-negative tool pair $(t_i^+, t_{i,j}^-)$ in the input sample, we compute a ratio of their retrieval probabilities and apply the sigmoid function σ to derive a preference score (i.e., the relative likelihood of selecting one tool over the other).

$$\begin{aligned} \mathcal{L}_{\text{po}} &= -\sum_{i \in 1,n} \log \sigma \left(\log \pi_{+,-_i} \right) \\ \pi_{+,-_i} &= \frac{\pi_\theta(t^+|q, d_{t^+})}{\pi_\theta(t_i^-|q, d_{t_i^-})} \end{aligned} \tag{5}$$

The retriever incurs a penalty through the preference policies π, as indicated in Eq. 5, when it shows an increased likelihood of selecting erroneous tools.

Training Objective PORTS combines the LLM-based retrieval proxy loss ($\mathcal{L}_{\text{replug}}$) and the preference optimization loss ($\mathcal{L}_{\text{po}}$) with a weighting factor λ: $\mathcal{L}_{\texttt{PORTS}} = \mathcal{L}_{\text{replug}} + \lambda \cdot \mathcal{L}_{\text{po}}$. Our method synchronizes the encoder selections with the LLM tool-calling patterns and imposes positive-negative embedding orientation constraints similar to deep metric learning (Kaya and Bilge, 2019).

Tool Triplets and Embeddings Asynchronous Update The learning effectiveness in contrastive approaches is heavily influenced by the choice of negative examples (Karpukhin et al., 2020). We therefore implement a hard-negative sampling technique, choosing the n tools most semantically similar to the query as negative instances, where similarity is quantified using cosine similarity between embeddings computed by the encoder itself during training. To maintain computational efficiency while adapting to shifts in the embedding space during training, we periodically update both the tool embeddings and the selection of hard negatives every T training iterations (Guu et al., 2020).

Motivations for Contrastive Learning We establish that, without fine-tuning, embeddings of tool docstrings are much more concentrated in space than traditional, general-domain text documents–due to recurring data types, keywords, and concise but distinctive functional signatures (see Appendix G). Without targeted supervision, these properties can lead the retriever to rely on superficial cues or converge to trivial matches. Our contrastive loss is designed to counteract this by steering retrieval toward semantically relevant and challenging negatives, promoting fine-grained distinctions beyond what is captured by LLM likelihoods alone. More details about the theoretical foundations of PORTS are in Appendix A.

4 Experimental Setup

4.1 Datasets

We evaluate PORTS on six popular tool-augmented datasets: ToolBench (Qin et al., 2024), API-Bank (Li et al., 2023), APIBench (Peng et al., 2023), BFCL-v2 (Yan et al., 2024), ToolE (Huang et al., 2024), Octopus-v2 (Chen and Li, 2024). This collection offers a heterogeneous testing ground characterized by varying scales, applications, and input modalities. When necessary, we adapt dataset instances to tool selection, which is the core task of our contributions. Extensive dataset documentation is available in Appendix C. Key information and statistics are reported in Table 1. For ToolBench, we focus on the most complex data split, G3, where queries demand the interplay of tools with dissimilar features, functions, and objectives. Only for training, we decompose multi-tool instances from BFCL, API-Bank, and ToolBench into separate examples, each targeting a single tool. When handling conversational inputs, we remove previous tool calls from the chat history of each fragment. We partition test-only benchmarks (Octopus-v2, ToolE, BFCL) into train and test sets using a 70/30 ratio. For Octopus-v2 and ToolE, which focus on single-tool selection without incorporating heterogeneous levels of complexity (e.g., difficulty levels or programming languages), we design in-domain and out-of-domain variants to evaluate generalization abilities to seen and unseen tools. Out-of-domain variants are created with an 80/20 tools split, avoiding overlap between training and test.

4.2 Evaluation Metrics

We quantify the retrieval performance using Recall@K (Zhu, 2004) and NDCG@K (Järvelin and Kekäläinen, 2002), with $K = \{1, 3, 5\}$ following Qin et al. (2024). Recall@K measures the proportion of cases where the positive tool appears in the top-K results, while NDCG@K also considers its ranking position within the top-K.

				Train		Test		All
Dataset*	**Description**	**Source†**	**Input‡**	**# Tools**	**# Instances**	**# Tools**	**# Instances**	**# Tools ↓**
❶ ToolBench	REST APIs; 49 domains (e.g., Social Media, E-Commerce, Weather)	⚙	❓	12,934	486,367	891	1,250	12,934
❷ API-Bank	General tools; 1,000 domains	⚙	💬	1,896	6,703	67	620	1,960
❸ APIBench	Tools for Java and Python programming; 90 domains	👤	❓/</>	188	15,218	152	188	1,557
❹ BFCL-v2 »	Python and Non-Python tools; 40 domains (e.g., Computing, Mathematics, Sports, Finance)	👤/⚙	❓/💬	781	1,260	415	541	1,015
❺ ToolE »	Tools inspired to OpenAI plugins; 6 main scenarios (Software, Utilities, Finance, Home, Education, Arts)	⚙	❓	199	16,491	199	4,123	199
❻ Octopus-v2 »	Android APIs (System, App, Smart Device Management)	⚙	❓	20	160	20	40	20
❼ 👁 ToolE »	Same as ❺			160	16,406	39	4,208	199
❽ 👁 Octopus-v2 »	Same as ❻			16	160	4	40	20

* » = We split test-only benchmarks with a 70/30 ratio; 👁 = Out-of-domain versions (no overlapping between train and test tools).
† 👤 = Human-sourced (manual or scraped); ⚙ = LLM-generated (reviewed or not). ‡ ❓ = Query, 💬 = Chat, </> = Code.

Table 1: **Summary of tool selection datasets**, sorted by descending total tool count.

4.3 Implementation Details

Models For the embedding function of $\mathcal{R}$, we test two prominent encoders: the 125M-parameter RoBERTa-base (Liu et al., 2019), a widely adopted baseline, and the 109M-parameter BGE-base (Xiao et al., 2024), a top performer on the MTEB leaderboard.[3] We evaluate each encoder's architectural compatibility and its impact on retrieval precision. For $\mathcal{G}$, we examine three open-source LLMs. (1) CODESTRAL-22B-v0.1,[4] a model specialized for code generation tasks with native tool calling, motivated by (Nexusflow.ai, 2023). (2) LLAMA3-8B (Dubey et al., 2024), a foundational model, and (3) LLAMA3-GROQ-8B-Tool-Use,[5] its fine-tuned variant specifically designed for advanced tool use. We employ 4-bit quantization for efficiency. Larger models were excluded due to computational limits, focusing on a selection balancing performance and efficiency. Prompt templates are in Appendix B.

Hyperparameters In our experiments, we used 3 negatives sampled every $T = 50$ training steps upon recalculating embeddings. We adopted training and evaluation batch sizes of 2 and 4, respectively, and set maximum sequence lengths of 512 and 1024 for the encoder and LLM inputs. We applied a loss weight λ of 0.3, with γ and β set to 0.5. We set the random seed to 42 for reproducibility and trained each configuration for 2 epochs using the AdamW optimizer, coupled with a cosine learning rate scheduler starting at $1e^{-5}$. We kept LLMs frozen during training. To simulate low-resource scenarios, we sampled 10K unique instances per dataset, testing PORTS' capacity to exploit few information more efficiently. Search spaces and hyperparameter insights are in Appendix D.

Hardware Setup Each run was performed on an internal workstation using a single Nvidia GeForce RTX3090 GPU with 24GB of dedicated memory, 64GB of RAM, and an Intel® Core™ i9-10900X1080 CPU @ 3.70GHz. The reference operating system is Ubuntu 20.04.3 LTS.

5 Results

We evaluate PORTS against REPLUG, our primary baseline for goal-driven retrieval. Other training methodologies are omitted from direct comparison due to divergent optimization goals and architectural assumptions. Our core findings are in Table 2, showing metric scores for all models, datasets, and a comparison of different losses for ablation. Given space constraints, we list tool selection outcomes only for the top-performing LLM in each encoder-dataset-loss configuration. We refer the reader to Appendix H for exhaustive scores. We observe that PORTS consistently elevates baseline effectiveness in all datasets (cf. the color gradients for Δ_{avg} columns linked to $\mathcal{L}_{\texttt{PORTS}}$ entries in Table 2). For seen tools, PORTS yields substantial gains, boosting average Recall@$\{1, 2, 3\}$ by up to 71.66 percentage points and average NDCG@$\{1, 3, 5\}$ by 70.16. Even with unseen tools, the improvements remain remarkable, reaching +61.24 and +59.79 points in Recall and NDCG, respectively.

[3] huggingface.co/spaces/mteb/leaderboard
[4] huggingface.co/mistralai/Codestral-22B-v0.1
[5] huggingface.co/Groq/Llama-3-Groq-8B-Tool-Use

Encoder	Dataset	Method	Best LLM	Recall (%) @1	Recall (%) @2	Recall (%) @3	NDCG (%) @1	NDCG (%) @3	NDCG (%) @5	Δ_{avg} Baseline$^{\diamond}$ Recall	Δ_{avg} Baseline$^{\diamond}$ NDCG
RoBERTa	❶$_{\dagger}$	$\mathcal{L}_{\text{PORTS}}$	CODESTRAL-22B-v0.1	18.23	21.10	25.60	18.23	20.28	24.31	20.63	19.94
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	12.56	20.11	24.80	12.56	19.67	22.19	18.14	17.14
	❷	$\mathcal{L}_{\text{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	49.84	64.35	70.80	49.84	62.27	65.32	57.10	54.14
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	45.32	61.94	68.23	45.32	59.95	62.44	53.93	50.90
	❸	$\mathcal{L}_{\text{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	21.50	27.40	30.53	21.50	25.22	26.78	25.94	23.50
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	8.74	12.61	15.35	8.74	12.55	14.56	10.70	10.95
	❹	$\mathcal{L}_{\text{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	58.12	69.21	73.52	58.12	68.38	69.22	60.30	59.24
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	53.97	64.88	68.39	53.97	62.61	64.93	55.76	54.50
	❺	$\mathcal{L}_{\text{PORTS}}$	LLAMA3-8B	60.33	72.45	77.15	60.33	70.29	72.34	57.51	55.65
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	49.38	59.84	64.23	49.38	58.17	60.79	45.35	44.11
	❻	$\mathcal{L}_{\text{PORTS}}$	LLAMA3-8B	95.00	100	100	95	95.25	98.25	71.66	70.16
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	87.50	97.50	100	87.50	95.06	95.06	68.33	66.54
	❼	$\mathcal{L}_{\text{PORTS}}$	LLAMA3-8B	74.60	83.90	86.80	74.60	81.23	83.55	61.24	59.79
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	58.51	68.54	74.10	58.51	67.62	69.78	46.53	45.30
	❽	$\mathcal{L}_{\text{PORTS}}$	LLAMA3-8B	96.00	100	100	96.00	98.22	98.22*	23.89	24.48
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	80.00	100	100	80.00	92.62	92.62*	19.16	15.41
BGE	❶$_{\dagger}$	$\mathcal{L}_{\text{PORTS}}$	CODESTRAL-22B-v0.1	25.80	36.05	43.35	25.80	35.50	42.20	8.71	9.50
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	22.56	33.74	40.51	22.56	33.00	36.65	5.92	5.73
	❷	$\mathcal{L}_{\text{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	59.12	76.80	81.50	59.12	75.40	76.10	14.94	14.21
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	56.29	75.00	80.00	56.29	70.60	73.32	12.90	10.73
	❸	$\mathcal{L}_{\text{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	30.64	37.20	41.06	30.64	33.90	35.20	20.18	17.26
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	19.55	28.29	33.05	19.55	27.45	30.02	10.84	10.67
	❹	$\mathcal{L}_{\text{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	67.20	73.23	78.10	67.20	74.60	73.10	5.31	5.63
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	66.17	73.20	77.82	66.17	72.92	74.31	4.86	5.13
	❺	$\mathcal{L}_{\text{PORTS}}$	CODESTRAL-22B-v0.1	67.35	79.48	83.75	67.35	77.00	78.00	14.71	14.12
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	67.23	77.54	81.06	67.23	75.50	76.82	13.12	13.18
	❻	$\mathcal{L}_{\text{PORTS}}$	LLAMA3-8B	97.50	100	100	97.50	100	100	1.67	2.17
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	95.00	100	100	95.00	98.00	98.00	0.83	0.22
	❼	$\mathcal{L}_{\text{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	89.87	92.20	94.04	89.87	91.10	92.35	14.18	17.11
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	74.60	86.24	90.30	74.60	83.98	85.09	5.85	7.22
	❽	$\mathcal{L}_{\text{PORTS}}$	LLAMA3-8B	97.50	100	100	97.50	100	100*	0.84	1.17
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	95.00	100	100	95.00	98.00	98.00*	0	0

† Results computed on the G3 split. * NDCG@4 since the out-of-domain version of Octopus-v2 has 4 tools only.
$^{\diamond}$ Δ_{avg} measures the average percentage point improvement across all ranks @K.

Table 2: **PORTS Recall@K and NDCG@K per encoder-dataset-loss (test set).** Reported results refer to the best LLM for each triplet. The positive gains in metric scores over the baselines are highlighted (the brighter, the better).

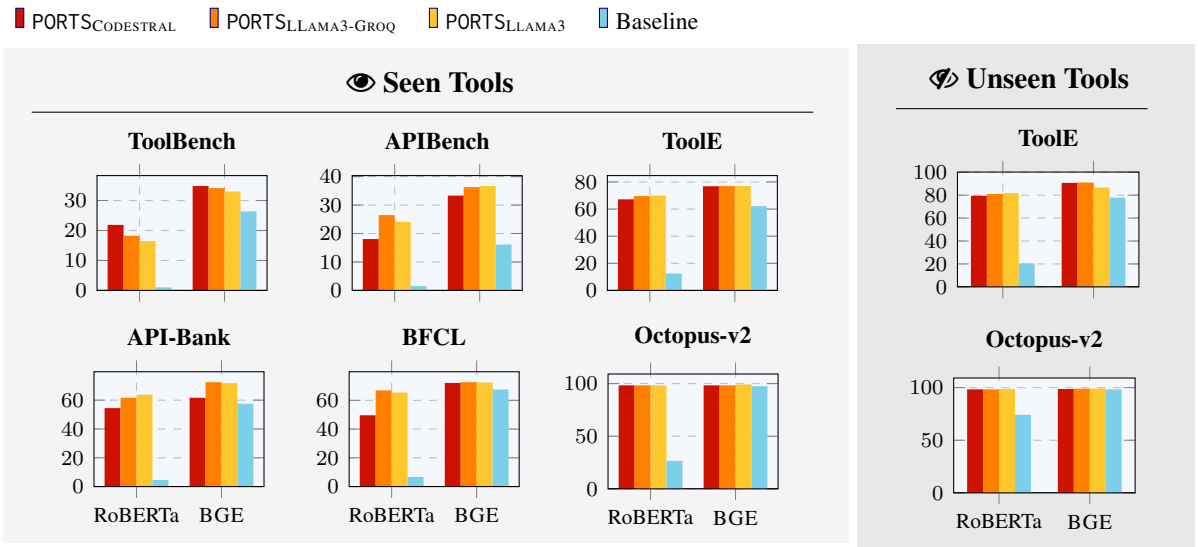


Figure 3: **Average Recall@K for each dataset (test set).** Effectiveness of PORTS-tuned retrievers ($\mathcal{L}_{\text{PORTS}}$) against frozen baselines, utilizing guidance from different LLMs. Evaluated in both in-domain and out-of-domain settings.

Baseline Impact Both RoBERTa and BGE fine-tuned with PORTS significantly outperform their respective base models, demonstrating the broad applicability of our framework. RoBERTa exhibits a more pronounced response to PORTS (Δ_{avg} Recall of 47.28 compared to BGE's 10.07), indicating that simpler and less specialized retrievers are more adaptable. Table 2 clearly illustrate that BGE begins with a notable disparity between positive and negative instances (average frozen Recall of 62.92 vs. RoBERTa's 18.45). This pre-existing imbalance in the retrieval distribution introduces a skew, which in turn attenuates the strength of our loss signal. Further robustness tests are reported in Appendix F.

LLM Impact The use of an LLM as a proxy during training requires careful scrutiny. Although the primary focus is not the direct accuracy of the downstream generative task, the distribution of perplexities between different generations raises questions about the optimal LLM to maximize learning influence. Empirical data did not elucidate a definitive model preference but instead accentuated PORTS' potential to transcend its intended tool usage scope. The performance delta across all experiments ranged from +1 to +4 percentage points, with peaks attaining a +13 average recall improvement. LLAMA3-8B demonstrated superior performance when paired with RoBERTa, where–as previously discussed–there was markedly less resistance to adaptation. This improvement can be attributed to the heightened uncertainty this model experiences in technical domains not anticipated during its pre-training regimen. Although elevated perplexity is generally undesirable for downstream task optimization, it is advantageous in the PORTS framework due to the resultant richer and less skewed log-likelihood distribution, as elucidated in Eq. 2.

Docstring Impact The efficacy of contrastive retrieval is heavily contingent on docstring quality. APIBench illustrates this relationship, where vague descriptions result in smaller changes in the tool retrieval metrics. In fact, the tool documents in the dataset reference generic features of pre-trained models from HuggingFace, which often omit parameter names and types that could better guide the retriever at training time. Importantly, LLAMA3-8B tends to perform better in datasets with less technical tool documentation due to its superior management of perplexity scores. On the other hand, LLAMA3-GROQ-8B-Tool-Use and CODESTRAL-22B-v0.1, with their advanced tool usage capabilities, excel in datasets with more detailed docstrings (e.g., arguments, outputs, types, defaults), namely ToolBench and Octopus-v2.

Training Loss Contrastive learning and REPLUG have known simultaneous success in various application domains. Our assertions on the positive impact of preference-oriented learning have been substantiated through comprehensive ablations presented in Table 2. PORTS outperforms REPLUG in all evaluated domains, with substantial disparities in recall performance, with +15.8 and +12.3 in ToolE and ApiBench, respectively. We underline that these datasets present unique challenges: ToolE necessitates precise tool decisions among similar options in complex scenarios, while ApiBench involves code generation primarily through pre-trained neural networks, described only in broad application contexts. Such unique characteristics are conducive to showcase the benefits of our comparative loss approach, which likely contributes to PORTS's superior performance in these settings. Zooming out, the statistics outlined in Figure 1 reflect an average Recall@K improvement of 6.7 (RoBERTa) and 3.4 (BGE) across all datasets and LLMs, ultimately corroborating the greater efficacy of our method over REPLUG in the context of tool retrieval.

Out-Of-Domain Analysis For domains with limited resources and scarce data, retrieval systems must be capable of effectively managing unfamiliar tools. Although previous research has developed systems that can adapt to new data (Gao et al., 2024), these systems are susceptible to overfitting, potentially due to biases in the distribution and usage patterns of tools. To evaluate the robustness of our method in addressing these challenges, we investigated the performance of retrieval models trained with REPLUG and PORTS when exposed to varying proportions of unseen tools from the ToolE dataset. Starting with a 90/10 ratio of seen to unseen tools, we progressively reduced the training dataset and assessed the performance of RoBERTa guided by LLAMA3-8B on a consistent test distribution. Figure 4 illustrates the superior generalization capabilities of PORTS. Employing preference optimization loss, our contrastive learning techniques effectively derive semantic insights into query-tool interactions without requiring extensive pairwise comparisons, thereby substantiating the enhanced low-resource capabilities of our solution

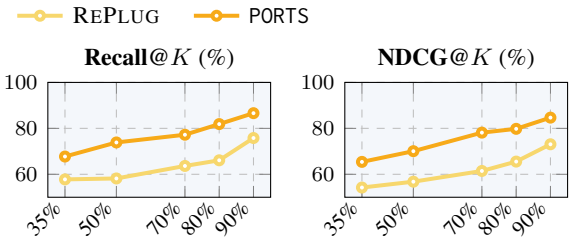


Figure 4: **Average out-of-domain Recall@K and NDGC@K with a progressive number of train tools (decreasing unseen).** Reported results refer to RoBERTa on ToolE with REPLUG and PORTS under LLAMA3-8B supervision.

and demonstrating its suitability for application areas with constantly evolving API documentation. Figure 3 reports the average recall across test datasets and encoder models for all PORTS' variants fine-tuned with different LLMs.

6 Conclusion

We introduce PORTS, a novel training method to optimize encoders for tool retrieval tasks. Our goal is twofold: to align tool selection with the preferences of the calling LLM, and to maximize the odds ratio between correct and incorrect tools. PORTS emphasizes low cost by leveraging LLMs' prior knowledge to navigate the latent space of tool document similarities, focusing on the impact of retrieved samples. Experiments across multiple models and diverse datasets show that PORTS achieves Recall@1 improvements of up to **+72.5%** and **+58.7%** over frozen baselines for in- and out-domain tools, with gains of **+15.24%** and **+14.71%** percentage points compared to REPLUG.

To further assess retrieval effectiveness in code generation scenarios, metrics like PASS@K could reflect the downstream impact on generative components. Incorporating relevance signals based on actual output effects and message similarity may allow PORTS to integrate discounting mechanisms for more goal-directed retrieval. Future work could also investigate PORTS in biomedical discovery workflows (Wang et al., 2025)–e.g. helping an agent decide whether to use specialized function discovery models (Domeniconi et al., 2016, 2014a; di Lena et al., 2015), call Gene Ontology tooling for term enrichment, query STRING for protein-protein interaction networks, or fetch data from the Unified Medical Language System–with dozens to hundreds of specialized data sources and tools with different formats, coverage, and update frequency.

Acknowledgment

Research partially supported by: AI-PACT (CUP B47H22004450008, B47H22004460001); National Plan PNC-I.1 DARE (PNC0000002, CUP B53C22006450001); PNRR Extended Partnership FAIR (PE00000013, Spoke 8); 2024 Scientific Research and High Technology Program, project "AI analysis for risk assessment of empty lymph nodes in endometrial cancer surgery", the Fondazione Cassa di Risparmio in Bologna; Chips JU TRISTAN (G.A. 101095947). LG Solution Srl for partially funding a PhD scholarship to L. Molfetta.

Limitations

Despite its strong results, PORTS has limitations that warrant further examination. First, its effectiveness is sensitive to the quality of tool documentation, with diminished gains in domains where docstrings are vague or underspecified. Second, although PORTS maintains a memory efficiency comparable to REPLUG LSR, it requires repeated querying of a frozen LLM to calculate the guidance signals. This dependence introduces additional computational overhead in both time and memory, potentially limiting scalability in resource-constrained settings. Future work may address these challenges by reducing the reliance on LLM inference or developing efficient approximations of guidance signals, thus improving the practicality of retrieval methods that use LLMs as proxies at training time.

References

Ibrahim Abdelaziz, Kinjal Basu, Mayank Agarwal, and 1 others. 2024. Granite-function calling model: Introducing function calling abilities via multi-task learning of granular tasks. *CoRR*, abs/2407.00121.

Raviteja Anantha, Bortik Bandyopadhyay, Anirudh Kashi, and 1 others. 2023. Protip: Progressive tool retrieval improves planning. *CoRR*, abs/2312.10332.

Daniel Bolya, Po-Yao Huang, and Peize Sun et al. 2025. Perception encoder: The best visual embeddings are not at the output of the network. *CoRR*, abs/2504.13181.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, and 1 others. 2022. Improving language models by retrieving from trillions of tokens. In *ICML*, volume 162 of *PMLR*, pages 2206–2240. PMLR.

Wei Chen and Zhiyuan Li. 2024. Octopus v2: On-device language model for super agent. *CoRR*, abs/2404.01744.

Xin Cheng, Di Luo, Xiuying Chen, and 1 others. 2023. Lift yourself up: Retrieval-augmented text generation with self-memory. In *NeurIPS*.

Paul F. Christiano, Jan Leike, Tom B. Brown, and 1 others. 2017. Deep reinforcement learning from human preferences. In *NeurIPS*, pages 4299–4307.

Pietro di Lena, Giacomo Domeniconi, Luciano Margara, and Gianluca Moro. 2015. GOTA: GO term annotation of biomedical literature. *BMC Bioinform.*, 16:346:1–346:13.

Giacomo Domeniconi, Marco Masseroli, Gianluca Moro, and Pietro Pinoli. 2014a. Discovering new gene functionalities from random perturbations of known gene ontological annotations. In *KDIR 2014*, pages 107–116. SciTePress.

Giacomo Domeniconi, Marco Masseroli, Gianluca Moro, and Pietro Pinoli. 2016. Cross-organism learning method to discover new gene functionalities. *Comput. Methods Programs Biomed.*, 126:20–34.

Giacomo Domeniconi, Gianluca Moro, Andrea Pagliarani, and Roberto Pasolini. 2015. Markov chain based method for in-domain and cross-domain sentiment classification. In *KDIR 2015*, pages 127–137. SciTePress.

Giacomo Domeniconi, Gianluca Moro, Roberto Pasolini, and Claudio Sartori. 2014b. Cross-domain text classification through iterative refining of target categories representations. In *KDIR 2014*, pages 31–42. SciTePress.

Giacomo Domeniconi, Gianluca Moro, Roberto Pasolini, and Claudio Sartori. 2014c. Iterative refining of category profiles for nearest centroid cross-domain text classification. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management, 2014*, volume 553 of *Communications in Computer and Information Science*, pages 50–67. Springer.

Abhimanyu Dubey, Abhinav Jauhri, and Abhinav Pandey et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Shen Gao, Zhengliang Shi, Minghang Zhu, and 1 others. 2024. Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum. In *AAAI*, pages 18030–18038. AAAI Press.

Kelvin Guu, Kenton Lee, Zora Tung, and 1 others. 2020. Retrieval augmented language model pre-training. In *ICML*, volume 119 of *PMLR*, pages 3929–3938. PMLR.

Shibo Hao, Tianyang Liu, Zhen Wang, and 1 others. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. In *NeurIPS*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. ORPO: monolithic preference optimization without reference model. *CoRR*, abs/2403.07691.

Yue Huang, Jiawen Shi, Yuan Li, and 1 others. 2024. Metatool benchmark for large language models: Deciding whether to use tools and which to use. In *ICLR*. OpenReview.net.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, pages 874–880. ACL.

Gautier Izacard, Patrick S. H. Lewis, Maria Lomeli, and 1 others. 2023. Atlas: Few-shot learning with retrieval augmented language models. *JMLR*, 24:251:1–251:43.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. In *EMNLP-IJCNLP 2019*, pages 2567–2577. Association for Computational Linguistics.

Karen Spärck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 60(5):493–502.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, and 1 others. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*, pages 6769–6781. ACL.

Mahmut Kaya and Hasan Sakir Bilge. 2019. Deep metric learning: A survey. *Symmetry*, 11(9):1066.

Yilun Kong, Jingqing Ruan, Yihong Chen, and 1 others. 2023. Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world systems. *CoRR*, abs/2311.11315.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and 1 others. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *CoRR*, abs/2203.05115.

Mingda Li, Xinyu Li, Yifan Chen, and 1 others. 2024. Unraveling and mitigating retriever inconsistencies in retrieval-augmented large language models. *CoRR*, abs/2405.20680.

Minghao Li, Yingxiu Zhao, Bowen Yu, and 1 others. 2023. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *EMNLP*, pages 3102–3116, Singapore. ACL.

Xi Victoria Lin, Xilun Chen, Mingda Chen, and 1 others. 2024. RA-DIT: retrieval-augmented dual instruction tuning. In *ICLR*. OpenReview.net.

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

Yinhan Liu, Myle Ott, Naman Goyal, and 1 others. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Stefano Lodi, Gianluca Moro, and Claudio Sartori. 2010. Distributed data clustering in multi-dimensional peer-to-peer networks. In *Database Technologies 2010, Twenty-First Australasian Database Conference (ADC 2010)*, volume 104 of *CRPIT*, pages 171–178. Australian Computer Society.

Nexusflow.ai. 2023. Nexusraven: Surpassing the state-of-the-art in open-source function calling llms.

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. *CoRR*, abs/1611.09268.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. TALM: tool augmented language models. *CoRR*, abs/2205.12255.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and 1 others. 2023. Gorilla: Large language model connected with massive apis. *CoRR*, abs/2305.15334.

Yun Peng, Shuqing Li, Wenwei Gu, and 1 others. 2023. Revisiting, benchmarking and exploring API recommendation: How far are we? *IEEE Trans. Software Eng.*, 49(4):1876–1897.

Shuofei Qiao, Honghao Gui, Huajun Chen, and 1 others. 2023. Making language models better tool learners with execution feedback. *CoRR*, abs/2305.13068.

Yujia Qin, Shihao Liang, Yining Ye, and 1 others. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *ICLR*. OpenReview.net.

Changle Qu, Sunhao Dai, Xiaochi Wei, and 1 others. 2024a. COLT: towards completeness-oriented tool retrieval for large language models. *CoRR*, abs/2405.16089.

Changle Qu, Sunhao Dai, Xiaochi Wei, and 1 others. 2024b. Tool learning with large language models: A survey. *CoRR*, abs/2405.17935.

Rafael Rafailov, Archit Sharma, Eric Mitchell, and 1 others. 2023. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, and 1 others. 2023. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*.

Yongliang Shen, Kaitao Song, Xu Tan, and 1 others. 2023. Hugginggpt: Solving AI tasks with chatgpt and its friends in hugging face. In *NeurIPS*.

Freda Shi, Xinyun Chen, Kanishka Misra, and 1 others. 2023. Large language models can be easily distracted by irrelevant context. In *ICML*, volume 202 of *PMLR*, pages 31210–31227. PMLR.

Weijia Shi, Sewon Min, Michihiro Yasunaga, and 1 others. 2024. REPLUG: Retrieval-augmented black-box language models. In *NAACL*, pages 8371–8384, Mexico City, Mexico. ACL.

Shamane Siriwardhana, Rivindu Weerasekera, Tharindu Kaluarachchi, and 1 others. 2023. Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering. *TACL*, 11:1–17.

Yifan Song, Weimin Xiong, Dawei Zhu, and 1 others. 2023. Restgpt: Connecting large language models with real-world applications via restful apis. *CoRR*, abs/2306.06624.

Wenxuan Wang, Zizhan Ma, Zheng Wang, Chenghan Wu, Jiaming Ji, Wenting Chen, Xiang Li, and Yixuan Yuan. 2025. A survey of llm-based agents in medicine: How far are we from baymax? In *ACL 2025*, pages 10345–10359. Association for Computational Linguistics.

Shitao Xiao, Zheng Liu, Peitian Zhang, and 1 others. 2024. C-pack: Packed resources for general chinese embeddings. In *SIGIR*, pages 641–649. ACM.

Fanjia Yan, Huanzhi Mao, Charlie Cheng-Jie Ji, and 1 others. 2024. Berkeley function calling leaderboard. `https://gorilla.cs.berkeley.edu/blogs/8_berkeley_function_calling_leaderboard.html`.

Rui Yang, Lin Song, Yanwei Li, and 1 others. 2023. Gpt4tools: Teaching large language model to use tools via self-instruction. In *NeurIPS*.

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 *EMNLP, 2018*, pages 2369–2380. Association for Computational Linguistics.

Shunyu Yao, Howard Chen, John Yang, and 1 others. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *NeurIPS*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, and 1 others. 2023. React: Synergizing reasoning and acting in language models. In *ICLR*. OpenReview.net.

Wenhao Yu, Hongming Zhang, Xiaoman Pan, and 1 others. 2023. Chain-of-note: Enhancing robustness in retrieval-augmented language models. *CoRR*, abs/2311.09210.

Lifan Yuan, Yangyi Chen, Xingyao Wang, and 1 others. 2024. CRAFT: customizing llms by creating and retrieving from specialized toolsets. In *ICLR*. OpenReview.net.

Yuanhang Zheng, Peng Li, Wei Liu, and 1 others. 2024. ToolRerank: Adaptive and hierarchy-aware reranking for tool retrieval. In *LREC-COLING 2024*, pages 16263–16273, Torino, Italia. ELRA and ICCL.

Mu Zhu. 2004. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2(30):6.

A In-Depth View on the Relevance of PORTS

Given an input query q and a set of tools t with description d_{t_i}, PORTS combines REPLUG's goal-directed retrieval with preference alignment, adapted explicitly for tool selection. This approach enhances both relevance and downstream performance, effectively addressing gaps in existing methods. The foundation lies in REPLUG's gradient structure, derived from minimizing the KL divergence between the retriever distribution $P_{\mathcal{R}}(t|q, d_t)$ and the LM's utility signal $Q_{\mathcal{G}}(t|q, d_t, y)$.

RePlug Gradient Derivation Starting from the KL divergence objective:

$$\begin{aligned}\mathcal{L}_{\text{replug}} &= \text{KL}(P_{\mathcal{R}}^{\theta} \parallel Q_{\mathcal{G}}) \\ &= \sum_t P_{\mathcal{R}}^{\theta}(t|q, d_t) \log \frac{P_{\mathcal{R}}^{\theta}(t|q, d_t)}{Q_{\mathcal{G}}(t|q, d_t, y)}, \quad (6)\end{aligned}$$

we decompose it into entropy and cross-entropy terms:

$$\text{KL} = \underbrace{\sum_t P_{\mathcal{R}}^{\theta} \log P_{\mathcal{R}}^{\theta}}_{-H(P_{\mathcal{R}}^{\theta})} - \underbrace{\sum_t P_{\mathcal{R}}^{\theta} \log Q_{\mathcal{G}}}_{-H(P_{\mathcal{R}}^{\theta}, Q_{\mathcal{G}})}. \quad (7)$$

To derive gradients with respect to retriever scores $sim(q, d_t)$, we differentiate both terms. First, for the entropy term $H(P_{\mathcal{R}}^{\theta})$:

$$\begin{aligned}\frac{\partial H(P_{\mathcal{R}}^{\theta})}{\partial sim(q, d_t)} &= \frac{1}{\gamma} P_{\mathcal{R}}^{\theta}(t|q, d_t) \cdot \\ &\cdot \Big(1 + \log P_{\mathcal{R}}^{\theta}(t|q, d_t) - H(P_{\mathcal{R}}^{\theta})\Big), \quad (8)\end{aligned}$$

where γ is the retriever's temperature parameter. For the cross-entropy term $H(P_{\mathcal{R}}^{\theta}, Q_{\mathcal{G}})$:

$$\begin{aligned}\frac{\partial H(P_{\mathcal{R}}^{\theta}, Q_{\mathcal{G}})}{\partial sim(q, d_t)} &= \frac{1}{\gamma} P_{\mathcal{R}}^{\theta}(t|q, d_t) \cdot \\ &\cdot \Big(\log Q_{\mathcal{G}}(t|q, d_t, y) - \mathbb{E}_{P_{\mathcal{R}}^{\theta}}[\log Q_{\mathcal{G}}]\Big). \quad (9)\end{aligned}$$

Subtracting these gradients yields:

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{replug}}}{\partial sim(q, d_t)} &= \frac{P_{\mathcal{R}}^{\theta}(t|q, d_t)}{\gamma} \cdot \\ &\cdot \left(\log \frac{P_{\mathcal{R}}^{\theta}(t|q, d_t)}{Q_{\mathcal{G}}(t|q, d_t, y)} - \text{KL}(P_{\mathcal{R}}^{\theta} \parallel Q_{\mathcal{G}})\right). \quad (10)\end{aligned}$$

Here, $\log \frac{P_{\mathcal{R}}^{\theta}}{Q_{\mathcal{G}}}$ provides per-document alignment signals, while the KL term stabilizes training by serving as a baseline for global distribution shifts.

PORTS Gradient Extension Our analysis of the full PORTS loss gradient highlights the limitations of REPLUG alone:

$$\begin{aligned}\nabla_{\theta}\mathcal{L}_{\text{PORTS}} &\propto \nabla_{\theta}\mathcal{L}_{\text{replug}} + \nabla_{\theta}\mathcal{L}_{\text{po}} \\ \nabla\mathcal{L}_{\text{replug}} &= P_{\mathcal{R}}^{\theta}(t|q, d_t)\Bigg(\log \frac{P_R(t|q, d_t)}{Q_{\mathcal{G}}(t|q, d_t, y)} + \\ &\qquad - \text{KL}(P_{\mathcal{R}}^{\theta} \parallel Q_{\mathcal{G}})\Bigg) \\ \nabla_{\theta}\mathcal{L}_{\text{po}} &= \left(1 + \frac{\pi_{\theta}(t|q, d_{t^+})}{\pi_{\theta}(t|q, d_{t^-})}\right)^{-1}. \quad (11)\end{aligned}$$

While $P_{\mathcal{R}}^{\theta}/Q$ compares the retriever confidence against the LLM's docstring utility assessment, the KL term acts as a global stabilizer that prevents over-adjustments to individual docstrings. Without our preference optimization component, the system simply minimizes differences between retrieval probabilities $P_{\mathcal{R}}^{\theta}$ and downstream log-likelihoods $\mathcal{Q}_{\mathcal{G}}$, overlooking LLM handling of imperfect descriptions. Specifically, pure REPLUG gradients lack awareness of *relative tool utility* – a critical shortfall when multiple tools have overlapping or ambiguous descriptions.

The preference optimization contrastive term addresses this by enforcing refinement of tool selection through pairwise comparisons. Unlike the KL penalty, which operates globally, the term

$$\left(1 + \frac{\pi_{\theta}(t|q, d_{t^+})}{\pi_{\theta}(t|q, d_{t^-})}\right)^{-1}$$

explicitly rewards the retriever for distinguishing between *semantically similar but functionally distinct* tools (d_{t^+} and d_{t^-}). This creates implicit links between tools based on their downstream task performance rather than surface-level similarity. For example, when two API descriptions share terminology but differ in required parameters, the contrastive component amplifies gradients for the tool whose documentation better resolves this ambiguity in practice.

In so doing, PORTS benefits from negative examples not only through the downstream task signal (which can be noisy due to LLM approximation errors) but also through structured semantic comparisons. The retriever learns to associate subtle

linguistic cues in tool descriptions with their *functional outcomes*, even when $Q_{\mathcal{G}}$ provides imperfect supervision. This dual mechanism proves critical in real-world scenarios where tool documentation quality varies widely – the contrastive term compensates for sparse or ambiguous $\mathcal{Q}_{\mathcal{G}}$ signals by reinforcing discriminative features across the toolset. PORTS' triplet formulation also enables flexible, state-independent negative sampling, unlike REPLUG's iterative sampling, which may introduce bias. A qualitative example of PORTS' retrieval and disambiguation capabilities is shown in Appendix I.

B Prompt Templates

The prompt template for Large Language Models (LLMs) is typically divided into two distinct components: a system message and a user instruction. The system message serves to establish the model's role and behavioral parameters. In contrast, the user instruction delineates the specific task or query to be addressed. In cases where a model's chat template does not inherently accommodate a discrete system message, this information is instead prepended to the user instruction. This ensures that the model is primed with all necessary contextual and behavioral guidelines before processing the task at hand. Within the PORTS framework, a frozen LLM is prompted with the input query and the docstring of a retrieved tool to gauge the probability of predicting the target call. During training, we mask the input up to the "Answer" tag and compute the next-token probability of the gold answer using the system and instruction sections as input with a causal attention approach. Our prompt templates are reported in Listing 5 and Listing 6. To better recall the prior knowledge of the model, we describe tools as API functions. We use each model's specific chat template, omitting special tokens in the listings for clarity. We use a different prompt when working with LLAMA3-8B to better adhere to the chat template on which it was trained, as suggested in the HuggingFace model card.[6]

C Software and Datasets: Details, Intended Use, and Impact

Despite the popularity of REPLUG, no implementation code was publicly available. **We dedicated significant effort to reconstructing the method from scratch**, carefully clarifying its methodological choices. To benefit the broader research community, **we release our complete implementation–including PORTS–as fully open-source under a permissive MIT license**. This ensures full reproducibility and establishes the first open-source solution for goal-directed encoder fine-tuning.

Queries, tools, and docstrings can vary greatly depending on the dataset. Table 3, Table 4, Table 5, Table 6, Table 7, and Table 8 show representative input-output examples sampled from the test set of each dataset. Each dataset has been pre-processed to ensure compatibility with our tool-selection task by decoupling instances that require the use of multiple tools. Tool descriptions have been enhanced with detailed information about input and output parameter types, formats, and purposes, following Python-style docstrings to better define the scope of each tool and facilitate the retrieval process. Given the conversational and multi-tool nature of the API-Bank dataset, we have distinguished between inputs for the retriever and generative models. The retriever's input excludes previous tool calls to avoid biases and inconsistencies in the similarity-based search, while generative models receive the full conversation history. This approach enhances tool selection accuracy while allowing generative models to leverage complete contextual information.

Licenses of Used Datasets All datasets used in our experiments are publicly available and released under permissive open-source licenses. Specifically, TOOLBENCH, APIBENCH, OCTOPUS, and BFCLV2 are distributed under the Apache 2.0 License, while TOOLE and APIBANK are released under the MIT License. These licenses allow for both academic and commercial use, ensuring full compliance with open-source standards and enabling reproducibility of our experiments.

D Hyperparameter Space

Table 9 presents a comprehensive overview of the hyperparameters explored in our study. This extensive search space was designed to optimize the model's performance across various dimensions, from basic configuration settings to more nuanced training parameters.

[6] huggingface.co/Groq/Llama-3-Groq-8B-Tool-Use

Prompt Template for Instruct LLMs

System Message

```
You are a function caller. You are given a user query and the description (docstring) of a single API function.

You must generate a function call using the exact name and parameters of the provided API function. You are not allowed to use any other function besides the one given.

Return only the function call, using single quotes for strings and separating parameters with commas.

You are not permitted to deviate from the given API function in any way. You must use the exact function name and parameter types specified, even if you think another function would be more appropriate for the user's request.

Example:
==============
Docstring:
def add_reminder(text: str, date: str, time: str):
    """
    Description:
    Set a reminder for a task on a specified date and time.

    Arguments:
    ——
    - text : str
        The description or name of the task for which the reminder is
        being set.
    - date : str
        The date on which the reminder should be scheduled.
    - time : str
        The time at which the reminder should be scheduled.
    """

Query:
"Add a reminder to buy groceries tomorrow at 2 PM"
Answer:
add_reminder(
  text='Buy groceries',
  date='tomorrow',
  time='2 PM'
)
==============
```

Instruction Message

```
Docstring: ${Docstring}
Query: ${Query}
Answer: ${Answer}
```

Figure 5: Prompt template for call generation with the retrieved tool for GEMMA3-1B, QWEN3-4B, LLAMA3.2-3B, LLAMA3-8B, and CODESTRAL-22B-v0.1.

Prompt Template for Tool LLMs

System Message

```
You are a function caller. You are given a user query and the definition of a single tool function within <tools></tools> XML tags.

You must generate a function call using the exact name and parameters of the provided tool. You are not allowed to use any other function besides the one given.

Return only the function call, using single quotes for strings and separating parameters with commas.

You are not permitted to deviate from the given API function in any way. You must use the exact function name and parameter types specified, even if you think another function would be more appropriate for the user's request.

Example:
==============
Docstring:
def add_reminder(text: str, date: str, time: str):
    """
    Description:
    Set a reminder for a task on a specified date and time.

    Arguments:
    ——
    - text : str
        The description or name of the task for which the reminder is
        being set.
    - date : str
        The date on which the reminder should be scheduled.
    - time : str
        The time at which the reminder should be scheduled.
    """

Query:
"Add a reminder to buy groceries tomorrow at 2 PM"
Answer:
add_reminder(
  text='Buy groceries',
  date='tomorrow',
  time='2 PM'
)
==============
```

Instruction Message

```
Docstring: ${Docstring}
Query: ${Query}
Answer: ${Answer}
```

Figure 6: Prompt template for call generation with the retrieved tool for LLAMA3-GROQ-8B-Tool-Use.

Field	Text
Query	Please provide me with the user information for the user with the username 'michaelbrown'. Also, fetch the order details for order ID 31415 and get the inventory status of the store.
Gold Tool's Docstring	def petstore_blitz.getUserByName(): """ Description: Fetch user by name. Arguments: ——— - username : STRING (required) Description: The name that needs to be fetched. Use user1 for testing. """
Answer	petstore_blitz.getUserByName(username="michaelbrown")

Table 3: ToolBench dataset sample ❶.

Field	Text
Query	What's the weather like in New York City for the next three days?
Gold Tool's Docstring	def get_weather_forecast(): """ Provides a weather forecast for a specified location over a given number of days. Each day's forecast includes a brief description of the expected weather conditions. Arguments: ———— - location (str): The location for which the weather forecast is desired. Can be a city name, ZIP code, or other location identifiers. Output: ——— - list[str]: A list of strings, each representing the weather forecast for one day. Each string includes the date and a brief description of the weather conditions. Formatted in 'YYYY-MM-DD: Description' format. """
Answer	get_weather_forecast('New York City')

Table 4: Octopus dataset sample ❷/❸.

Field	Text
Query	**User:** Can you give me a list of all the stress reduction technique categories you offer? **API-Request:** [getCategories()]->["Breathing Techniques", "Progressive Muscle Relaxation", "Meditation", "Yoga", "Biofeedback"] **AI:** Our available stress reduction technique categories are Breathing Techniques, Progressive Muscle Relaxation, Meditation, Yoga, and Biofeedback. **User:** Can you suggest some stress-reducing techniques that are easy to learn? **API-Request:** [getTechniques(category=Ńone, difficulty=éasy)]->["name": "Breathing and Grounding Exercise", "description": "This simple technique involves taking a deep breath, exhaling, and grounding yourself by imagining roots growing out of your feet into the ground. This exercise is an excellent way to fall asleep quickly or to calm down when feeling anxious or stressed.", "category": "Breathing Techniques", "difficulty": "easy", "name": "Progressive Muscle Relaxation", "description": "A technique in which you slowly tense and then relax each muscle group of the body. It is particularly useful for relaxation and reducing anxiety.", "category": "Progressive Muscle Relaxation", "difficulty": "easy", "name": "Body Scan Meditation", "description": "This technique requires focusing your attention on different parts of your body and deliberately relaxing them, which can help reduce stress and anxiety.", "category": "Meditation", "difficulty": "easy"] **AI:** I recommend the following techniques for easy stress reduction: Breathing and Grounding Exercise, Progressive Muscle Relaxation, and Body Scan Meditation. **User:** Can you add the V́isualizationt́echnique to your stress-reducing technique list? **Generate API Request:**
Gold Tool's Docstring	**User:** Can you give me a list of all the stress reduction technique categories you offer? **AI:** Our available stress reduction technique categories are Breathing Techniques, Progressive Muscle Relaxation, Meditation, Yoga, and Biofeedback. **User:** Can you suggest some stress-reducing techniques that are easy to learn? **AI:** I recommend the following techniques for easy stress reduction: Breathing and Grounding Exercise, Progressive Muscle Relaxation, and Body Scan Meditation. **User:** Can you add the 'Visualization' technique to your stress-reducing technique list? **Generate API Request:**
Answer	def addTechnique(): """ Description: Add a new stress reduction technique Arguments: ——— - name : string (optional) Description: The name of the new stress reduction technique Format: Not specified - description : string (optional) Description: A description of the new stress reduction technique Format: Not specified - category : string (optional) Description: The category of the new stress reduction technique Format: Not specified - difficulty : string (optional) Description: The difficulty level of the new stress reduction technique Format: Not specified Output: ——— - data : object (optional) Description: The newly added stress reduction technique Format: Not specified Properties: - name : string (optional) Description: The name of the newly added stress reduction technique Format: Not specified - description : string (optional) Description: A description of the newly added stress reduction technique Format: Not specified - category : string (optional) Description: The category of the newly added stress reduction technique Format: Not specified - difficulty : string (optional) Description: The difficulty level of the newly added stress reduction technique Format: Not specified """
Answer	addTechnique(name='Visualization', description='a relaxation exercise in which you create a peaceful mental image of a place or situation', category='Meditation', difficulty='easy')

Table 5: API-Bank dataset sample ❷.

Field	Text
Query	Users want to engage in a conversation with a fictional character based on their persona. This conversation will be used as part of a script for an animation series.
Gold Tool's Docstring	def AutoModelForCausalLM.from_pretrained('pygmalion-6b'): """ Description: Pygmalion 6B is a proof-of-concept dialogue model based on EleutherAI's GPT-J-6B. The fine-tuning dataset consisted of 56MB of dialogue data gathered from multiple sources, which includes both real and partially machine-generated conversations. The model was initialized from the uft-6b ConvoGPT model and fine-tuned on 48.5 million tokens for 5k steps on 4 NVIDIA A40s using DeepSpeed. """
Answer	AutoModelForCausalLM.from_pretrained('pygmalion-6b')

Table 6: APIBench dataset sample ❸.

Field	Text
Query	Search for a Chicken Noodle Soup recipe and a Vegan Salad recipe.
Gold Tool's Docstring	def recipe_search.find(): """ Description: Locate recipes based on the type of dish. Arguments: ——— - dish : string = None (required) The name of the dish to search for. - diet : string = Keto (optional) Dietary preference. """
Answer	recipe_search.find(dish="Chicken Noodle Soup", diet="Vegan")

Table 7: BFCL dataset sample ❹.

Field	Text
Query	Help me with a quick d20 roll, I've got a crucial decision to make in my game.
Gold Tool's Docstring	def diceroller(): """ Description: App for rolling dice using the d20 or Fate/Fudge systems. """
Answer	diceroller()

Table 8: ToolE dataset sample ❺/❼.

Hyperparameter	Search space
Random seed	{0, 42∗, 100}
|Negatives|	{1, 2, 3∗}
Negatives selection	Sampling every $T = 50$ training steps
Max sequence length (encoder)	512
Max sequence length (LLM)	1024
Loss weighting factor λ	{0.1, 0.3∗, 0.5, 0.7, 0.9}
Retriever likelihood temperature γ	{0.3, 0.5∗, 0.7, 1}
LLM likelihood temperature β	{0.3, 0.5∗, 0.7, 1}
|Epochs|	2
Fine-tuning optimizer	AdamW (0.9 β_1, 0.999 β_2, 0.01 w. decay)
Training batch size	2
Test batch size	4
Cosine learning rate	{$1e^{-6}$, $1e^{-5}$∗, $5e^{-5}$, $1e^{-4}$, $2e^{-4}$}

Table 9: Explored hyperparameters along with their empirical search grid. ∗ marks the final picked values.

E Computational Budget

All experiments were performed on machines equipped with NVIDIA RTX 3090 GPUs (24GB VRAM). The total compute time required for training and evaluation across all PORTS' variants amounted to approximately 500 GPU-hours. This includes finetuning on multiple datasets, ablative experiments, running inference with large language models, and conducting retrieval evaluations.

F Robustness

The efficacy of PORTS was evaluated through ablation studies to determine optimal parameter configuration and assess robustness across configuration variations, using the ToolE ❺ dataset with RoBERTa as encoder and LLAMA3-8B as generative models. To demonstrate the effectiveness of the contrastive loss, we examined the impact of using different numbers of negative examples in the learning process, with results in Figure 7 illustrating advantages of incorporating larger numbers of examples which better guide preference optimization. We investigated the effects of varying weighting factors β and γ, with Figure 8 showing higher β and lower γ values yield improved results, optimal when both are set to 0.5. Additionally, we examined the influence of random seeds on our method, focusing on their impact on input data distribution and dropout layer behavior, with results in Table 10 demonstrating the robustness and effectiveness of PORTS and its low variance in response to such configuration changes.

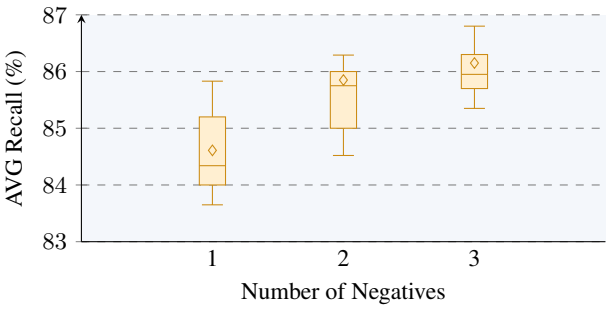


Figure 7: Average Recall@K across different numbers of negatives and seeds on the ToolE ❺ dataset using ModernBERT and LLAMA3-8B.

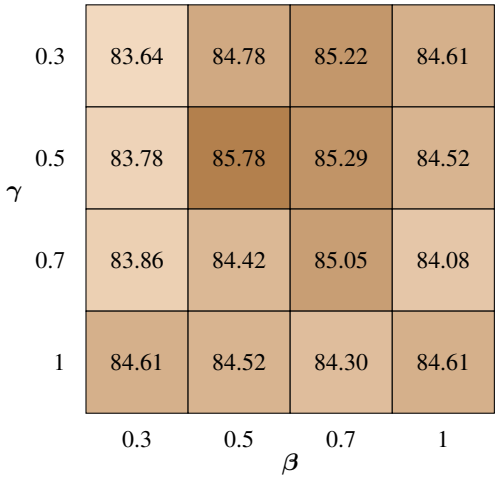


Figure 8: Confusion Matrix for Average Recall@K on the ToolE ❺ dataset using ModernBERT and LLAMA3-8B, with varying β and γ hyperparameters.

Dataset	AVG Recall	Seed	σ^2
ToolBench ❶	**17.45**	**0**	1.76
	14.38	42	
	<u>16.84</u>	<u>100</u>	
API-Bank ❷	<u>56.93</u>	<u>0</u>	33.59
	60.38	**42**	
	46.73	100	
APIBench ❸	26.10	0	1.20
	27.75	**42**	
	<u>27.10</u>	<u>100</u>	
BFCL ❹	55.45	0	7.82
	<u>60.51</u>	<u>42</u>	
	61.98	**100**	
ToolE ❺	86.86	0	0.69
	88.32	**42**	
	<u>86.37</u>	<u>100</u>	
Octopus ❻	<u>95.00</u>	<u>0</u>	19.39
	96.66	**42**	
	86.60	100	

Table 10: Per-dataset variance (σ^2) of the average Recall@K with training runs using different random seeds. Bold and underline denote the best and second-best runs for each dataset.

G Clustering Properties of Tool Embeddings

API docstrings exhibit skewed token distributions, dominated by recurring elements such as data types and keywords. Compared to general-domain retrieval corpora, tool-related datasets form well-separated semantic clusters, owing to their concise yet distinctive functional signatures. Without targeted supervision, these structural properties can lead retrieval models to rely on superficial lexical cues or converge toward trivial matches. To characterize the clustering tendency of these representations, we apply standard unsupervised algorithms–including K-Means (with $K \in [2, 16]$), Agglomerative Clustering (Ward linkage), and DBSCAN (with $\epsilon = 0.3$)—on 20,000 randomly sampled embedding vectors per dataset. We use four top-performing models from the MTEB leaderboard[7] to extract these representations: `BAAI/bge-m3`,[8] `intfloat/multilingual-e5-large-instruct`,[9] `answerdotai/ModernBERT-base`,[10] and `Alibaba-NLP/gte-multilingual-base`.[11]
The silhouette coefficient is computed to quantify clustering quality, capturing both intra-cluster cohesion and inter-cluster separation. As summarized in Figure 9, tool-centric datasets consistently achieve higher silhouette scores than general-domain corpora, including MSMarco (Nguyen et al., 2016), HotpotQA (Yang et al., 2018), and PubMedQA (Jin et al., 2019). This high clustering tendency presents a fundamental limitation where semantically similar tools concentrate within the same dense clusters, making contrastive supervision essential to differentiate tools that share similar descriptive features yet possess distinct functionalities and produce varying effects on LLM behavior, necessitating targeted intra-cluster contrastive learning to prevent training from optimizing merely for tool relevance rather than functional effectiveness. This gap highlights the stronger intrinsic structure of tool embeddings and further motivates our contrastive supervision strategy to promote fine-grained, functionally meaningful distinctions beyond those induced by LLM likelihoods alone.

[7] huggingface.co/spaces/mteb/leaderboard
[8] huggingface.co/BAAI/bge-m3
[9] huggingface.co/intfloat/multilingual-e5-large-instruct
[10] huggingface.co/answerdotai/ModernBERT-base
[11] huggingface.co/Alibaba-NLP/gte-multilingual-base

H Complete Results

Table 11 and Table 12 complement the results of the main paper, listing the retrieval scores achieved by training encoders under the supervision signal of each LLM explored.

I Qualitative Example

Table 13 presents a specific query from the ToolE test set, demonstrating the contrast between the top-3 tools retrieved by BGE with and without `PORTS` tuning. The results clearly illustrate that our alignment process not only successfully positions the correct tool at the top rank, but also generates a significantly sharpened preference distribution.

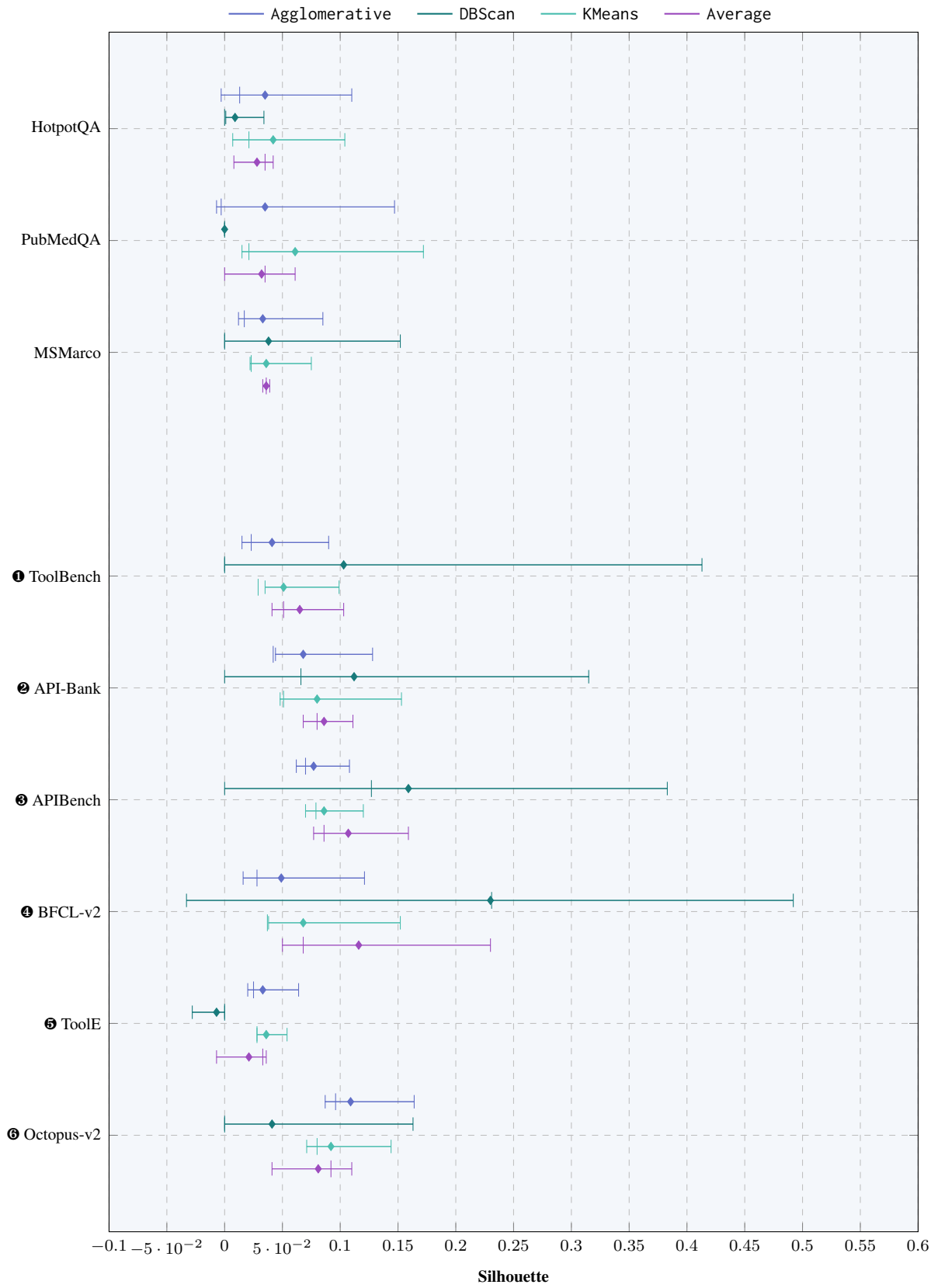


Figure 9: Average Silhouette scores across datasets (top–general-domain, bottom–tool-specific) and clustering algorithms, computed over 20,000 sampled embeddings per dataset. Results are aggregated across multiple encoder models. Higher scores indicate more compact and well-separated clusters, reflecting stronger semantic structure in the embedding space.

Encoder	Dataset	Method	LLM	Recall (%) @1	Recall (%) @2	Recall (%) @3	NDCG (%) @1	NDCG (%) @3	NDCG (%) @5	Δ Baseline Recall	Δ Baseline NDCG
ModernBERT-base	❶	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	11.89	17.13	20.45	11.89	17.10	19.10	15.48	15.18
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	10.34	16.50	20.58	10.34	16.11	18.08	14.80	14.09
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	12.11	19.20	23.40	12.11	19.30	23.10	17.23	18.13
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	11.53	18.52	22.80	11.53	18.05	21.19	16.61	16.11
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	18.23	21.10	25.60	18.23	20.28	24.31	20.63	19.94
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	12.56	20.11	24.80	12.56	19.67	22.19	18.14	17.14
	❷	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	49.70	62.78	68.06	49.70	61.29	63.38	56.47	53.75
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	4.00	5.81	8.87	4.00	6.30	8.00	1.67	1.12
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	49.84	64.35	70.80	49.84	62.27	65.32	57.10	54.14
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	45.32	61.94	68.23	45.32	59.95	62.44	53.93	50.90
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	43.87	54.67	64.84	43.87	58.10	61.20	49.90	50.23
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	10.00	14.19	18.06	10.00	15.17	18.16	9.52	9.13
	❸	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	18.94	25.27	28.12	18.94	22.00	23.80	22.58	20.34
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	8.74	12.61	15.35	8.74	12.55	14.56	10.70	10.95
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	21.50	27.40	30.53	21.50	25.22	26.78	25.94	23.50
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	5.66	8.46	9.86	5.66	8.21	9.23	6.46	6.22
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	13.45	18.93	21.80	13.45	16.90	18.00	16.53	15.06
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	6.78	9.52	11.99	6.78	10.01	11.03	7.90	8.02
	❹	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	57.85	67.65	70.79	57.85	66.11	67.12	58.78	57.09
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	48.43	58.60	64.14	48.43	58.18	60.09	50.40	49.06
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	58.12	69.21	73.52	58.12	68.38	69.22	60.30	59.24
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	53.97	64.88	68.39	53.97	62.61	64.93	55.76	54.50
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	39.74	52.14	56.93	39.74	49.60	52.50	42.95	41.34
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	38.26	47.87	53.05	38.26	47.11	49.18	39.74	38.12
	❺	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	60.33	72.45	77.15	60.33	70.29	72.34	57.51	55.65
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	10.53	13.92	16.18	10.53	14.21	15.23	1.08	1.21
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	59.90	71.90	76.90	59.90	70.04	72.06	57.10	55.02
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	49.38	59.84	64.23	49.38	58.17	60.79	45.35	44.11
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	56.70	70.00	74.89	56.70	67.11	69.16	54.73	52.12
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	40.65	51.54	57.53	40.65	51.03	53.04	37.44	36.02
	❻	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	95.00	100	100	95.00	95.25	98.25	71.66	70.16
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	87.50	97.50	100	87.50	95.06	95.06	68.33	66.54
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	95.00	100	100	95.00	95.20	98.20	71.53	70.03
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	75.00	90.00	97.50	75.00	88.00	89.00	60.83	58.00
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	95.00	100	100	95.00	95.20	98.20	71.53	70.03
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	85.00	90.00	97.50	85.00	92.00	93.00	64.17	64.00
	❼	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	74.60	83.90	86.80	74.60	81.23	83.55	61.24	59.79
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	56.82	66.99	72.55	56.82	66.02	68.14	44.94	44.07
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	72.98	83.60	86.50	72.98	81.21	83.01	60.51	59.11
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	58.51	68.54	74.10	58.51	67.62	69.78	46.53	45.30
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	71.10	81.70	85.57	71.10	79.13	81.08	58.94	57.09
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	53.90	64.00	69.08	53.90	63.11	66.02	41.81	41.00
	❽	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	96.00	100	100	96.00	98.22	98.22*	23.89	24.48
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	80.00	100	100	80.00	92.62	92.62*	19.16	15.41
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	95.00	100	100	95.00	98.00	98.20*	23.57	24.00
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	77.50	100	100	77.50	92.00	92.00*	18.33	15.00
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	95.00	100	100	95.00	98.00	98.20*	23.57	24.00
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	77.50	100	100	77.50	92.00	92.00*	18.33	15.00

* NDCG@4 since the out-of-domain version of Octopus-v2 has 4 tools only.

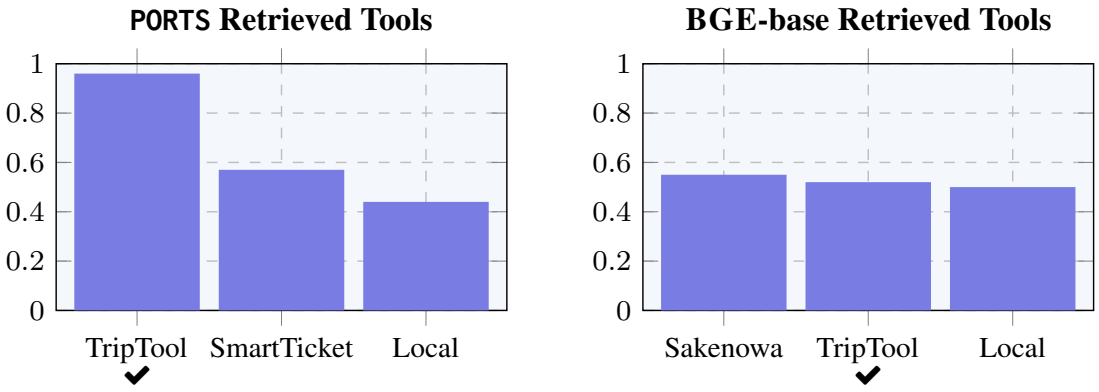
Table 11: **PORTS Recall@K and NDCG@K per dataset-loss-generator (test set) with ModernBERT as base encoder model.** The positive gains in metric scores over the baselines are highlighted (the brighter, the better).

Encoder	Dataset	Method	LLM	Recall (%) @1	Recall (%) @2	Recall (%) @3	NDCG (%) @1	NDCG (%) @3	NDCG (%) @5	Δ Baseline Recall	Δ Baseline NDCG
BGE-base	❶	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	23.20	34.60	41.24	23.20	36.20	38.70	6.66	8.00
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	22.56	33.74	40.51	22.56	33.00	36.65	5.92	5.73
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	24.64	35.30	42.46	24.64	35.22	41.60	7.78	11.13
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	22.32	33.58	40.43	22.32	32.83	36.13	5.84	5.67
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	25.80	36.05	43.35	25.80	35.50	42.20	8.71	9.50
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	21.51	33.38	40.60	21.51	33.04	37.07	5.48	6.05
	❷	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	59.00	75.65	80.80	59.00	72.21	74.23	14.29	12.22
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	55.32	71.94	78.06	55.32	69.01	72.05	10.91	9.03
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	59.12	76.80	81.50	59.12	75.40	76.10	14.94	14.21
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	56.29	75.00	80.00	56.29	70.60	73.32	12.90	10.73
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	49.52	64.03	71.45	49.52	62.30	66.10	4.14	3.00
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	45.40	60.15	65.97	45.40	57.20	59.38	3.98	2.79
	❸	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	30.42	37.11	40.68	30.42	33.50	34.70	20.10	17.21
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	19.55	28.29	33.05	19.55	27.45	30.02	10.84	10.67
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	30.64	37.20	41.06	30.64	33.90	35.20	20.18	17.26
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	19.10	28.57	33.05	19.10	27.05	30.04	10.79	10.03
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	25.83	34.45	39.72	25.83	30.90	32.90	17.22	14.06
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	18.88	28.42	32.89	18.88	26.80	29.80	7.53	9.07
	❹	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	65.25	73.75	78.19	65.25	72.70	74.60	4.87	5.13
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	65.06	73.57	77.82	65.06	73.22	74.24	4.62	5.11
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	67.20	73.23	78.10	67.20	74.60	73.10	5.31	5.63
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	66.17	73.20	77.82	66.17	72.92	74.31	4.86	5.13
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	64.00	74.00	78.00	64.00	72.02	73.30	4.56	4.03
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	59.35	65.40	73.06	59.35	67.26	71.18	3.56	3.78
	❺	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	66.60	78.10	82.29	66.60	76.03	77.05	14.59	14.06
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	66.65	76.91	80.84	66.65	75.04	77.07	12.65	12.06
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	66.59	78.02	82.29	66.59	76.16	77.17	14.61	14.10
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	67.23	77.54	81.06	67.23	75.50	76.82	13.12	13.18
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	67.35	79.48	83.75	67.35	77.00	78.00	14.71	14.12
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	65.10	77.01	80.95	65.10	73.78	76.67	12.37	13.00
	❻	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	97.50	100	100	97.50	100	100	1.67	2.17
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	95.00	100	100	95.00	98.00	98.00	0.83	0.22
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	95.00	100	100	95.00	98.00	98.00	0.83	0.22
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	95.00	97.50	100	95.00	98.00	98.00	0	0
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	95.00	100	100	95.00	98.00	98.00	0.83	0.22
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	95.00	97.50	100	95.00	98.00	98.00	0	0
	❼	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	78.85	89.07	91.89	78.85	86.21	87.19	8.74	9.17
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	73.93	85.19	89.14	73.93	83.11	84.13	4.89	6.12
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	89.87	92.20	94.04	89.87	91.10	92.35	14.18	17.11
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	74.60	86.24	90.30	74.60	83.98	85.09	5.85	7.22
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	89.30	92.10	92.90	80.00	87.00	88.00	12.84	11.00
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	74.60	86.24	90.30	74.60	84.12	85.17	5.85	7.11
	❽	$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-8B	97.50	100	100	97.50	100	100*	0.84	1.17
		$\mathcal{L}_{\text{replug}}$	LLAMA3-8B	95.00	100	100	95.00	98.00	98.00*	0	0
		$\mathcal{L}_{\texttt{PORTS}}$	LLAMA3-GROQ-8B-Tool-Use	97.00	100	100	96.00	100	100*	0.67	0.36
		$\mathcal{L}_{\text{replug}}$	LLAMA3-GROQ-8B-Tool-Use	95.00	100	100	95.00	98.00	98.00*	0	0
		$\mathcal{L}_{\texttt{PORTS}}$	CODESTRAL-22B-v0.1	96.00	100	100	95.00	100	100*	0.33	0.12
		$\mathcal{L}_{\text{replug}}$	CODESTRAL-22B-v0.1	95.00	100	100	95.00	98.00	98.00*	0	0

* NDCG@4 since the out-of-domain version of Octopus-v2 has 4 tools only.

Table 12: **PORTS Recall@K and NDCG@K per dataset-loss-generator (test set) with BGE as base encoder model.** The positive gains in metric scores over the baselines are highlighted (the brighter, the better).

Query: I'm looking for a hotel in Sapporo.



Gold Tool Docstring: Offer discounted hotel and accommodation bookings, along with personalized hotel and product searches, travel planning, image editing, and more, helping users easily plan their trips and find accommodation and transportation options.

Table 13: **Input-output tool selection example from the ToolE test set.** Cosine similarity comparison between PORTS-tuned BGE (left) and baseline, frozen BGE (right).