Tools Are Not Islands: Set-Level Tool Retrieval for LLM Agents via Query-Conditioned Hyperedge Prediction

Xinyi Hong[1], **Pinjun Dong**[2], **Xinyang Yu**[2], **Binyan Jiang**[2*]

[1]School of Mathematical Sciences, Shanghai Jiao Tong University
[2]Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University
xinyi.hong@sjtu.edu.cn, pin-jun.dong@connect.polyu.hk, X.Yu59@lse.ac.uk, by.jiang@polyu.edu.hk

Live Demo GitHub Hugging Face

Abstract

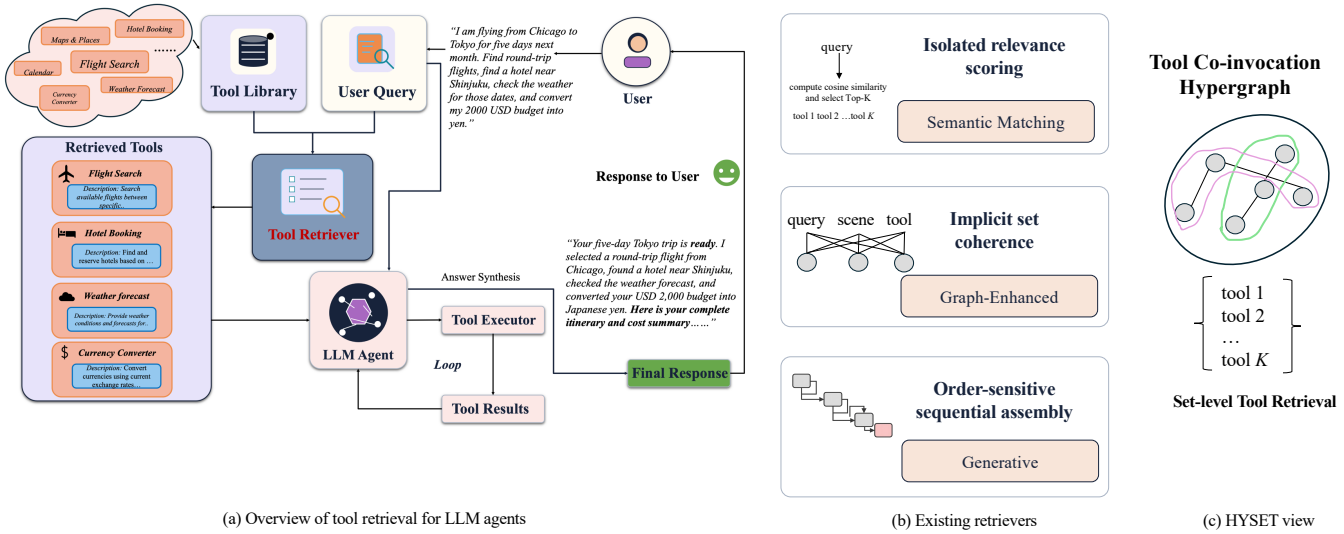
Large language model (LLM) agents increasingly rely on invoking external tools to complete real-world tasks. Tool retrieval, which selects a small task-relevant subset from a library of thousands of tools before the agent acts, has therefore become a critical component of LLM agent pipelines. However, existing retrievers either score each tool in isolation or assemble the tool set sequentially, so the joint utility of a candidate set is never evaluated as a whole. In this paper, we propose **HYSET**, short for **HY**peredge-based **SE**t-level **T**ool retrieval. Our contributions are threefold: (i) we formulate tool retrieval as query-conditioned hyperedge prediction on a tool co-invocation hypergraph, under which the tool set itself becomes the unit of scoring and most existing retrieval paradigms reduce to restricted instances; (ii) we capture size-dependent tool compatibility through cardinality-specific interactions; and (iii) we design HYSET as a pre-selection module requiring no modification to the downstream agent. Experiments on ToolBench demonstrate that HYSET consistently outperforms state-of-the-art baselines in both tool retrieval performance and end-to-end task success. Beyond the in-domain setting, HYSET further supports zero-shot/few-shot transfer, generalizing to held-out tools/categories and unseen domains with minimal supervision.

1 Introduction

Large language models (LLMs) are increasingly deployed as autonomous agents capable of invoking external tools to complete real-world tasks (Schick et al. 2023; Shen et al. 2023; Patil et al. 2023; Yao et al. 2022). As agents grow more capable and their invocation demands expand, tool libraries have scaled from dozens of handcrafted functions to large-scale API ecosystems containing thousands of real-world endpoints. ToolBench (Qin et al. 2023), for instance, includes 16,464 API endpoints spanning 49 categories from the RapidAPI Hub, a scale that renders exhaustive in-context presentation of every tool description impractical. One might expect that increasingly long context windows would eventually eliminate the need for explicit tool selection. However, empirical evidence shows that the effective use of information degrades sharply when context exceeds tens of thousands of tokens (Liu et al. 2023), and injecting the full tool library into every prompt would incur prohibitive latency and cost at agent scale. Moreover, the difficulty is not one of scale alone. Real-world tasks rarely depend on a single API, and a query is typically resolved by several APIs invoked jointly (Qin et al. 2023; Qu et al. 2024), so what must be retrieved is not a ranked list of individually relevant tools but a jointly useful tool set. Tool retrieval, which selects such a small task-relevant subset from a large API library before the agent acts, therefore remains a persistent infrastructure bottleneck rather than a transitional one.

Existing tool retrievers address this bottleneck through three representative paradigms. The most common paradigm, semantic-matching retrieval, scores each API independently against the query. Sparse methods such as BM25 (Robertson and Zaragoza 2009) rank APIs by lexical overlap, while dense bi-encoders such as Contriever (Izacard et al. 2021) and Sentence-BERT-style models (Reimers and Gurevych 2019; Qin et al. 2023) score APIs by cosine similarity in a shared embedding space. All such methods treat each tool as an independently scorable item, so the score of a candidate set is simply an aggregate of per-tool signals. This treatment breaks down whenever tool utility is a joint property of the set rather than a sum of individual values. A tool that ranks low in isolation may become indispensable alongside other selected APIs, and a group of individually high-scoring tools may collectively fail to cover the full scope of a task. This gap is evident in practice. On ToolBench, a fine-tuned Contriever retriever reported by Qu et al. (2024) attains a Recall@3 of 68.6%, yet its COMP@3, the fraction of queries whose complete required tool set is covered by the top-3 results, is only 39.7%. To move beyond such independent scoring, the other two paradigms have recently been explored. Graph-enhanced retrievers such as COLT (Qu et al. 2024) augment semantic matching with dual-view collaborative learning over a bipartite query-scene-tool graph, improving the recovery of the complete required tool set. Nevertheless, their collaborative signal is distilled into per-tool embeddings during training, and inference still reduces to independent top-k ranking, so set-level coherence is only approximated rather than scored explicitly. Generative retrievers such as ToolGen (Wang et al. 2024), building on generative retrieval (Tay et al. 2022), instead emit tool identifiers directly from the language model. However, the tool set is assembled step by step from local

*Corresponding author.



(a) Overview of tool retrieval for LLM agents (b) Existing retrievers (c) HYSET view

Figure 1: **Overview and motivation of HYSET.** (a) Tool-retrieval pipeline; (b) limitations of existing retrievers; and (c) our set-level view as query-conditioned hyperedge prediction.

conditional probabilities under a sequence-likelihood objective, so complete candidate sets are never explicitly compared, and whether the assembled set is coherent and complete can only be judged after generation ends. Figure 1 contrasts these paradigms with our set-level view. Supplementary Material H gives extended related work.

Across all three paradigms, tools are either scored independently or generated sequentially, and no candidate set is ever assessed as a whole. We therefore argue that **tools are not islands, and tool retrieval is inherently a set-level problem**. This position raises two questions. ***(Q1)*** *Can tool sets be scored as a whole, and does this improve retrieval?* ***(Q2)*** *Do tool co-invocation patterns vary with set size, and does modeling these differences help?* Both are grounded in a representative ToolBench example. For the travel-planning query "*I am flying from Chicago to Tokyo for five days next month. Find round-trip flights, book a hotel near Shinjuku, check the weather for those dates, and convert my 2,000 USD budget into yen*", whose ground-truth tools are $\{Flight, Hotel, Weather, Currency\}$, flight terms dominate the wording and a fine-tuned dense retriever scores *Flight*, *CheapFlight*, *FlightTracker*, *Hotel*, *Weather* and *Currency* at 0.92, 0.89, 0.87, 0.80, 0.55 and 0.32. Top-4 retrieval therefore returns $\{Flight, CheapFlight, FlightTracker, Hotel\}$ and leaves two subtasks uncovered. The failure is structural rather than a matter of calibration. Once *Flight* is selected, a second flight API contributes almost nothing, yet each per-tool score is assigned in ignorance of what else has been selected, which is the concern of ***(Q1)*** and is visible only when tools are scored as a whole. However, scoring the set jointly is still not enough, because the interaction among tools must itself depend on the cardinality of the set. A currency converter and a weather lookup belong together in the four-tool request above, but in a two-tool task they are co-invoked only under a contrived query such as "*convert my budget into yen and tell me whether it will rain in Tokyo*". On ToolBench the two co-occur in 0.4% of two-tool sets but in 23% of four-tool sets, a gap far wider than the mechanical increase in pair count with set size would explain. This underlies ***(Q2)*** and motivates cardinality-specific interaction modeling.

We therefore propose **HYSET**, short for **HY**peredge-based **SE**t-level **T**ool retrieval. Our main contributions are summarized as follows:

- **Formulation.** To the best of our knowledge, we are the first to recast tool retrieval for LLM agents as query-conditioned hyperedge prediction over a tool co-invocation hypergraph, making the tool set itself the unit of scoring. We further provide a unified view where existing tool-retrieval paradigms arise as restricted instances.
- **Method.** We design HYSET as a pre-selection module requiring no modification to the downstream agent, which scores candidate tool sets via cardinality-specific interaction matrices so that tool compatibility can vary with set size.
- **Experiments.** On ToolBench, HYSET consistently outperforms strong baselines from all three paradigms, with relative improvements of **11.6%** in COMP@5 and up to **13.1%** in end-to-end pass rate over the strongest baselines. It further transfers zero-shot to held-out tools and categories, and recovers **93.2%** of fully supervised performance with only **5** labeled examples per category.

2 Methodology

2.1 Preliminaries

Let $\mathcal{T} = \{t_1, \ldots, t_{N_\mathcal{T}}\}$ be a tool library. We assume each query x in the natural-language query space $\mathcal{X}$ can be fulfilled by jointly invoking a subset $E \subseteq \mathcal{T}$, and we assume access

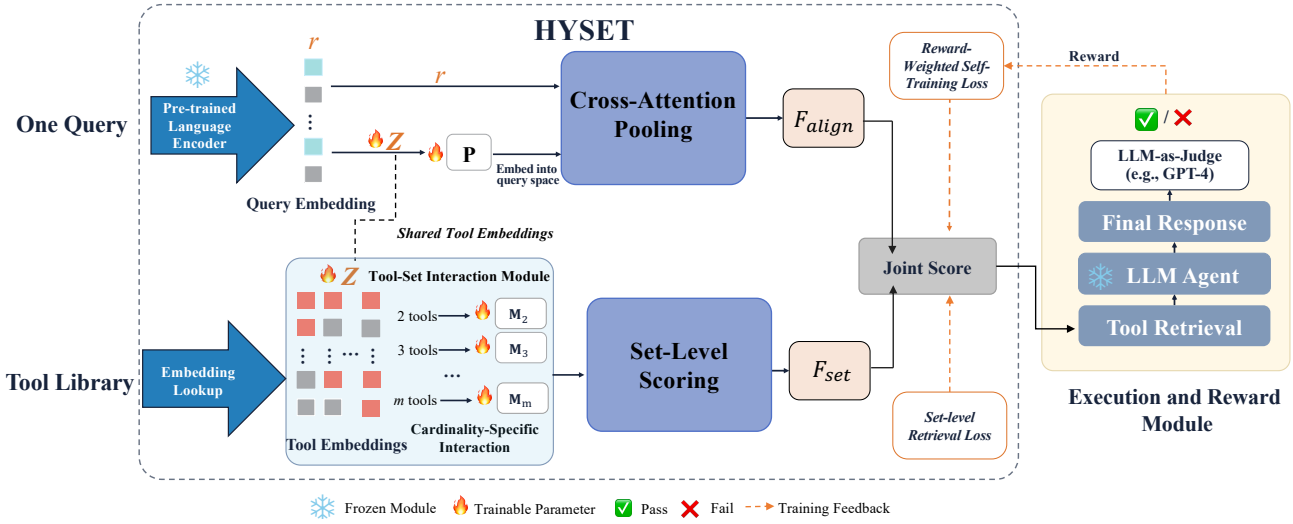


Figure 2: **HYSET framework.** HYSET consists of two components: query-set alignment, measuring relevance to the input query, and set-level scoring, treating each candidate set as a hyperedge and modeling cardinality-specific interactions among its tools.

to a training set $\mathcal{D}_{\text{tr}} = \{(x_i, E_i^\star)\}_{i=1}^N$, where $E_i^\star$ is the annotated tool set for x_i. Each $E_i^\star$ is the ground truth for x_i but need not be the unique feasible set, and it is unordered because invocation order is decided by the downstream agent rather than by the retriever. These sets admit a hypergraph representation (Battiston et al. 2020) that we call the tool co-invocation hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, whose node set $\mathcal{V} = \mathcal{T}$ collects all tools and whose hyperedges $\mathcal{E}$ are the observed $E_i^\star$, so the unit of supervision shifts from the relevance of an individual tool to the joint utility of an entire set. Writing $M = \max_i |E_i^\star|$ for the largest observed tool-set size, the admissible candidate hyperedge space $\mathcal{E}_M$ collects all $E \subseteq \mathcal{V}$ with $1 \le |E| \le M$, assuming that tool sets required at inference do not exceed M.

2.2 Problem Formulation

Set-level tool retrieval seeks a mapping $\mathcal{R} : \mathcal{X} \to \mathcal{E}_M$ returning the complete tool set for a query. Since $\mathcal{V} = \mathcal{T}$, every candidate set $E \in \mathcal{E}_M$ is a hyperedge over $\mathcal{V}$, so the task is query-conditioned hyperedge prediction. We model it through a parametric scoring family $F_\theta : \mathcal{X} \times \mathcal{E}_M \to \mathbb{R}$, where $F_\theta(x, E)$ measures the joint utility of selecting E for x, and estimate $\widehat{\theta} = \arg\min_{\theta \in \Theta} \mathcal{L}(\theta; \mathcal{D}_{\text{tr}})$ with $\mathcal{L}$ the empirical loss of Section 2.4. For a new query x_{new}, the learned scoring function $\widehat{F} = F_{\widehat{\theta}}$ predicts

$$\widehat{E}(x_{\text{new}}) = \arg\max_{E \in \mathcal{E}_M} F_{\widehat{\theta}}(x_{\text{new}}, E). \tag{1}$$

The resulting set $\widehat{E}(x_{\text{new}})$ is subsequently provided to the frozen downstream LLM agent to complete the user query.

2.3 A Unified View of Tool Retrieval

Under the hypergraph formulation and scoring family introduced in Section 2.2, major existing tool-retrieval paradigms can be subsumed by a unified set-scoring framework.

(i) **Semantic-matching retrievers** induce $F_{\text{match}} : \mathcal{X} \times \mathcal{E}_M \to \mathbb{R}$ with $F_{\text{match}}(x, E) = \sum_{t \in E} F_{\text{match}}(x, \{t\}) = \sum_{t \in E} s_{\text{match}}(x, t)$, where $\{t\} \in \mathcal{E}_M$ is the singleton hyperedge associated with tool t, and $s_{\text{match}}(x, t)$ is typically a lexical or dense query-tool matching score.

(ii) **Graph-enhanced retrievers** induce $F_{\text{graph}} : \mathcal{X} \times \mathcal{E}_M \to \mathbb{R}$ with $F_{\text{graph}}(x, E) = \sum_{t \in E} F_{\text{graph}}(x, \{t\}) = \sum_{t \in E} s_{\mathcal{G}}(x, t)$, where $s_{\mathcal{G}}(x, t)$ is typically the similarity between a query embedding and a graph-contextualized embedding of the individual tool t.

(iii) **Generative retrievers** induce $F_{\text{gen}} : \mathcal{X} \times \mathcal{E}_M \to \mathbb{R}$ with $F_{\text{gen}}(x, E) = \log \sum_{\pi \in \mathfrak{S}(E)} p_\phi(\pi, \text{EOS} \mid x)$, where $\mathfrak{S}(E)$ collects the $|E|!$ orderings of E, and p_ϕ is typically the autoregressive probability of a language model over tool-identifier tokens. Its factorization is given in Supplementary Material A.2.

The formulations above provide a unified mathematical view of existing paradigms. Supplementary Material A.2 derives the three induced scores and A.3 shows that they form a strict hierarchy of interaction orders.

2.4 The Framework of HYSET

Scoring Function Parameterization. We parameterize the scoring function $F_\theta : \mathcal{X} \times \mathcal{E}_M \to \mathbb{R}$ introduced in Section 2.2 through the decomposition

$$F_\theta(x_i, E) = F_{\text{set}}(E) + F_{\text{align}}(x_i, E), \tag{2}$$

where F_{set} models the internal interaction among the tools in E independently of the query, and F_{align} models the alignment between x_i and E. First, motivated by latent-space models of hypergraph data (Turnbull et al. 2019; Wu, Xu, and Zhu 2024; Hong, Xu, and Yu 2026), we parameterize the set-interaction term F_{set} as

$$F_{\text{set}}(E) = \sum_{1 \le a < b \le m} \mathbf{z}_{j_a}^\top \mathbf{M}_m \mathbf{z}_{j_b}, \tag{3}$$

Algorithm 1 Training and Inference of HYSET

Training input: Tool node set $\mathcal{V}$, training set $\mathcal{D}_{\text{tr}} = \{(x_i, E_i^\star)\}_{i=1}^N$, frozen query encoder $\mathbf{r}(\cdot)$, frozen agent $\mathcal{A}$, candidate-pool size K_{neg}, maximum cardinality M, execution limit R, and weights η, λ

Inference input: Query x_{new} and shortlist sizes $K_1 < K_{\text{pool}}$

Output: Learned scoring function $\widehat{F}$ and predicted tool set $\widehat{E}(x_{\text{new}})$

Training phase

repeat
  Sample a minibatch index set $\mathcal{B} \subseteq \{1, \dots, N\}$
  for each $i \in \mathcal{B}$ **do**
    Construct $\mathcal{C}_i \subseteq \mathcal{E}_M$ of size K_{neg} using the negative-sampling procedure in Section 2.4
    Compute $\widehat{E}_i$ from $\mathcal{C}_i$ via Eqs. (2) and (7)
    Run $\mathcal{A}$ on x_i using DFSDT (Qin et al. 2023) with tool set $\widehat{E}_i$ for at most R steps
    Obtain the task reward ρ_i from the execution result
  end for
  Update θ via Eq. (11)
until convergence
Obtain $\widehat{\theta} \leftarrow \theta$ and $\widehat{F} = F_{\widehat{\theta}}$

Inference phase

Construct the K_{pool}-tool shortlist $\mathcal{S}$ using Eqs. (12) and (13)
Compute $\widehat{E}(x_{\text{new}})$ via Eq. (14)
return $\widehat{F}$ and $\widehat{E}(x_{\text{new}})$

where $E = \{t_{j_1}, \dots, t_{j_m}\} \in \mathcal{E}_M$ is a candidate hyperedge of cardinality $m = |E|$, $\mathbf{z}_{j_a} \in \mathbb{R}^{d_z}$ is the learnable embedding of tool node t_{j_a}, $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d_z}$ collects all tool embeddings as rows, and $\mathbf{M}_m \in \mathbb{R}^{d_z \times d_z}$ is a symmetric interaction matrix shared across candidate hyperedges of cardinality m. A larger $F_{\text{set}}(E)$ indicates that the tools of E better conform to the learned patterns of co-invocation and functional complementarity. For $m \geq 2$, the pair contribution $\mathbf{z}_{j_a}^\top \mathbf{M}_m \mathbf{z}_{j_b}$ is allowed to vary with the cardinality of the set containing it, which is precisely what $\{\mathbf{M}_m\}_{m=2}^M$ is introduced to model. To formalize the distinction, we introduce the notion of a *pairwise-decomposable* set function based on the Möbius transform (Grabisch, Marichal, and Roubens 2000). Specifically, for a given function $f : \mathcal{E}_M \cup \{\varnothing\} \to \mathbb{R}$, the Möbius transform and its inverse are defined as

$$\tilde{f}(E) := \sum_{T \subseteq E} (-1)^{|E|-|T|} f(T), \quad f(E) = \sum_{T \subseteq E} \tilde{f}(T).$$

Intuitively, the Möbius coefficient $\tilde{f}(T)$, whose interaction order is $|T|$, measures the joint contribution attributable specifically to T after subtracting the contributions of its proper subsets. Hence, we call f *pairwise-decomposable* if $\tilde{f}(E) = 0$ whenever $3 \leq |E| \leq M$, meaning that it contains no interactions above order two. For instance, when $M \geq 3$, the third-order interaction of $S = \{t_i, t_j, t_k\}$ is

$$\tilde{F}_{\text{set}}(S) = \sum_{\{t_a, t_b\} \subseteq S} \mathbf{z}_a^\top (\mathbf{M}_3 - \mathbf{M}_2) \mathbf{z}_b.$$

Thus, $\mathbf{M}_3 - \mathbf{M}_2$ induces the third-order effect without an explicit three-way tensor. More generally, we have the following theorem:

Theorem 1. *Assume $|\mathcal{V}| \geq M + 2$. The set function $F_{\text{set}} : \mathcal{E}_M \cup \{\varnothing\} \to \mathbb{R}$ [1] is pairwise-decomposable if and only if*

$$\mathbf{z}_i^\top \mathbf{M}_2 \mathbf{z}_j = \cdots = \mathbf{z}_i^\top \mathbf{M}_M \mathbf{z}_j, \qquad \forall\, t_i, t_j \in \mathcal{V},\ i \neq j.$$

We prove Theorem 1 in Supplementary Material A.4. The theorem identifies the exact role of the cardinality-specific matrices. F_{set} reduces to a fixed pairwise model precisely when every pair receives the same score at all cardinalities from 2 to M. Otherwise, its cardinality dependence induces interactions above order two. Thus, although F_{set} is parameterized as a sum over pairs, it can represent structured joint effects at orders up to M and thereby capture multi-tool compatibility. These structured effects are generated by $O(Md_z^2)$ parameters, rather than the $O(d_z^M)$ parameters required by an unrestricted explicit M-way tensor. Second, we parameterize the query-set alignment F_{align} as

$$F_{\text{align}}(x_i, E) = \mathbf{r}(x_i)^\top \mathbf{s}(x_i, E), \tag{4}$$

where $\mathbf{r} : \mathcal{X} \to \mathbb{R}^{d_r}$ is a frozen pre-trained language model that maps query x_i to its embedding $\mathbf{r}(x_i)$ (Reimers and Gurevych 2019), and $\mathbf{s} : \mathcal{X} \times \mathcal{E}_M \to \mathbb{R}^{d_r}$ produces a query-conditioned representation $\mathbf{s}(x_i, E)$ of candidate hyperedge E. Specifically, we compute $\mathbf{s}(x_i, E)$ using cross-attention pooling with $\mathbf{r}(x_i)$ as the query and $\{\mathbf{P}\mathbf{z}_{j_k}\}_{k=1}^m$ as the keys and values (Vaswani et al. 2017),

$$\mathbf{s}(x_i, E) = \sum_{k=1}^m \alpha_k(x_i, E) \mathbf{P}\mathbf{z}_{j_k}, \tag{5}$$

where $\alpha_k(x_i, E) = \frac{\exp\big(\ell(x_i, t_{j_k})\big)}{\sum_{q=1}^m \exp\big(\ell(x_i, t_{j_q})\big)}$ is the normalized attention weight assigned to t_{j_k} within E, and $\ell(x_i, t_{j_k}) = \mathbf{r}(x_i)^\top \mathbf{P}\mathbf{z}_{j_k}$ is the matching score between query x_i and tool t_{j_k}. The matrix $\mathbf{P} \in \mathbb{R}^{d_r \times d_z}$ maps each tool embedding into the query space. Substituting Eq. (5) into Eq. (4) gives

$$F_{\text{align}}(x_i, E) = \sum_{k=1}^m \alpha_k(x_i, E) \ell(x_i, t_{j_k}). \tag{6}$$

For $m \geq 2$, each attention weight depends on all tools in E, so the alignment term is itself a query-conditioned set-level score (Supplementary Material A.5). The trainable parameters are $\theta = (\mathbf{Z}, \{\mathbf{M}_m\}_{m=2}^M, \mathbf{P})$, with $\mathbf{Z}$ shared by the two terms.

Training Objective. In practice, we construct the training objective of HYSET from negative-sampled set supervision and execution feedback, as depicted in Figure 2 and summarized in Algorithm 1. Since the admissible space of Section 2.1 contains $|\mathcal{E}_M| = \sum_{m=1}^M \binom{|\mathcal{V}|}{m}$ candidate hyperedges, exhaustive training is infeasible, and we use negative sampling both to build tractable candidate pools and to supply contrasting tool sets for the retrieval loss. For each query x_i with ground-truth hyperedge $E_i^\star$, we form the pool $\mathcal{C}_i = \{E_i^\star\} \cup \mathcal{N}_i$

[1] $F_{\text{set}}(E) := 0$ for $|E| \leq 1$.

with $\mathcal{N}_i \subseteq \mathcal{E}_M \setminus \{E_i^\star\}$ and $|\mathcal{N}_i| = K_{\text{neg}} - 1$ distinct negatives, drawn from three sources in the fixed proportion $50\%/30\%/20\%$: size-matched sets sampled uniformly from the $|E_i^\star|$-tool subsets of $\mathcal{V}$, in-batch negatives taken from the ground-truth sets of the other queries of the minibatch (Oord, Li, and Vinyals 2018), and hard negatives obtained by replacing one or two tools of $E_i^\star$ with their nearest neighbors under $\mathbf{Z}$ (Karpukhin et al. 2020; Robinson et al. 2020). Supplementary Material A.8 states the sampling procedure in full, gives an adaptive mixture recovering this one as the constant special case, and measures the resulting false-negative rate. Supplementary Material C.8 sweeps the proportion. Given $\mathcal{C}_i$, we score each candidate using Eq. (2) and select

$$\widehat{E}_i = \arg\max_{E \in \mathcal{C}_i} F_\theta(x_i, E), \tag{7}$$

The frozen LLM agent $\mathcal{A}$ then processes x_i under the DFSDT search procedure (Qin et al. 2023) with its tool set restricted to $\widehat{E}_i$ for at most R steps, and its answer $\widehat{y}_i$ receives the execution reward $\rho_i = \text{ExecScore}(x_i, \widehat{y}_i) \in [0, 1]$, whose definition, configuration and refresh schedule are given in Supplementary Material A.9. The candidate pool induces the retrieval loss

$$\mathcal{L}_{\text{ret}}(\theta) = -\sum_{i \in \mathcal{B}} \log \frac{\exp F_\theta(x_i, E_i^\star)}{\sum_{E \in \mathcal{C}_i} \exp F_\theta(x_i, E)}, \tag{8}$$

which raises $F_\theta(x_i, E_i^\star)$ relative to $F_\theta(x_i, E)$ for every $E \in \mathcal{N}_i$. We further use execution feedback to reinforce successful model-selected sets through the reward-weighted self-training loss

$$\mathcal{L}_{\text{self}}(\theta) = -\sum_{i \in \mathcal{B}} \rho_i \log \frac{\exp F_\theta(x_i, \widehat{E}_i)}{\sum_{E \in \mathcal{C}_i} \exp F_\theta(x_i, E)}, \tag{9}$$

where $\widehat{E}_i$ and ρ_i are treated as constants within each parameter update. Combining Eqs. (8) and (9), the full objective is

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{ret}}(\theta) + \eta \mathcal{L}_{\text{self}}(\theta) + \lambda \sum_{m=2}^{M} \|\mathbf{M}_m\|_F^2, \tag{10}$$

where $\eta > 0$ controls the execution-feedback term and $\lambda > 0$ controls the regularization of the cardinality-specific interaction matrices. We estimate

$$\widehat{\theta} = \arg\min_{\theta \in \Theta} \mathcal{L}(\theta) \quad \text{s.t.} \quad \|\mathbf{z}_j\|_2 = 1 \quad \forall\, t_j \in \mathcal{V}, \tag{11}$$

where $\|\mathbf{z}_j\|_2 = 1$ fixes the scale of each tool embedding.

Set-Level Inference. The training procedure yields the learned scoring function $\widehat{F} = F_{\widehat{\theta}}$ with estimated parameters $\widehat{\theta} = (\widehat{\mathbf{Z}}, \{\widehat{\mathbf{M}}_m\}_{m=2}^M, \widehat{\mathbf{P}})$. Given a new query $x_{\text{new}} \in \mathcal{X}$, directly solving Eq. (1) requires maximizing $\widehat{F}$ over $\mathcal{E}_M$. Since $\mathcal{E}_M$ contains all subsets of $\mathcal{V}$ with cardinality between 1 and M, its size grows combinatorially with $|\mathcal{V}|$, making exhaustive enumeration infeasible. We therefore approximate Eq. (1) by exploiting the additive decomposition of $\widehat{F}$ in Eq. (2) to construct a two-stage procedure consisting of per-tool retrieval and set-level reranking (Zheng et al. 2024, 2026). In the first stage, we score each individual tool $t_j \in \mathcal{V}$ against x_{new}. Substituting the singleton hyperedge $E = \{t_j\}$ into Eq. (2) yields

$$\widehat{F}(x_{\text{new}}, \{t_j\}) = \mathbf{r}(x_{\text{new}})^\top \widehat{\mathbf{P}} \widehat{\mathbf{z}}_j. \tag{12}$$

Let $\mathcal{S}_0 \subseteq \mathcal{V}$ collect the K_1 highest-scoring tools under Eq. (12). We score every remaining tool by its learned complementarity with $\mathcal{S}_0$, $g(t_j) = \max_{t_i \in \mathcal{S}_0} \widehat{\mathbf{z}}_j^\top \widehat{\mathbf{M}}_M \widehat{\mathbf{z}}_i$, and write $t_{[1]}, t_{[2]}, \ldots$ for the tools of $\mathcal{V} \setminus \mathcal{S}_0$ in decreasing order of g. The shortlist is then

$$\mathcal{S} = \mathcal{S}_0 \cup \{t_{[1]}, \ldots, t_{[K_{\text{pool}} - K_1]}\}, \tag{13}$$

where $M \le K_{\text{pool}} \ll |\mathcal{V}|$, so the second term admits a tool of low individual relevance but high complementarity with $\mathcal{S}_0$. In the second stage, we evaluate Eq. (2) on every candidate in the reduced space $\mathcal{E}_M(\mathcal{S}) = \{E \subseteq \mathcal{S} : 1 \le |E| \le M\}$, which approximates Eq. (1) by

$$\widehat{E}(x_{\text{new}}) = \arg\max_{E \in \mathcal{E}_M(\mathcal{S})} \widehat{F}(x_{\text{new}}, E). \tag{14}$$

Eq. (14) compares sets of different cardinality, and since F_{set} aggregates $\binom{m}{2}$ pairs while F_{align} is a convex combination of per-tool scores, the two scale differently in m, which the cardinality-specific $\mathbf{M}_m$ absorbs. The reduced space contains $\sum_{m=1}^{M} \binom{K_{\text{pool}}}{m}$ hyperedges, independently of $|\mathcal{V}|$. As $\widehat{E}(x_{\text{new}})$ is unordered and of variable size, rank-based metrics require a second output, obtained by greedy marginal gain under the same $\widehat{F}$ with $A_0 = \varnothing$ and $A_k = A_{k-1} \cup \{\arg\max_{t \in \mathcal{S} \setminus A_{k-1}} \widehat{F}(x_{\text{new}}, A_{k-1} \cup \{t\})\}$, so every step after the first scores a candidate jointly with the tools already selected. Rank-based metrics are computed from $\widehat{E}^{(K)}(x_{\text{new}}) = (t_{\pi(1)}, \ldots, t_{\pi(K)})$, while $\widehat{E}(x_{\text{new}})$ is passed to the agent $\mathcal{A}$. Supplementary Material A.10 reports the predicted cardinality distribution and a calibrated variant, A.11 the inference path in pseudocode, and E.3 a comparison against exhaustive maximization.

3 Experiments

3.1 Experimental Setup

Task, datasets and baselines. We evaluate offline retrieval by testing whether the retrieved set covers $E^\star$, and end-to-end execution by testing whether the frozen agent solves the query when restricted to that set. Our primary benchmark is ToolBench (Qin et al. 2023), whose official filtering retains 13,860 callable API endpoints forming $\mathcal{V}$ and 200,311 instructions with ground-truth API sets. We train on the combined official training portions and evaluate on the six held-out test sets comprising 600 queries. Since training and evaluation share one library, we also report train-test overlap and results restricted to unseen tool sets. To check that the conclusions are not library-specific, we further evaluate on UltraTool (Huang et al. 2024), whose 2,032 tools across 22 domains are disjoint from ToolBench and in which only 6.1% of the ground-truth sets are singletons, against 20.4% on ToolBench. We compare against six baselines spanning the three paradigms: BM25 (Robertson and Zaragoza 2009), Contriever (Izacard et al. 2021), ToolLLaMA-Retriever (Qin et al.

Method	Sup.	Recall@5 ↑	NDCG@5 ↑	COMP@5 ↑	Pass Rate ↑	
					GPT-4	Human
BM25	A	41.02	39.68	22.38	19.21 ± 0.44	15.37 ± 1.47
Contriever	A	41.08	38.30	22.04	18.80 ± 0.41	14.17 ± 1.42
ToolLLaMA-Ret	A	68.88	70.12	61.11	64.05 ± 0.68	61.83 ± 1.98
ToolRerank	A	80.71 ± 0.17	81.63 ± 0.08	69.93 ± 0.07	63.39 ± 0.72	58.30 ± 2.01
COLT	A	77.07 ± 0.10	83.84 ± 0.06	68.98 ± 0.19	63.54 ± 0.66	59.36 ± 2.01
ToolGen	A	81.41	84.76	70.01	62.30 ± 0.70	59.04 ± 2.01
HYSET (BERT)	A+R	$\mathbf{84.75 \pm 0.09}$	$\mathbf{88.99 \pm 0.16}$	$\mathbf{77.55 \pm 0.12}$	$\mathbf{69.69 \pm 0.61}$	$\mathbf{66.17 \pm 1.93}$
HYSET (Qwen2.5)	A+R	$\mathbf{88.61 \pm 0.12}$	$\mathbf{91.07 \pm 0.18}$	$\mathbf{78.13 \pm 0.16}$	$\mathbf{71.11 \pm 0.58}$	$\mathbf{69.92 \pm 1.87}$

Table 1: Main results on ToolBench. All values are percentages, best in boldface. Sup.: A denotes annotated tool sets only, while A+R additionally uses the execution reward of Eq. (9). Error bars follow Supplementary Material B.8. Results at a cutoff 3 are reported in Supplementary Table 13.

2023), ToolRerank (Zheng et al. 2024), COLT (Qu et al. 2024) and ToolGen (Wang et al. 2024). Supplementary Material B.1 and B.2 give dataset, overlap and baseline details.

Evaluation metrics. We report Recall@K and NDCG@K for retrieval quality and COMP@K (Qu et al. 2024) for set completeness with $K \in \{3, 5\}$, together with GPT-4 and Human Pass Rate under the official ToolEval protocol (Qin et al. 2023). Every method passes at least as many tools to the agent as HYSET does, so the tool budget of the main results is generous to the baselines. Since the rank-based metrics are computed from a length-K ranking rather than from the variable-size set $\widehat{E}(x)$ that reaches the agent, Section 3.5 scores $\widehat{E}(x)$ directly. Supplementary Material B.3 to B.6 and B.8 give the definitions, the tool-budget rule, the annotation protocol, the significance tests and the error-bar accounting.

Implementation details. We instantiate HYSET with two frozen backbones, a BERT-base retriever (Qin et al. 2023) and a tuned Qwen2.5-1.5B retriever (Wang et al. 2024), and run every method under the same frozen ToolLLaMA-2-7B-v2 agent (Qin et al. 2023) and DFSDT protocol. Only $\theta = (\mathbf{Z}, \{\mathbf{M}_m\}_{m=2}^{M}, \mathbf{P})$ is trained, with 13.59M parameters under the BERT configuration at $d_z = 768$, including 10.64M for $\mathbf{Z}$, compared with 109.5M parameters in the frozen encoder. The query encoder, the tool encoder that initializes $\mathbf{Z}$ from API descriptions, and the agent are all frozen. We set $M = 5$, the largest annotated set size, with $K_1 = 15$, $K_{\text{pool}} = 20$ and $K_{\text{neg}} = 64$, so reranking scores 21,699 candidate sets per query. Execution feedback rewards $N_{\text{sub}} = 5{,}000$ training queries and refreshes them every $T_{\text{ref}} = 20{,}000$ steps, capping it at 20,000 rollouts and as many judge calls, all served from the cached StableToolBench API mirror rather than from live endpoints. Every configuration uses three seeds and early stopping on validation Recall@5. Supplementary Material B.7, B.9 and B.10 give the remaining hyperparameters, all prompts and the reproducibility details.

Variant	R@5	C@5	PR
HYSET (Full)	$\mathbf{84.75 \pm 0.09}$	$\mathbf{77.55 \pm 0.12}$	$\mathbf{69.69 \pm 0.61}$
(a) Component removal			
w/o F_{set}	72.04 ± 0.18	67.36 ± 0.09	57.97 ± 0.73
w/o Exec. fb.	82.14 ± 0.07	77.02 ± 0.07	65.14 ± 0.63
(b) Interaction-matrix design			
$\mathbf{M}_m = \mathbf{I}$	75.05 ± 0.16	68.68 ± 0.15	59.92 ± 0.71
Shared $\mathbf{M}$	78.43 ± 0.20	73.66 ± 0.08	64.37 ± 0.69
w/o Reg.	83.17 ± 0.07	75.34 ± 0.13	68.93 ± 0.64

Table 2: Ablation of HYSET (BERT). R@5: Recall@5; C@5: COMP@5; PR: GPT-4 Pass Rate. All values are percentages.

3.2 Main Results

Table 1 shows that HYSET outperforms every baseline on every retrieval and end-to-end metric. Retrieval gains over the strongest baseline reach 15.3% relative for the BERT configuration and 17.8% for Qwen, and are largest on COMP, where COMP@5 improves by 10.8% and 11.6% relative over ToolGen. Hyperedge scoring therefore recovers complete tool sets rather than merely reranking individual tools, and the two configurations differ by at most 5.7% relative, so the gains come from set-level modeling rather than from the backbone. Every margin is significant at the 5% level under a paired bootstrap with 10^4 resamples for the retrieval metrics and the exact McNemar test for Pass Rate. COMP@5 gives $p < 10^{-4}$ with a 95% confidence interval of $[4.03, 11.05]$ for the difference, while the two Pass Rate margins are weakest at $p = 1.6 \times 10^{-3}$ and $p = 1.5 \times 10^{-2}$, the human one being the only margin that would not survive a Bonferroni correction over all eight tests at the 1% level (B.6). The Sup. column of Table 1 records that the two HYSET rows additionally use execution feedback while every baseline is trained on annotations alone, so the regimes must be separated before the end-to-end margin is read. Trained on annotations alone, HYSET (BERT) still reaches 77.02% COMP@5, an improvement of 10.0% relative over ToolGen, but its Recall@5 falls to 82.14% and its advantage over ToolGen narrows to 0.9% relative. Retraining

the two strongest baselines with the identical reward, judge and 20,000-rollout budget raises ToolGen to 83.12% Recall@5, 70.94% COMP@5 and 66.85% Pass Rate, and ToolLLaMA-Retriever to 66.42% Pass Rate. Execution feedback thus helps them by about as much as it helps HYSET, whose own Pass Rate gains 7.0% relative over its annotation-only value of 65.14%. The GPT-4 Pass Rate margin consequently falls from 8.8% to 4.3% relative while the COMP@5 margin falls only from 10.8% to 9.3%. Supplementary Material C.2 reports every method under both regimes and C.3 the execution budget.

3.3 Ablation Study

Table 2 ablates the two aspects that carry our claims. In panel (a), removing F_{set} costs 13.1% of COMP@5 and 16.8% of Pass Rate, so scoring the set as a whole is the main source of the gains. Removing execution feedback costs 6.5% of Pass Rate but only 0.7% of COMP@5, because $\mathcal{L}_{\text{self}}$ rewards sets that execute successfully rather than the annotated ones. Completeness therefore comes from set-level modeling, and execution feedback is worth its cost only when end-to-end success is the target. In panel (b), a learned shared matrix improves every metric over identity matrices by up to 7.4% on Pass Rate, using cardinality-specific matrices adds a further 7.1%, and regularization a final 2.9% on COMP@5 (all tabulated in C.6), confirming that tool compatibility must vary with set size. Supplementary Material C.4, C.7 and C.8 report the robustness of Pass Rate to the choice of judge, the Qwen backbone and the negative-sampling sweep.

Removing parts of HYSET shows that its components matter but not that this set-level model is needed rather than any set-level model. We therefore compare against three further families on the same frozen BERT backbone, shortlist and training data, with every learned scorer trained under the same annotation-only objective so that only the set scorer differs. Predicting the cardinality and truncating a dense ranking reaches 64.72% COMP@5, maximal marginal relevance 65.93% and greedy facility-location maximization 67.16%, so a fixed pairwise redundancy or coverage term is worth little, as Theorem 1 predicts for scores whose interactions vanish above order two. A general set encoder helps considerably more: DeepSets reaches 69.85% and a Set Transformer with two induced-set-attention blocks reaches 72.41%, the latter using 21.4M scorer parameters and 19.3 ms per query against 13.6M and 12.4 ms for HYSET. Neither recovers HYSET, which leads the Set Transformer by 6.4% relative under the same objective, and the variant isolating why is a shared cardinality-agnostic $\mathbf{M}$, which already matches the Set Transformer at 72.87% with half its parameters. Only the cardinality index on $\{\mathbf{M}_m\}$ separates it from HYSET. Under the same 20,000-rollout budget, the two encoders move to 70.61% and 73.15% COMP@5, leaving HYSET ahead by 9.8% and 6.0% relative, so the advantage over unrestricted set encoders is a modeling rather than a supervision effect. Supplementary Material C.11 reports both regimes with latencies and parameter counts.

Setting	Sup.	Overall		
		R@5	C@5	PR
In-domain reference				
ID	Full	$\mathbf{84.75 \pm 0.09}$	$\mathbf{77.55 \pm 0.12}$	$\mathbf{69.69 \pm 0.61}$
Zero-shot transfer				
UT	0-shot	79.48 ± 0.19	70.71 ± 0.11	63.64 ± 0.79
UC	0-shot	75.35 ± 0.09	65.28 ± 0.14	58.23 ± 0.85
CD	0-shot	72.83 ± 0.17	61.96 ± 0.18	54.85 ± 0.92
Few-shot target adaptation (UC)				
UC	1-shot	76.30 ± 0.07	67.23 ± 0.08	59.88 ± 0.88
UC	5-shot	79.13 ± 0.10	71.40 ± 0.14	63.93 ± 0.81
UC	10-shot	81.58 ± 0.12	74.38 ± 0.10	66.48 ± 0.77
UC	Full	83.57 ± 0.11	76.60 ± 0.19	68.80 ± 0.74

Table 3: Generalization and data efficiency of HYSET (BERT). UT denotes held-out tools, UC denotes held-out categories, and CD denotes cross-domain transfer. All values are percentages.

3.4 Generalization and Data Efficiency

Table 3 evaluates transfer within a fixed library while withholding supervision for selected tools, whole categories, or the target domain. Performance decreases as the shift grows, yet cross-domain transfer retains 79.9% of in-domain set completeness. Five examples per target category recover 93.2% of fully supervised performance. A stricter protocol that also excludes held-out tools from every negative pool produces comparable results. The fixed-library splits isolate query and label shifts rather than tool novelty. When the complete tool set is unseen during training, HYSET improves COMP@5 over the strongest baseline by 13.1%. The improvement rises to 15.9% when an unseen tool pair is also required. On UltraTool, whose library is disjoint from ToolBench, HYSET improves COMP@5 by 11.5% over ToolGen. Direct transfer from ToolBench without target training retains 77.9% of the performance obtained by training on UltraTool. Supplementary Material D.1 to D.4 give the full setup and results.

3.5 The Set Delivered to the Agent

The metrics above use rankings of fixed length, whereas the agent receives $\widehat{E}(x)$, a set whose size varies, with about half as many tools as a top-5 retriever. Direct evaluation shows that HYSET improves coverage of the complete required set by about 7% over ToolGen and nearly doubles exact match accuracy. It also achieves a slightly higher Pass Rate at its predicted cardinality than when forced to return five tools. The maximizer and greedy ranking agree on nearly 90% of queries, indicating that their main difference is set size. Supplementary Material C.10 gives the full metrics and the breakdown by predicted cardinality.

3.6 Cost, Sensitivity and Scalability

Inference costs 12.4 ms and 3.21 GB per query at $|\mathcal{V}| = 13{,}860$, against 6.5 ms and 1.71 GB for the single dense pass of ToolLLaMA-Retriever and 378.4 ms and 6.83 GB

for ToolGen. Training costs 43.2 GPU hours and USD 186 in judge calls on two RTX 4090 GPUs, against 38.6 GPU hours for ToolGen. Enlarging $|\mathcal{V}|$ with distractor tools leaves Recall@5 and COMP@5 stable, since the reranking cost depends on K_{pool} and M alone. Varying η, λ and K_{pool} one at a time leaves HYSET stable, and the matrix expanding the shortlist in Eq. (13) changes COMP@5 by at most 0.40 points, compared with the 1.07-point loss caused by removing the expansion. Supplementary Material E and F give the sweeps and the runtime analyses.

4 Conclusion

We argued that LLM tool retrieval is inherently a set-level problem and recast it as query-conditioned hyperedge prediction over a tool co-invocation hypergraph. We proposed HYSET, a module that scores candidate tool sets as a whole and captures how tool compatibility varies with set size. Extensive experiments show that HYSET consistently outperforms strong baselines and generalizes to held-out tools and categories under zero-shot and few-shot settings. Extending it to dynamically growing tool libraries is a natural direction for future work.

References

Battiston, F.; Cencetti, G.; Iacopini, I.; Latora, V.; Lucas, M.; Patania, A.; Young, J.-G.; and Petri, G. 2020. Networks beyond pairwise interactions: Structure and dynamics. *Physics reports*, 874: 1–92.

Grabisch, M.; Marichal, J.-L.; and Roubens, M. 2000. Equivalent representations of set functions. *Mathematics of Operations Research*, 25(2): 157–178.

Hong, X.; Xu, S.; and Yu, Z. 2026. HYVINT: Intensity-Driven Hypergraph Generation with Variational Representations. *arXiv.org*.

Huang, S.; Zhong, W.; Lu, J.; Zhu, Q.; Gao, J.; Liu, W.; Hou, Y.; Zeng, X.; Wang, Y.; Shang, L.; et al. 2024. Planning, Creation, Usage: Benchmarking LLMs for Comprehensive Tool Utilization in Real-World Complex Scenarios. In *Annual Meeting of the Association for Computational Linguistics*, 4363–4400. Association for Computational Linguistics.

Izacard, G.; Caron, M.; Hosseini, L.; Riedel, S.; Bojanowski, P.; Joulin, A.; and Grave, E. 2021. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*

Karpukhin, V.; Oguz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; and Yih, W.-t. 2020. Dense passage retrieval for open-domain question answering. In *Conference on Empirical Methods in Natural Language Processing*, 6769–6781. Association for Computational Linguistics.

Liu, N. F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; and Liang, P. 2023. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12: 157–173.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv.org*.

Patil, S. G.; Zhang, T.; Wang, X.; and Gonzalez, J. E. 2023. Gorilla: Large language model connected with massive apis. *Neural Information Processing Systems*, 37: 126544–126565.

Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.-T.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, 9695–9717.

Qu, C.; Dai, S.; Wei, X.; Cai, H.; Wang, S.; Yin, D.; Xu, J.; and Wen, J.-R. 2024. Towards completeness-oriented tool retrieval for large language models. In *International Conference on Information and Knowledge Management*, 1930–1940. ACM.

Reimers, N.; and Gurevych, I. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Conference on Empirical Methods in Natural Language Processing*, 3980–3990. Association for Computational Linguistics.

Robertson, S.; and Zaragoza, H. 2009. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Emerald.

Robinson, J.; Chuang, C.-Y.; Sra, S.; and Jegelka, S. 2020. Contrastive learning with hard negative samples. *International Conference on Learning Representations*.

Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Zettlemoyer, L.; Cancedda, N.; and Scialom, T. 2023. Toolformer: Language models can teach themselves to use tools. *Neural Information Processing Systems*, 36: 68539–68551.

Shen, Y.; Song, K.; Tan, X.; Li, D.; Lu, W.; and Zhuang, Y. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Neural Information Processing Systems*, 36: 38154–38180.

Tay, Y.; Tran, V. Q.; Dehghani, M.; Ni, J.; Bahri, D.; Mehta, H.; Qin, Z.; Hui, K.; Zhao, Z.; Gupta, J.; et al. 2022. Transformer memory as a differentiable search index. *Neural Information Processing Systems*, 35: 21831–21843.

Turnbull, K.; Lunag'omez, S.; Nemeth, C.; and Airoldi, E. 2019. Latent space modeling of hypergraph data. *Journal of the American Statistical Association*, 119(548): 2634–2646.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Neural Information Processing Systems*, 30.

Wang, R.; Han, X.; Ji, L.; Wang, S.; Baldwin, T.; and Li, H. 2024. Toolgen: Unified tool retrieval and calling via generation. In *International Conference on Learning Representations*, volume 2025, 73473–73498.

Wu, S.; Xu, G.; and Zhu, J. 2024. A general latent embedding approach for modeling non-uniform high-dimensional sparse hypergraphs with multiplicity. *arXiv preprint arXiv:2410.12108*.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2022. React: Synergizing reasoning and acting in language models. *International Conference on Learning Representations*.

Zheng, Y.; Li, P.; Liu, W.; Liu, Y.; Luan, J.; and Wang, B. 2024. Toolrerank: Adaptive and hierarchy-aware reranking for tool retrieval. In *International Conference on Language*

Resources and Evaluation, 16263–16273. European Language Resources Association (ELRA) and ICCL.

Zheng, Y.; Zhang, Z.; Ma, C.; Yu, Y.; Zhu, J.; Wu, Y.; Xu, T.; Dong, B.; Zhu, H.; Huang, R.; et al. 2026. Skillrouter: Skill routing for llm agents at scale. *arXiv.org*.