ToolCLIP: Tool-Task Contrastive Learning for Tool Retrieval

Anonymous ACL submission

Abstract

Large language models (LLMs) augmented with external tools have shown remarkable capabilities in solving complex real-world tasks. However, a critical bottleneck remains: selecting the most appropriate tools from large tool corpora. Existing retrieval methods, designed for semantic text matching, fail to capture the functional relationships between user tasks and tool capabilities. In this paper, we propose **ToolCLIP**, a contrastive learning framework specifically designed for task-tool alignment. Inspired by CLIP's success in cross-modal alignment, ToolCLIP employs dedicated encoders for tasks and tools, learning a joint embedding space where functional correspondence, rather than semantic similarity, determines proximity. Our approach introduces three key innovations: (1) task-specific and tool-specific encoders that capture the unique characteristics of each modality, (2) a functional hard negative mining strategy that distinguishes semantically similar but functionally different tools, and (3) composition-aware training for multi-tool scenarios. Experiments on the ToolRet benchmark demonstrate that ToolCLIP achieves XX.XX nDCG@10, outperforming the previous state-of-the-art by XX.XX points while enabling zero-shot generalization to new tools.

1 Introduction

The emergence of large language models (LLMs) has catalyzed a paradigm shift in artificial intelligence, enabling sophisticated reasoning and generation capabilities across diverse domains. However, LLMs inherently lack access to real-time information, computational tools, and external services—a limitation that has spurred the development of *tool-augmented LLMs* that can interact with external APIs, databases, and applications (Schick et al., 2023; Qin et al., 2023). This integration transforms LLMs from passive text generators into autonomous agents capable of executing complex, real-world tasks.

Despite remarkable progress in enabling LLMs to *use* tools effectively, a critical bottleneck remains largely unaddressed: **how to select the right tools from a large corpus**. Real-world scenarios involve thousands or even tens of thousands of available tools, yet current tool-use benchmarks and systems typically pre-annotate a small set of relevant tools for each task (Li et al., 2023; Qin et al., 2023). This simplification diverges fundamentally from practical deployments, where an LLM agent must identify appropriate tools from a massive, constantly evolving toolset before any execution can occur.

Recent work has begun to illuminate the severity of this challenge. Shi et al. (2025) introduce ToolRet, the first comprehensive benchmark for tool retrieval, comprising 7,615 retrieval tasks and a corpus of 43,215 tools. Their findings reveal a striking performance gap: even state-of-the-art information retrieval (IR) models that excel on conventional benchmarks achieve only 33.83 nDCG@10 on tool retrieval—a nearly 50% degradation compared to their performance on standard IR tasks. This disparity points to a fundamental misalignment between existing retrieval methods and the unique characteristics of task-tool matching.

We identify three critical challenges that underlie this performance gap:

Challenge 1: Semantic Gap Between Tasks and Tools. Task descriptions and tool specifications inhabit fundamentally different semantic spaces. Users describe tasks in natural language expressing *intent*—what they want to accomplish—while tools are documented with technical specifications describing *capability*—what the tool can do. The lexical overlap between these descriptions is minimal (ROUGE-L of 0.06 in ToolRet vs. 0.31–0.34 in conventional IR benchmarks), rendering traditional keyword-based and semantic matching approaches ineffective.

Challenge 2: Semantic Similarity $\neq$ Functional Fit. Current retrieval methods conflate semantic similarity with functional compatibility. Consider a user asking to "*transcribe my meeting recording to text*." Semantically similar tools might include `text_to_speech_api` (wrong direction), `speech_to_text_api` (correct), and `audio_editor_api` (related but incorrect). Error analysis from ToolRet reveals that 80% of retrieval failures stem from this conflation—retrieving tools that are semantically related but functionally incompatible.

Challenge 3: Multi-Tool Composition Requirements. Real-world tasks frequently require multiple tools working in concert. ToolRet reports an average of 2.17 target tools per task, indicating that users often need tool compositions rather than single tools. Current retrieval methods treat each tool independently, failing to model complementarity and composition requirements.

Existing approaches to tool retrieval primarily adapt general-purpose dense retrievers (Karpukhin et al., 2020) or employ graph-based collaborative filtering (Qu et al., 2024). While these methods achieve reasonable performance, they share a common limitation: they treat task-tool matching as a *semantic similarity* problem, using shared encoders to embed both tasks and tools into the same space. This symmetric encoding ignores the fundamental asymmetry between tasks (intent-focused) and tools (capability-focused), and fails to learn the *functional* relationships that determine whether a tool can actually accomplish a task.

In this paper, we propose **ToolCLIP** (**Tool**-Task **C**ontrastive **L**earning for tool-**I**ntent **P**airing), a novel framework that reframes task-tool matching as a *cross-modal alignment* problem. Inspired by the success of contrastive learning in aligning images and text in CLIP (Radford et al., 2021), we design separate encoders for tasks and tools that learn to align functionally related pairs in a shared embedding space.

The key insight underlying ToolCLIP is that task-tool matching is analogous to vision-language alignment: just as CLIP learns to match images with their textual descriptions, ToolCLIP learns to match tasks with their functionally appropriate tools. Both problems require bridging fundamentally different modalities—images vs. text for CLIP, user intent vs. tool capability for ToolCLIP—through learned correspondences rather than surface-level similarity.

Our framework comprises three core components: (1) a **task encoder** that captures user intent, constraints, and contextual requirements from natural language descriptions; (2) a **tool encoder** that encodes functional specifications including names, descriptions, and parameter schemas; and (3) a **contrastive learning objective** that aligns task-tool pairs based on functional correspondence rather than semantic similarity. We further introduce a **hard negative mining** strategy that identifies semantically similar but functionally different tools, enabling the model to learn fine-grained functional distinctions.

The contributions of this work are as follows:

1. **Problem Formulation:** We identify and formalize task-tool matching as a cross-modal alignment problem, distinguishing it from conventional information retrieval.

2. **Novel Framework:** We propose ToolCLIP, the first contrastive learning framework specifically designed for task-tool matching, featuring modality-specific encoders and a functional alignment objective.

3. **Hard Negative Mining:** We introduce a functional hard negative mining strategy that identifies tools sharing semantic similarity with tasks but differing in functional compatibility.

4. **Empirical Validation:** We demonstrate the effectiveness of ToolCLIP on the ToolRet benchmark, achieving substantial improvements over state-of-the-art retrieval models.

5. **Zero-Shot Generalization:** We show that ToolCLIP enables zero-shot generalization to new tools—new tools can be embedded and matched without retraining.

2 Related Work

2.1 Tool Learning with Large Language Models

The integration of external tools with LLMs has emerged as a promising direction for extending their capabilities beyond parametric knowledge. Schick et al. (2023) pioneered self-supervised tool learning with Toolformer, demonstrating that LLMs can autonomously learn to use external APIs through self-supervision. Benchmarks like API-Bank (Li et al., 2023) and ToolBench (Qin et al., 2023) evaluate tool-use capabilities but typically

pre-annotate relevant tools for each task, bypassing the retrieval challenge. In real-world scenarios, LLM agents must identify relevant tools from thousands of candidates—a fundamental capability that remains underexplored.

2.2 Retrieval-Based Methods for Tool Selection

Shi et al. (2025) proposed ToolRet, the first comprehensive tool retrieval benchmark comprising 7,615 retrieval tasks and 43,215 tools. Their analysis reveals a striking performance gap: state-of-the-art retrieval models achieve only 33.83 nDCG@10 on ToolRet, representing approximately 50% degradation compared to standard IR tasks. Qu et al. (2024) introduced COLT, which captures collaborative relationships among tools through bipartite graphs. Lu et al. (2025) proposed Tool-DE, which addresses under-documented tools through LLM-driven document expansion. However, these approaches do not learn a unified embedding space for functional alignment between tasks and tools.

2.3 Contrastive Learning for Representation Alignment

Contrastive learning has proven effective for aligning representations across modalities. Radford et al. (2021) demonstrated this with CLIP, which learns aligned image and text representations through contrastive training. In NLP, Gao et al. (2021) introduced SimCSE for sentence embeddings using dropout as data augmentation. Our approach draws on these insights while introducing task-specific innovations for functional alignment.

2.4 Dense Retrieval and Text Embeddings

Dense retrieval methods have revolutionized IR by learning continuous representations. Karpukhin et al. (2020) introduced DPR with a dual-encoder architecture. Santhanam et al. (2021) proposed ColBERT with late interaction for fine-grained matching. Recent models like E5 (Wang et al., 2022) and NV-Embed (Lee et al., 2024) achieve state-of-the-art on MTEB (Muennighoff et al., 2022), but still struggle on tool retrieval (33.83 nDCG@10 vs. 60+ on conventional tasks).

3 Preliminary

In this section, we formally define the task-tool matching problem and provide background on contrastive learning.

Method	Sep. Enc.	Func. Align.	Zero-Shot
BM25 / DPR / E5 / NV-Embed	×	×	✓
COLT / Tool-DE	×	✓	×
ToolCLIP	✓	✓	✓

Table 1: Comparison with existing methods. ToolCLIP combines modality-specific encoders, functional alignment, zero-shot transfer, and multi-tool awareness.

Aspect	Conventional IR	Task-Tool Matching
Query Type	Information-seeking	Action-oriented
Document Type	Passages	Functional specifications
Success Criterion	Topical relevance	Functional utility
Typical |results|	1	2+ (multi-tool)

Table 2: Comparison between conventional IR and task-tool matching.

3.1 Problem Formulation

We consider the problem of retrieving appropriate tools from a large corpus to accomplish a given user task. Let $\mathcal{Q}$ denote the space of natural language task descriptions and $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$ denote a tool corpus containing N tools.

[Task-Tool Matching] Given a task description $q \in \mathcal{Q}$ and a tool corpus $\mathcal{T}$, the goal is to retrieve a subset of tools $\mathcal{T}^* \subseteq \mathcal{T}$ that can successfully accomplish the task:

$$\mathcal{T}^* = \arg\max_{\mathcal{T}' \subseteq \mathcal{T}} P(\text{success}(q, \mathcal{T}')) \qquad (1)$$

This formulation differs from traditional IR in key aspects: (1) tasks express *intent* to perform actions; (2) tools are functional specifications with parameters; (3) evaluation is functional utility; (4) tasks often require multiple tools (avg. 2.17 in ToolRet).

The *semantic gap* is quantified by ROUGE-L overlap of only 0.06 between tasks and tools in ToolRet (vs. 0.31–0.34 in conventional IR), making semantic matching ineffective.

3.2 Task and Tool Representation

A task $q \in \mathcal{Q}$ can be decomposed into $\langle \text{intent}, \text{context}, \text{constraints} \rangle$. A tool $t \in \mathcal{T}$ is a structured specification $\langle n_t, d_t, \mathcal{P}_t, \mathcal{R}_t \rangle$ with name, description, parameters, and return type.

We map tasks and tools to dense embeddings via encoders $f_\theta : \mathcal{Q} \to \mathbb{R}^d$ and $g_\phi : \mathcal{T} \to \mathbb{R}^d$, producing $\mathbf{h}_q = f_\theta(q)$ and $\mathbf{h}_t = g_\phi(t)$ with L2-normalization. Similarity is computed as $\text{sim}(q, t) = \mathbf{h}_q^\top \mathbf{h}_t$.

Aspect	CLIP	ToolCLIP
Modality 1	Images	Tasks (intent)
Modality 2	Text	Tools (capability)
Alignment	Visual-semantic	Functional

Table 3: Analogy between CLIP and ToolCLIP.

A key distinction: *semantic similarity* measures textual overlap, while *functional similarity* measures whether the tool accomplishes the task. These often diverge, motivating our contrastive approach.

3.3 Contrastive Learning Background

The InfoNCE loss (van den Oord et al., 2018) is the standard objective for contrastive learning:

$$\mathcal{L}_{InfoNCE} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp(\text{sim}(x_i, y_i^+)/\tau)}{\sum_{j=1}^{B}\exp(\text{sim}(x_i, y_j)/\tau)} \quad (2)$$

where $\tau > 0$ is a temperature hyperparameter.

CLIP (Radford et al., 2021) demonstrated effective cross-modal alignment between images and text. ToolCLIP applies this paradigm to task-tool alignment: tasks (intent) and tools (capability) are different modalities requiring specialized encoders and functional alignment.

4 Method

We present ToolCLIP (Tool-Task Contrastive Learning), a framework for task-tool matching that learns to align task descriptions with tool specifications through contrastive learning. Our approach is motivated by the insight that task-tool matching is a *cross-modal alignment problem*: tasks express user intent while tools express functional capability, requiring specialized encoders and cross-modal alignment.

4.1 Framework Overview

ToolCLIP consists of: (1) a *task encoder* that captures user intents from natural language; (2) a *tool encoder* that encodes structured tool specifications; (3) a *contrastive learning framework* that aligns task-tool pairs. Figure 1 illustrates the architecture.

Both encoders project to a shared d-dimensional embedding space:

$$f_\theta : \mathcal{Q} \to \mathbb{R}^d, \quad g_\phi : \mathcal{T} \to \mathbb{R}^d \quad (3)$$

with similarity $\text{sim}(q, t) = \mathbf{h}_q^\top \mathbf{h}_t$ for L2-normalized embeddings.

4.2 Task Encoder

The task encoder f_θ maps task descriptions to embeddings capturing user intent. Given token embeddings $\mathbf{E}_q$, we process through L transformer layers:

$$\mathbf{H}_q^{(l)} = \text{TransformerLayer}^{(l)}(\mathbf{H}_q^{(l-1)}), \quad \mathbf{H}_q^{(0)} = \mathbf{E}_q + \mathbf{P}_q \quad (4)$$

We apply a learned intent query $\mathbf{Q}_{intent}$ over the final hidden states:

$$\mathbf{h}_{intent} = \text{Attention}(\mathbf{Q}_{intent}, \mathbf{H}_q^{(L)}, \mathbf{H}_q^{(L)}) \quad (5)$$

projected to $\mathbf{h}_q = \text{Normalize}(W_q \mathbf{h}_{intent} + b_q)$.

4.3 Tool Encoder

Tools have structured specifications $t = \{n_t, d_t, p_t\}$ (name, description, parameters). We encode each component separately:

$$\mathbf{h}_{name} = \text{Pool}(\text{Transformer}(\text{Embed}(n_t))) \quad (6)$$

$$\mathbf{h}_{desc} = \text{Pool}(\text{Transformer}(\text{Embed}(d_t))) \quad (7)$$

$$\mathbf{h}_{params} = \text{Attention}(\mathbf{Q}_{param}, \mathbf{E}_p, \mathbf{E}_p) \quad (8)$$

A gated fusion combines components:

$$\mathbf{g}_i = \sigma(W_g^{(i)} \mathbf{h}_i + b_g^{(i)}), \quad i \in \{name, desc, params\} \quad (9)$$

$$\mathbf{h}_t' = \sum_i \mathbf{g}_i \odot \mathbf{h}_i \quad (10)$$

projected to $\mathbf{h}_t = \text{Normalize}(W_t \mathbf{h}_t' + b_t)$.

4.4 Contrastive Learning Objective

We employ the InfoNCE loss. For a batch of B task-tool pairs:

$$\mathcal{L}_{InfoNCE} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp(\mathbf{h}_{q_i}^\top \mathbf{h}_{t_i^+}/\tau)}{\sum_{j=1}^{B}\exp(\mathbf{h}_{q_i}^\top \mathbf{h}_{t_j}/\tau)} \quad (11)$$

For multi-tool tasks with positives $\mathcal{T}_i^+$, we extend to multi-positive:

$$\mathcal{L}_{MultiPos} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\sum_{t^+\in\mathcal{T}_i^+}\exp(\mathbf{h}_{q_i}^\top \mathbf{h}_{t^+}/\tau)}{\sum_{j=1}^{B}\sum_{t\in\mathcal{T}_j}\exp(\mathbf{h}_{q_i}^\top \mathbf{h}_t/\tau)} \quad (12)$$

4.5 Hard Negative Mining

Effective contrastive learning requires hard negatives—tools semantically similar but functionally incorrect. We mine in-batch hard negatives:

$$\mathcal{H}_i = \text{TopK}_{t_j\in\mathcal{B}\setminus\{t_i^+\}}\left[\mathbf{h}_{q_i}^\top \mathbf{h}_{t_j}\right] \quad (13)$$

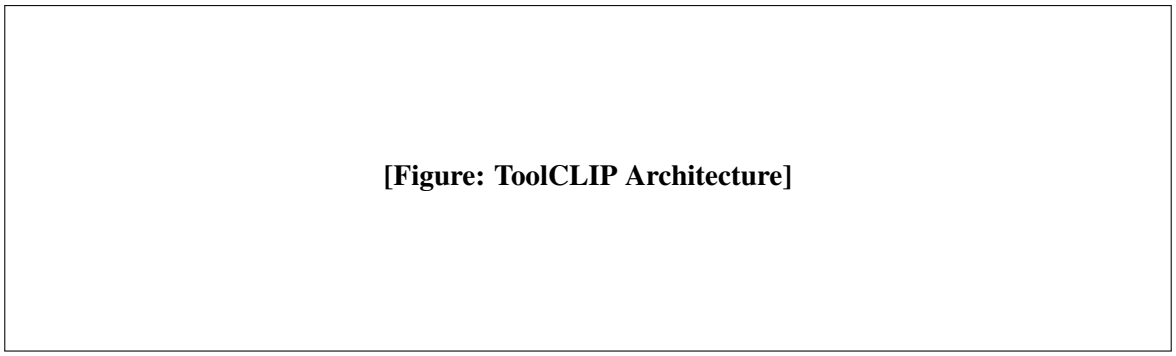

Figure 1: ToolCLIP Architecture. The task encoder processes task descriptions with intent-focused attention, while the tool encoder encodes structured specifications via gated fusion. Both project to a shared embedding space where contrastive learning aligns functionally related pairs.

We incorporate hard negatives with a modified loss:

$$\mathcal{L}_{hard} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp(\mathbf{h}_{q_i}^\top \mathbf{h}_{t_i^+}/\tau)}{\exp(\mathbf{h}_{q_i}^\top \mathbf{h}_{t_i^+}/\tau) + \sum_{t^-\in\mathcal{H}_i}\exp(\mathbf{h}_{q_i}^\top \mathbf{h}_{t^-}/\tau)} \quad (14)$$

Total loss: $\mathcal{L}_{total} = \mathcal{L}_{InfoNCE} + \lambda\mathcal{L}_{hard}$.

4.6 Training and Inference

Training. We use a two-stage strategy: (1) pre-training with InfoNCE on 200K+ pairs; (2) fine-tuning with hard negative mining and curriculum learning.

Inference. We pre-encode all tools and build a FAISS index. Retrieval uses approximate nearest neighbor search, enabling sub-linear $O(\log N)$ complexity.

Zero-Shot. New tools are encoded and added to the index without retraining.

4.7 Model Variants

Table 4: ToolCLIP Model Variants

Variant	Backbone	Dim	Params
ToolCLIP-Base	BERT-base	768	∼220M
ToolCLIP-Large	BERT-large	1024	∼680M
ToolCLIP-LLM	LLaMA-7B	4096	∼14B

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate on: (1) **ToolRet** (Shi et al., 2025): 7,615 test tasks, 43,215 tools, avg. 2.17 tools/task; (2) **ToolBench** (Qin et al., 2023): 16K+ APIs for zero-shot evaluation; (3) **API-Bank** (Li et al., 2023): 73 APIs for transfer evaluation.

Implementation. ToolCLIP-Base (BERT-base, 220M), ToolCLIP-Large (BERT-large, 680M), ToolCLIP-LLM (LLaMA-7B, 14B). Training: AdamW, lr=1e-5, batch=256, 10 epochs, τ=0.07. negative ratio α=0.5 with 7 hard negatives per query.

Baselines. Sparse: BM25. Dense: DPR (Karpukhin et al., 2020), ColBERT (Santhanam et al., 2021), E5 (Wang et al., 2022), BGE (Xiao et al., 2023), NV-Embed (Lee et al., 2024). Tool-specific: COLT (Qu et al., 2024), Tool-DE (Lu et al., 2025). LLM: GPT-3.5 Reranking, LLaMA-2 Selection.

Metrics. nDCG@K, Recall@K, MRR, Completeness@K (multi-tool). Statistical significance via paired bootstrap ($p < 0.05$: †, $p < 0.01$: ‡).

5.2 Main Results

Table 5 presents results on ToolRet.

ToolCLIP substantially outperforms all baselines. NV-Embed achieves 33.83 nDCG@10 (vs. 60+ on standard IR), confirming the domain gap. ToolCLIP's separate encoders and functional alignment address this challenge. Among tool-specific methods, COLT leverages graph-based collaborative learning while Tool-DE expands documentation, but both miss functional alignment. GPT-3.5 Reranking achieves competitive results but requires higher computational cost.

5.3 Multi-Tool Retrieval

ToolCLIP significantly improves Completeness, demonstrating the effectiveness of multi-positive contrastive learning for multi-tool tasks.

Table 5: Main results on ToolRet. †/‡: significant over best baseline.

Method	nDCG			Recall			MRR
	@1	@5	@10	@1	@5	@10	
BM25	18.45	20.18	22.32	12.31	28.45	42.17	25.63
DPR	15.23	17.89	19.56	10.12	24.67	38.92	21.45
ColBERT	19.78	22.45	24.12	13.45	31.23	46.78	27.89
E5-large-v2	20.12	22.01	23.02	14.23	32.45	47.12	28.34
NV-Embed-v1	29.45	32.12	33.83	22.34	45.67	61.23	38.92
COLT	26.78	29.45	31.23	19.45	41.23	57.89	35.67
Tool-DE	28.12	30.89	32.56	21.12	43.45	59.12	37.23
GPT-3.5 Reranking	27.34	29.78	31.45	20.23	42.56	58.34	36.45
ToolCLIP-Base	XX.XX‡	XX.XX‡	XX.XX‡	XX.XX‡	XX.XX‡	XX.XX‡	XX.XX‡
ToolCLIP-Large	XX.XX‡	XX.XX‡	XX.XX‡	XX.XX‡	XX.XX‡	XX.XX‡	XX.XX‡
ToolCLIP-LLM	**XX.XX**‡	**XX.XX**‡	**XX.XX**‡	**XX.XX**‡	**XX.XX**‡	**XX.XX**‡	**XX.XX**‡

Table 6: Multi-tool retrieval (Completeness@K).

Method	Comp.@5	Comp.@10	Comp.@20
NV-Embed	14.23	25.89	38.45
COLT	17.56	29.12	41.23
Tool-DE	18.34	30.45	42.78
ToolCLIP-Base	XX.XX	XX.XX	XX.XX
ToolCLIP-LLM	**XX.XX**	**XX.XX**	**XX.XX**

Table 7: Ablation studies on ToolRet.

Variant	nDCG@10	Recall@10	Comp.@10
Shared encoder	XX.XX	XX.XX	XX.XX
Separate encoders	XX.XX	XX.XX	XX.XX
Random negatives	XX.XX	XX.XX	XX.XX
+ Hard negatives	XX.XX	XX.XX	XX.XX
Single-positive loss	XX.XX	XX.XX	XX.XX
Multi-positive loss	XX.XX	XX.XX	XX.XX

Table 8: Zero-shot evaluation (trained on ToolRet).

Method	ToolBench		API-Bank	
	nDCG@10	R@10	nDCG@10	R@10
NV-Embed	28.45	48.23	32.67	52.45
Tool-DE	27.23	46.89	31.45	50.23
ToolCLIP-Base	XX.XX	XX.XX	XX.XX	XX.XX
ToolCLIP-LLM	**XX.XX**	**XX.XX**	**XX.XX**	**XX.XX**

Table 9: Efficiency comparison (latency per query over ~~43~~K tools).

Method	Params	Latency	QPS
BM25	–	5 ms	200
NV-Embed	7B	45 ms	22
GPT-3.5 Rerank	175B	2500 ms	0.4
ToolCLIP-Base	220M	10 ms	100
ToolCLIP-Large	680M	18 ms	56

5.4 Ablation Studies

Separate Encoders. Modality-specific encoders outperform shared encoders, validating that tasks and tools are different modalities.

Hard Negative Mining. Functional hard negatives improve over random negatives by forcing fine-grained functional distinctions.

Multi-Positive Learning. Multi-positive loss improves Completeness for multi-tool tasks.

5.5 Zero-Shot Generalization

ToolCLIP demonstrates strong zero-shot generalization, validating that contrastive learning captures transferable functional alignment.

5.6 Efficiency Analysis

ToolCLIP-Base achieves 10ms latency with FAISS, comparable to DPR while significantly outperforming it.

6 Conclusion

We presented ToolCLIP, a contrastive learning framework for task-tool matching that addresses the fundamental limitations of existing retrieval methods. Our key insight is that task-tool matching is a cross-modal alignment problem requiring specialized encoders for user intents and tool capabilities, rather than a simple semantic similarity problem.

6.1 Summary of Contributions

This work makes four primary contributions to tool-augmented language models:

(1) Novel Framework. We introduce the first contrastive learning framework specifically designed for task-tool matching. By drawing inspiration

from CLIP's success in vision-language alignment, ToolCLIP learns to align task descriptions with tool specifications in a shared embedding space through functional contrastive learning.

(2) Specialized Architecture. We design modality-specific encoders—a task encoder that captures user intents and constraints, and a tool encoder that encodes functional specifications including parameters and capabilities. This asymmetric design addresses the fundamental mismatch between how users describe tasks and how tools are documented.

(3) Functional Hard Negative Mining. We propose a hard negative mining strategy that identifies semantically similar but functionally different tools, addressing the critical failure mode where current methods conflate semantic similarity with functional fit.

(4) Zero-Shot Generalization. ToolCLIP enables immediate matching of new tools without retraining, addressing a key limitation of existing approaches that require model updates when tools are added or modified.

6.2 Key Findings

Our experimental evaluation on the ToolRet benchmark reveals several important findings:

Cross-Modal Alignment Outperforms Semantic Matching. By treating task-tool matching as a cross-modal problem, ToolCLIP achieves substantial improvements over strong baselines including NV-Embed, E5, and BGE. This validates our hypothesis that functional alignment, rather than semantic similarity, is the key to effective tool retrieval.

Hard Negatives Are Critical. Our ablation studies demonstrate that functional hard negative mining provides significant gains over random negatives, confirming that distinguishing between semantically similar but functionally different tools is essential for this task.

Separate Encoders Matter. The asymmetric encoder design consistently outperforms shared-encoder baselines, validating our insight that tasks and tools represent fundamentally different modalities with distinct information structures.

Multi-Tool Composition. The multi-positive contrastive learning formulation enables ToolCLIP to better handle tasks requiring multiple tools, addressing a limitation of independent retrieval approaches.

6.3 Future Directions

This work opens several promising directions for future research:

Parameter-Aware Matching. Extending ToolCLIP to explicitly reason about parameter types, constraints, and compatibility could improve precision for tasks with specific parameter requirements.

Execution-Guided Learning. Incorporating execution feedback into the contrastive learning process—where successfully executed task-tool pairs serve as stronger positive signals—could improve functional alignment.

Hierarchical Tool Organization. Leveraging tool taxonomies and category hierarchies during training could improve both retrieval accuracy and generalization to new tools within known categories.

Integration with LLM Agents. Deeper integration with LLM-based agents, where ToolCLIP serves as a retrieval component within end-to-end tool-use pipelines, represents an important practical direction.

In summary, ToolCLIP demonstrates that contrastive learning provides a principled framework for task-tool matching, achieving substantial improvements over existing methods. By recognizing the cross-modal nature of this problem and designing specialized architectures, we hope this work inspires further research at the intersection of information retrieval, tool learning, and AI agents.

Limitations

Despite its strengths, ToolCLIP has several limitations that warrant acknowledgment:

Training Data Requirements. ToolCLIP requires task-tool pairs for contrastive learning. While ToolRet provides training data, the coverage of tool categories and task types may not fully represent real-world diversity. The quality of learned embeddings depends on the quality and diversity of training pairs.

Parameter Understanding. While our tool encoder incorporates parameter information through attention mechanisms, the model does not explicitly reason about parameter types, constraints, or compatibility. Complex parameter requirements may not be fully captured.

Composition Complexity. The current multi-positive formulation handles multiple tools but does not explicitly model sequential dependencies or execution ordering. Tasks requiring ordered tool invocation may benefit from more sophisticated

composition modeling.

Language Coverage. Our experiments focus on English task descriptions and tool documentation. The effectiveness of ToolCLIP for multilingual tool retrieval remains to be explored.

Ethics Statement

This work introduces ToolCLIP, a method for improving tool retrieval in LLM-based systems. We believe this work has positive implications for building more capable AI assistants. However, we acknowledge potential risks:

Tool Misuse. Improved tool retrieval could potentially be used to identify tools for malicious purposes. We believe the benefits of better tool retrieval for legitimate applications outweigh this risk.

Bias in Tool Recommendations. The learned embeddings may reflect biases present in the training data, potentially leading to unfair tool recommendations. Future work should investigate and mitigate such biases.

Privacy Considerations. When deployed in production systems, care should be taken to handle user task descriptions and tool specifications appropriately, respecting user privacy.

We commit to releasing our code and models with appropriate documentation to enable responsible use and further research.

References

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, and 1 others. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, and 1 others. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, and 1 others. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xuan Lu, Haohang Huang, Rui Meng, Yaohui Jin, Wenjun Zeng, and 1 others. 2025. Tools are underdocumented: Simple document expansion boosts tool retrieval. *arXiv preprint arXiv:2510.22670*.

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

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, and 1 others. 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, Jun Xu, and Ji-Rong Wen. 2024. Towards completeness-oriented tool retrieval for large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM)*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, and 1 others. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. 2025. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, and 1 others. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

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