# TOOL RETRIEVERS ARE UNDERESTIMATED: ANNOTATION EXPANSION REVEALS TRUE CAPABILITY

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In open-world scenarios with massive and evolving tool repositories, tool-augmented large language models rely on a retriever to surface relevant tools for a given query. Because such repositories often contain many tools that implement the same functionality, a single query can often be resolved by several distinct but functionally equivalent tool combinations, making the natural query-to-tool mapping inherently one-to-many. However, existing tool retrieval benchmarks annotate each query with a single relevant tool combination, collapsing this one-to-many mapping into a rigid one-to-one annotation and causing valid retrieved tools to be misjudged as failures. To address this, we propose **TOOLEX** (Tool Equivalent eXpansion), a framework that automatically discovers and annotates the tool combinations functionally equivalent to the labeled ones. Applied to the 7,360-query TOOL-DE benchmark, TOOLEX finds that 67.9% of sub-queries admit equivalent alternatives, expanding the singular ground truth to an average of 5.3 valid combinations per query. Using the expanded benchmark TOOLEQ, we re-evaluate eight base retrievers and two fine-tuned variants; metrics on TOOLEQ rise substantially over TOOL-DE, showing that one-to-one annotation systematically underestimates retrievers and that 30–47% of the reported fine-tuning gain is an evaluation artifact rather than genuine improvement. Applying the same pipeline to skill retrieval on SKILLRET further confirms that the one-to-one problem extends beyond tool retrieval.

## 1 INTRODUCTION

As LLM agents move toward open-domain deployment, tool retrieval becomes a prerequisite for execution: an agent cannot invoke a capability it fails to retrieve. This has motivated dedicated tool retrieval benchmarks such as ToolRET (Shi et al., 2025), TOOL-DE (Lu et al., 2025), and MetaTool (Huang et al., 2024b). These benchmarks are typically built by aggregating tools and task demonstrations from large tool-use corpora, including ToolBench (Qin et al., 2024), ToolACE (Liu et al., 2024a), and ToolEyes (Ye et al., 2024). However, this construction process inevitably introduces tools with overlapping functionality under different names, providers, or interfaces. The resulting query-to-tool relation is therefore naturally one-to-many: a single query may be solvable by multiple distinct but functionally equivalent tool combinations. Existing benchmarks ignore this equivalence and instead assign each query a single annotated ground-truth combination. Consequently, a retriever that returns an equivalent but unannotated tool is scored as wrong despite producing a valid retrieval. We call this the **one-to-one annotation problem**.

Consider a concrete query that asks to "retrieve a list of historical events that occurred on a specific calendar date." A dense retriever ranks candidate tools by similarity, and the library contains several that implement exactly this capability: `Get Historical Events of Today`, `/v1/historicalevents`, `Historical Events by API Ninjas`, and `v1_historicalevents`. Yet only `QueryHistoryToday` is annotated as the ground truth, so every other equivalent tool is scored as a miss despite producing a valid retrieval, inducing a systematic underestimate of retrieval quality.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To close this gap, we propose TOOLEX, a fully automated annotation pipeline that discovers functionally equivalent tool combinations without human labeling. TOOLEX operates in three stages (Figure 1): an LLM first decomposes each composite query into $k$ atomic sub-queries, one per tool in the ground-truth combination; for each sub-query, a dense retriever (Qwen3-Embedding-4B (Zhang et al., 2025)) retrieves the top-20 candidate tools, which an LLM then verifies to determine whether they satisfy the capability specified by the sub-query; finally, the verified per-sub-query tool sets are merged via Cartesian product, ranked with Reciprocal Rank Fusion (RRF) (Cormack et al., 2009), and filtered by an LLM dependency check that removes combinations violating inter-tool constraints. Applied to the 7,360-query TOOL-DE evaluation set, which uses the same queries as TOOLRET (Shi et al., 2025) but augments the tool documentation as in TOOL-DE (Lu et al., 2025), TOOLEX expands the average annotation from 1 to 5.3 tool combinations per query, with 67.9% of sub-queries receiving at least one additional equivalent tool beyond the original ground truth.

Using these expanded annotations, we construct **TOOLEQ**, a new benchmark that credits a retrieval whenever it matches *any* valid equivalent combination, and re-evaluate eight base IR models and two fine-tuned tool retrievers. TOOLEQ corrects a systematic underestimation: across the eight base retrievers it raises NDCG@10 by 5–7 pp on average over TOOL-DE (Table 2), recovering the credit withheld from retrievals of functionally equivalent but unannotated tools. This underestimate also inflates the apparent fine-tuning gain: correcting it narrows the gap between fine-tuned and base retrievers on TOOLEQ (0.6B: $+6.4$ pp $\rightarrow$ $+4.5$ pp; 4B: $+7.2$ pp $\rightarrow$ $+3.8$ pp), so part of the gain reported on TOOL-DE reflects the evaluation protocol rather than only improved retrieval. We further find that applying the same pipeline to the SKILLRET benchmark (Cho et al., 2026) confirms the one-to-one annotation problem is ecosystem-wide.

In summary, our contributions are fourfold:

- We unveil the one-to-one annotation problem in existing tool retrieval benchmarks, demonstrating that scoring functionally equivalent tools as misses systematically underestimates retriever capability.
- We propose TOOLEX, a fully automated three-stage pipeline that discovers functionally equivalent tool combinations without human annotation, and construct **TOOLEQ** as a new evaluation framework that credits any valid equivalent.
- By expanding singular ground truths to an average of 5.3 valid combinations per query, TOOLEQ corrects the underestimation: eight base retrievers gain 5–7 pp NDCG@10 on average over TOOL-DE. Correcting the metric also exposes that fine-tuning gains are partly an evaluation artifact, as the fine-tuned-to-base gap shrinks once equivalents are credited.
- Extending TOOLEX to SKILLRET shows the one-to-one problem is ecosystem-wide, confirming that functional equivalence is not specific to tool repositories.

## 2 RELATED WORK

**Tool Learning Ecosystem: Use and Retrieval Benchmarks.** Tool learning equips large language models (LLMs) with external tools, enabling them to execute actions and solve practical tasks as agents. A broad line of benchmarks evaluates tool-use capabilities across diverse execution scenarios, including ToolBench (Qin et al., 2024), APIBank (Li et al., 2023), ToolAlpaca (Qiao et al., 2024), AppBench (Yang et al., 2024), GTA (Mao et al., 2024), MetaTool (Chang et al., 2024), ToolEyes (Ye et al., 2024), ToolACE (Liu et al., 2024a), and APIGen (Liu et al., 2024b). However, in open-domain scenarios where tool/skill repositories scale to tens of thousands of items, feeding all documentation directly into the LLM context window becomes fundamentally infeasible. Tool and skill retrieval have thus emerged as indispensable prerequisites for open-domain agent execution, giving rise to dedicated benchmarks such as TOOLRET (Shi et al., 2025), TOOL-DE (Lu et al., 2025), and SKILLRET (Cho et al., 2026). To construct these retrieval benchmarks, standard pipelines collect large pools of candidate tools or skills, randomly sample seed tool subsets, and prompt LLMs to synthesize corresponding user queries. Crucially, this bottom-up synthesis pipeline inherently couples each generated query *exclusively* with its seeded tools, creating an artificial **one-to-one annotation problem** that completely ignores functionally equivalent alternatives co-existing in the candidate repository.

**Tool Retrieval Methodologies.** To tackle the open-domain retrieval bottleneck, recent methodologies focus on optimizing retrieval quality along several complementary technical routes. A major line of work employs dense retriever fine-tuning via contrastive learning with negative sampling (Karpukhin et al., 2020; Xiong et al., 2021), as well as specialized vector space alignments (Moon et al., 2024) and generative identifiers (Wang et al., 2025), to directly bridge the encoding gap between query intent and tool documentation. Complementary to representation learning, query reformulation and decomposition techniques (Fang & Glass, 2026; Liu et al., 2025; Sengupta et al., 2026; Huang et al., 2024a) rewrite complex user intents into atomic sub-goals or leverage structural graph modeling (Gao et al., 2025) to better match tool capabilities. More recently, joint optimization paradigms utilize reinforcement learning to co-optimize retrieval policies with agent execution in open-world environments (Huang et al., 2026). Crucially, while these approaches significantly advance *how* queries and tools are encoded, reformulated, or retrieved, they universally evaluate predictions against a single, rigidly annotated target. As a result, whenever a model successfully surfaces a functionally equivalent alternative, traditional benchmarks falsely count it as a false negative. Our work is strictly orthogonal: rather than introducing another retrieval algorithm, we reform the evaluation paradigm itself by expanding singular ground truths into valid sets of functionally equivalent tool combinations.

# 3 TOOLEX: EXPANDING EQUIVALENT GROUND-TRUTHS

In this section, we develop TOOLEX, a framework that corrects the one-to-one annotation problem in existing tool retrieval benchmarks. First, Section 3.1 analyzes the limitations of current benchmarks and formalizes the one-to-one annotation problem. Next, Section 3.2 introduces the TOOLEX annotation pipeline for equivalent ground-truth expansion. Finally, Section 3.3 presents the benchmarks TOOLEQ along with a new evaluation metric.

## 3.1 LIMITATIONS OF CURRENT RETRIEVAL BENCHMARKS

Let $\mathcal{Q}$ denote a set of user queries and $\mathcal{T}$ a heterogeneous tool library. Existing benchmarks (Shi et al., 2025; Lu et al., 2025) annotate each query $q \in \mathcal{Q}$ with a single ground-truth tool combination $C^* = \{t_1, \ldots, t_k\} \subseteq \mathcal{T}$. This one-to-one convention rests on a hidden assumption: that each query admits exactly one correct tool combination. In open-world repositories, this assumption fails. Libraries assembled from independent sources inevitably contain *functionally equivalent* tools—APIs and services that perform the same operation under different names, providers, or documentation styles—so the true set of valid combinations for a query is

$$\mathcal{C}_q = \{C \subseteq \mathcal{T} \mid C \text{ can fully solve } q\}, \tag{1}$$

with $|\mathcal{C}_q| \gg 1$ whenever the library contains equivalents. Standard metrics treat only $C^*$ as relevant and count every $C \in \mathcal{C}_q \setminus \{C^*\}$ as a false negative, systematically underestimating the true capability of tool retrievers. We call this the **one-to-one annotation problem**.

This annotation convention has a direct consequence on the *evaluation* side: a retriever that correctly identifies an unlabeled equivalent receives no credit, while one that overfits to the single annotated combination is rewarded. Measured gains on benchmarks such as TOOL-DE may therefore substantially overstate real improvement, and one-to-one benchmark scores alone cannot distinguish genuine retrieval ability from annotation artifact. TOOLEX targets this structural source of bias by approximating $\mathcal{C}_q$ automatically, without human annotation.

## 3.2 ANNOTATION PIPELINE

TOOLEX operates in three stages, as illustrated in Figure 1. In the experiments reported here, the pipeline is applied to the TOOL-DE evaluation set derived from ToolRet (Shi et al., 2025) and TOOL-DE (Lu et al., 2025), where ToolRet provides the query–tool structure and TOOL-DE provides expanded tool documentation used for retrieval.

**Stage 1: Multi-Tool Query Decomposition.** The goal of this stage is to rewrite each composite query $q$ as $k$ atomic, provider-agnostic sub-queries $\{s_1, \ldots, s_k\}$, one per tool in the ground-truth combination $C^*$, so that each $s_i$ can retrieve functionally equivalent tools without revealing their

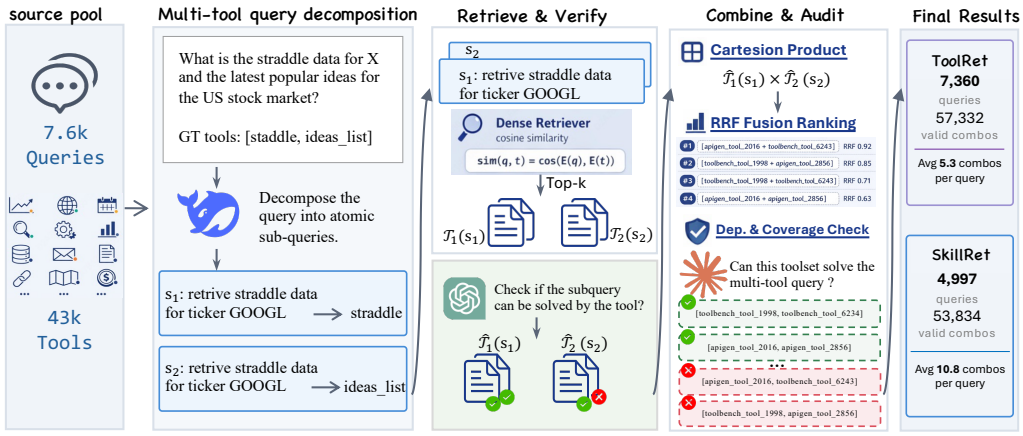


Figure 1: **The TOOLEX annotation pipeline.** Stage 1 decomposes each query into atomic sub-queries via DeepSeek-V3.2; Stage 2 retrieves top-$K$ candidates per sub-query with Qwen3-Embedding-4B and verifies functional alignment via GPT-4o-mini; Stage 3 assembles Cartesian combinations, ranks them with RRF, and filters via Claude-Sonnet-5 dependency checking, yielding valid equivalent combinations.

names. DeepSeek-V3.2 (DeepSeek-AI et al., 2025) performs the decomposition, given the query, the task instruction, and the documentation of all $k$ ground-truth tools, and is instructed to minimize the semantic gap between each sub-query and its corresponding tool documentation (the full prompt is given in Listing 1 in Appendix A). Stage 1 conditions the decomposition on the benchmark-provided ground-truth tools, ensuring that the number of sub-queries equals the number of tools in $C^*$ and establishing a one-to-one correspondence between sub-queries and the components of the combination to be expanded. Each sub-query describes two aspects: (a) the *functionality* (the precise operation, e.g., retrieve, calculate, search) and (b) the *entity* (the parameter values the query context can supply, e.g., a stock ticker, a calendar date). The functionality term aligns the sub-query with equivalent tool documentation, while the entity term lets the Stage 2 verifier confirm the caller can supply the inputs the candidate tool requires.

**Stage 2: Candidate Retrieval and Functional Verification.** The goal of this stage is, for each sub-query $s_i$, to collect every tool in $\mathcal{T}$ that is functionally equivalent to the annotated ground-truth tool $t_i$ and could substitute for it in solving $s_i$. We first retrieve the top-$K$ candidate tools for $s_i$ with Qwen3-Embedding-4B (Zhang et al., 2025), then verify each candidate $t$ with GPT-4o-mini (full prompt in Listing 2 in Appendix A). The verifier receives the sub-query $s_i$, the candidate's full documentation as raw JSON (to avoid information loss from field parsing), and the ground-truth tool's documentation as a reference, and judges whether $t$ performs the same core operation as the reference and whether its output covers what $s_i$ requires with parameters that $s_i$ can supply. A candidate is accepted only when both conditions hold; under genuine uncertainty the verifier defaults to "no" to favor precision. The verified set is $\hat{\mathcal{T}}_i = \{t \in \mathcal{T} : \text{LLM verifies } t \text{ for } s_i\}$, and by construction $t_i$ always passes verification (it is compared against itself), guaranteeing $C^* \in$ the final expansion. A potential concern is the overlap between the candidate retriever and the evaluation: Qwen3-Embedding-4B also appears among the evaluated models, so the expansion could be biased toward candidates that this retriever ranks highly. Two observations mitigate this concern: the candidate pool is broad — after Stage 1 decomposition, Qwen3-Embedding-4B attains a sub-query-level Recall@20 of 0.854 against the original one-to-one annotations — and the measured underestimation does not hinge on this choice, as the non-Qwen retrievers in Table 2, including the sparse BM25s, show NDCG@10 gaps of +4.6–6.3 pp versus +7.5–8.5 pp for the Qwen3 series.

**Stage 3: Combination Assembly and Audit.** This stage assembles the per-sub-query verified sets $\hat{\mathcal{T}}_i$ into query-level tool combinations that genuinely solve $q$, discarding those that do not. Candidate

Table 1: Annotation statistics produced by the TOOLEX pipeline. Side-by-side statistics are reported for the evaluation sets of TOOLEQ and SKILLEQ.

| Statistic | ToolEq | SkillEq |
|---|---|---|
| Queries annotated | 7,360 | 4,997 |
| Sub-queries (Stage 1) | 13,506 | 8,347 |
| Verified items (Stage 2) | 57,332 | 43,934 |
| Avg. items / sub-query | 4.2 | 5.26 |
| Valid combinations (Stage 3) | 39,087 | 53,834 |
| Avg. combos / query | 5.3 | 10.8 |

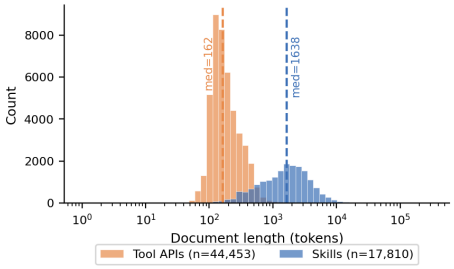


Figure 2: Documentation length distribution for Tool API docs (orange) and Skill docs (blue).

combinations are formed by Cartesian product,

$$\hat{\mathcal{C}}_q = \hat{\mathcal{T}}_1 \times \hat{\mathcal{T}}_2 \times \cdots \times \hat{\mathcal{T}}_k, \tag{2}$$

and ranked by Reciprocal Rank Fusion (RRF (Cormack et al., 2009)) over per-sub-query retrieval scores. Claude-Sonnet-5 then audits each combination (full prompt in Listing 3 in Appendix A) against the original query, the task instruction, and the sub-query breakdown, with the ground-truth set as a reference, checking two conditions: **(1) Function coverage** — the combination collectively performs every operation the query requires, using the sub-query breakdown as a checklist; **(2) Dependency consistency** — when one sub-query's output (e.g., an authentication token, resource ID, or session handle) is consumed by another, the corresponding tools come from the same platform, since runtime values cannot cross platforms. Combinations passing both checks form the valid equivalent set $\hat{\mathcal{C}}_q \supseteq \{C^*\}$.

**Annotation Results.** Table 1 summarizes the expansion annotation statistics on the evaluation sets of TOOLEQ and SKILLEQ. On the TOOLEQ evaluation set, the pipeline verifies 4.2 equivalent tools per sub-query on average, yielding 5.3 valid equivalent combinations per query. On SKILLEQ, the same pipeline verifies 5.26 equivalent skills per sub-query and yields 10.8 valid combinations per query. These results confirm that functional equivalence is widespread: 67.9% of sub-queries admit at least one equivalent tool beyond the annotated one, and 75.2% of queries admit at least one equivalent combination beyond the annotated ground truth. Because Stage 2 retrieves only the top-20 candidates per sub-query, any equivalent outside this pool remains undiscovered, so these prevalence figures are lower bounds on the true extent of functional equivalence.

**Human Validation.** To assess annotation quality, we audit 1,000 final combinations per benchmark (stratified by query category and number of sub-queries) with two independent reviewers and a third adjudicating disagreements; the protocol and failure-mode distribution appear in Appendix B. The resulting precision is 85.9% for TOOLEQ and 95.7% for SKILLEQ. The dominant tool-side failure mode is missed inter-tool dependency constraints (98.7% of invalid tool combinations) — e.g., adding a memo reminder requires logging in first, feasible only when the login and reminder APIs come from the same platform — whereas skills rarely interact at run time.

### 3.3 BENCHMARK AND EVALUATION METRIC

Using the expanded annotations on the evaluation set, we construct **TOOLEQ**, a new tool retrieval benchmark in which each query is paired with a *set of pre-annotated valid tool combinations* rather than just one labeled ground-truth.

Let $L_q$ be the ordered retrieval list for query $q$, and let $s_q(t)$ denote the retrieval score of tool $t$ under the system being evaluated. For a single reference combination $C$, write $\mathrm{NDCG}(L_q, C)$, $\mathrm{Recall}(L_q, C)$, $\mathrm{Comp}(L_q, C)$ for the three standard IR metrics (defined in §4.1). Given the expanded reference set $\hat{\mathcal{C}}_q = \{C_1, \ldots, C_M\}$ of valid equivalent combinations, TOOLEQ treats every $C_j \in \hat{\mathcal{C}}_q$ as a labeled ground-truth answer for $q$, and evaluates against the reference that the retrieval

Table 2: Main results pairing the one-to-one benchmark TOOL-DE against the new benchmark **TOOLEQ** (one-to-many, best-match metric). All values are in %; each metric cell reports *old / new*, i.e. TOOL-DE (gray) / TOOLEQ (black). $\Delta$ is the mean gain (TOOLEQ − TOOL-DE) averaged over N@10/R@10/C@10 within each category (pp); † oracle-assisted sub-query diagnostic with RRF.

| Model | Code | | | | Web | | | | Customized | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N@10 | R@10 | C@10 | Δ | N@10 | R@10 | C@10 | Δ | N@10 | R@10 | C@10 | Δ |
| *Query-level* | | | | | | | | | | | | |
| BM25s | 45.2/49.5 | 58.6/62.7 | 57.2/61.3 | **+4.2** | 28.5/38.1 | 35.9/45.8 | 24.0/30.1 | **+8.5** | 44.4/48.6 | 50.1/54.1 | 39.0/42.1 | **+3.8** |
| gte-Qwen2-1.5B | 41.4/46.0 | 53.2/56.9 | 51.2/54.9 | **+4.0** | 37.1/43.5 | 47.3/53.0 | 29.8/33.8 | **+5.4** | 48.1/53.9 | 56.6/62.0 | 43.3/48.0 | **+5.3** |
| e5-mistral-7b | 41.9/47.4 | 55.9/60.4 | 53.9/58.4 | **+4.8** | 31.9/39.7 | 41.4/48.9 | 27.5/32.6 | **+6.8** | 43.6/49.3 | 50.8/58.4 | 39.3/46.0 | **+6.7** |
| GritLM-7B | 24.0/27.5 | 33.4/37.4 | 32.0/36.1 | **+3.9** | 29.5/36.2 | 38.1/45.6 | 24.4/29.7 | **+6.5** | 41.8/46.6 | 49.1/54.8 | 37.6/42.7 | **+5.2** |
| NV-Embed-v1 | 48.8/52.1 | 63.0/63.2 | 60.0/63.5 | **+3.4** | 32.5/38.1 | 39.6/45.6 | 24.0/28.8 | **+5.5** | 39.6/44.5 | 45.6/51.1 | 34.4/39.2 | **+5.1** |
| Qwen3-Embedding-0.6B | 48.4/53.8 | 62.3/66.0 | 59.9/63.6 | **+4.3** | 37.3/45.2 | 46.3/52.9 | 29.4/33.8 | **+6.3** | 42.5/51.7 | 48.9/58.3 | 39.1/47.2 | **+8.9** |
| Qwen3-Embedding-4B | 53.5/60.2 | 70.7/74.2 | 69.2/72.5 | **+4.5** | 38.7/46.7 | 47.9/54.9 | 30.4/35.5 | **+6.7** | 42.4/53.1 | 50.9/61.7 | 40.2/49.0 | **+10.1** |
| Qwen3-Embedding-8B | 52.6/59.1 | 68.1/71.1 | 66.2/69.0 | **+4.1** | 40.8/49.1 | 49.5/56.7 | 31.9/37.1 | **+6.9** | 43.9/54.7 | 52.5/62.1 | 41.9/49.9 | **+9.5** |
| Tool-Embed-0.6B | 52.1/56.0 | 65.7/67.5 | 64.0/65.8 | **+2.5** | 42.3/49.1 | 52.5/58.1 | 35.6/39.7 | **+5.5** | 53.1/59.0 | 61.8/65.9 | 47.6/50.4 | **+4.3** |
| Tool-Embed-4B | 55.6/59.9 | 70.6/72.6 | 68.7/70.7 | **+2.8** | 44.6/51.3 | 54.5/60.2 | 37.4/41.8 | **+5.6** | 55.9/60.2 | 62.1/65.5 | 47.5/50.2 | **+3.5** |
| *Sub-query-level (†)* | | | | | | | | | | | | |
| BM25s† | 71.6/80.2 | 83.6/88.7 | 80.8/85.7 | **+6.2** | 32.0/40.5 | 41.8/49.4 | 29.6/34.8 | **+7.1** | 50.6/55.7 | 58.6/62.9 | 46.8/50.3 | **+4.3** |
| gte-Qwen2-1.5B† | 74.1/82.7 | 87.2/90.5 | 85.0/88.2 | **+5.0** | 37.7/45.3 | 49.2/55.9 | 33.9/39.1 | **+6.5** | 56.8/62.7 | 64.8/69.6 | 50.8/54.4 | **+4.8** |
| e5-mistral-7b† | 74.2/83.1 | 85.8/90.1 | 82.4/86.8 | **+5.9** | 30.7/40.0 | 41.2/50.3 | 29.7/35.7 | **+8.1** | 49.7/55.9 | 58.9/64.6 | 46.3/50.5 | **+5.4** |
| GritLM-7B† | 78.3/86.4 | 90.5/93.1 | 87.5/90.3 | **+4.5** | 33.3/42.5 | 43.2/52.1 | 31.2/37.5 | **+8.1** | 57.3/62.7 | 66.2/70.4 | 53.3/56.0 | **+4.1** |
| NV-Embed-v1† | 78.5/87.8 | 91.6/94.9 | 88.3/91.5 | **+5.3** | 31.7/41.1 | 41.0/50.1 | 29.0/35.4 | **+8.3** | 57.3/64.0 | 64.9/69.3 | 51.6/54.6 | **+4.7** |
| Qwen3-Embedding-0.6B† | 76.2/84.7 | 88.5/92.0 | 84.5/88.1 | **+5.2** | 38.5/47.1 | 50.1/57.3 | 34.3/39.6 | **+7.0** | 48.1/56.8 | 56.6/64.4 | 48.2/53.7 | **+7.3** |
| Qwen3-Embedding-4B† | 76.5/86.5 | 91.0/94.2 | 88.1/91.3 | **+5.5** | 38.6/47.6 | 48.8/56.3 | 33.9/39.3 | **+7.3** | 49.8/59.7 | 59.6/67.9 | 49.2/55.9 | **+8.3** |
| Qwen3-Embedding-8B† | 75.4/83.8 | 87.8/90.7 | 81.3/84.1 | **+4.7** | 48.0/61.7 | 59.9/70.4 | 48.7/58.8 | **+11.5** | 46.9/55.8 | 55.1/62.8 | 39.5/44.9 | **+7.3** |
| Tool-Embed-0.6B† | 81.2/87.1 | 90.8/92.4 | 86.6/88.2 | **+3.0** | 53.5/65.4 | 65.7/73.9 | 53.8/61.5 | **+9.3** | 49.4/55.7 | 56.1/61.4 | 38.3/41.9 | **+5.1** |
| Tool-Embed-4B† | 82.1/87.2 | 90.8/92.5 | 86.4/88.2 | **+2.8** | 54.2/65.8 | 67.0/75.3 | 55.4/63.4 | **+9.3** | 54.4/60.7 | 61.4/66.3 | 42.3/45.7 | **+4.9** |

itself matches best:

$$C_q^{\dagger} = \arg\max_{C \in \hat{\mathcal{C}}_q} \sum_{t \in C} s_q(t),$$
$$M_q^{\text{TOOLEQ}} = M(L_q, C_q^{\dagger}), \qquad M \in \{\text{NDCG, Recall, Comp}\}. \tag{3}$$

All combinations in $\hat{\mathcal{C}}_q$ share the same cardinality (one tool per sub-query), so the alignment score $\sum_{t \in C} s_q(t)$ is comparable across references. A retrieval is therefore credited as soon as its ranked list aligns with one of the pre-annotated valid alternatives, instead of being penalized for missing the benchmark's originally chosen combination.

## 4 EXPERIMENTS

This section first presents the experimental settings in §4.1, followed by the main results in §4.2, a downstream task evaluation on ToolBench (Qin et al., 2024) in §4.4 to verify that TOOLEX-expanded annotations improve end-to-end agent performance, and an extended experiment on skill retrieval in §4.3.

### 4.1 EXPERIMENTAL SETUP

All models are evaluated on both TOOL-DE (Lu et al., 2025) (original, one-to-one annotation) and **TOOLEQ** (multi-set, best-match metric from Equation 3).

**Retrieval Methods.** We evaluate two retrieval methods that differ in how queries are formulated and results are aggregated:

- **Query-level:** the multi-tool query is encoded and matched against tool documents as a single retrieval unit. This is the standard approach used by prior benchmarks.
- **Sub-query-level diagnostic (†):** we reuse the Stage 1 decomposition from the annotation pipeline to study how atomic capability descriptions affect ranking. Because this decomposition uses the benchmark-provided tool count and reference tools, it is an oracle-assisted upper bound, not a deployable retrieval setting. Each sub-query retrieves an independent top-$K$ candidate list, and

the per-sub-query results are merged via Reciprocal Rank Fusion (RRF (Cormack et al., 2009)) — with the same damping constant $\kappa$=60 as in Stage 3 — to produce the final query-level ranking:

$$s_q(t) \;=\; \sum_{i\,:\,t \in \mathrm{top}K(s_i)} \frac{1}{\kappa + r_i(t)}, \tag{4}$$

where $r_i(t)$ is the 0-based rank of tool $t$ in the top-$K$ list of sub-query $s_i$; we set $K = 20$ per sub-query, yielding at most $k \times 20$ candidates.

**Baselines.** We evaluate TOOL-EMBED-{0.6B, 4B}, trained with TOOL-DE data and eight representative retrieval models on both TOOL-DE and TOOLEQ:

- **Sparse retriever:** BM25s (Stefanović, 2024).
- **Dense retrievers:** gte-Qwen2-1.5B-instruct (Li et al., 2024), e5-mistral-7b-instruct (Wang et al., 2023), GritLM-7B (Muennighoff et al., 2024), NV-Embed-v1 (Lee et al., 2024), and Qwen3-Embedding (Zhang et al., 2025) (0.6B, 4B, 8B).

**Metrics.** We adopt three widely used IR metrics to evaluate tool retrieval performance: (i) *NDCG@$K$* ($N@K$), which considers both the relevance of retrieved tools and their ranking positions; (ii) *Recall@$K$* ($R@K$), which measures the proportion of target tools successfully retrieved within the top-$K$ results; and (iii) *Comprehensiveness@$K$* ($C@K$), which assigns $C@K = 1$ if all target tools are included in the top-$K$ results and 0 otherwise. We report $K = 10$ per query category and average. On TOOLEQ, scores follow the best-match metric (Equation 3).

### 4.2 MAIN RESULTS

**Observation 1: One-to-one annotation systematically underestimates tool retrievers.** Table 2 pairs TOOL-DE (gray) with TOOLEQ (black) for eight base retrievers under query-level retrieval, and all eight score higher under TOOLEQ. Averaged over Code, Web, and Customized, the NDCG@10 gap runs from +4.6 pp (NV-Embed-v1) to +8.5 pp (Qwen3-Embedding-8B), mean +6.5 pp; BM25s, e5-mistral-7b, gte-Qwen2-1.5B, and GritLM-7B each show a 5.0–6.3 pp gap, so the correction is not confined to any single model family. The pattern is uniform across metrics: Comprehensiveness@10, averaged over categories, improves for every model, e.g., BM25s 40.1 → 44.5 and Qwen3-Embedding-4B 46.6 → 52.3, so the recovered credit is not a metric-specific artifact. Each retriever's top-$K$ list already contains functionally equivalent tools, but TOOL-DE leaves them unlabeled and so never scores them; TOOLEQ corrects this by crediting any equivalent combination (Equation 3), so the old–new gap measures the underestimation that TOOLEQ recovers.

**Observation 2: The underestimation tracks category-level tool redundancy.** Using the Qwen3-Embedding-0.6B TOOLEQ vs. TOOL-DE NDCG@10 delta, the Customized category benefits most (+9.2 pp: 42.5 → 51.7), Web moderately (+7.9 pp: 37.3 → 45.2), and Code the least (+5.4 pp: 48.4 → 53.8), reflecting that customized and web libraries aggregate numerous functionally similar APIs across providers while code repositories expose more distinctive interfaces. The same ordering holds across the full model pool: averaged over the eight base retrievers, the Web NDCG@10 gap is +7.5 pp and Customized +7.0 pp, versus +5.0 pp for Code, confirming that the ranking of categories is a property of the libraries rather than of any single retriever.

**Observation 3: Query decomposition unlocks large gains without changing retriever parameters.** We report a controlled sub-query decomposition diagnostic ($^\dagger$) that reuses the Stage 1 sub-queries from the annotation pipeline, while keeping the retriever unchanged and retrieving over the full tool library. Under this setting, zero-shot Qwen3-Embedding-0.6B$^\dagger$ gains 12.6 pp NDCG@10 over its composite-query counterpart on TOOLEQ (averaged across categories), and Tool-Embed-4B$^\dagger$ reaches 70.2 NDCG@10. The effect is not specific to dense retrievers: on the one-to-one TOOL-DE scores, the decomposition gain ranges from +10.1 pp (Qwen3-Embedding-4B) to +24.5 pp (GritLM-7B), with the sparse BM25s also improving by +12.0 pp. These results suggest that current retrievers already encode much of the required tool semantics, and that a large share of the remaining error comes from semantic interference within composite queries rather than from a lack of retriever capacity.

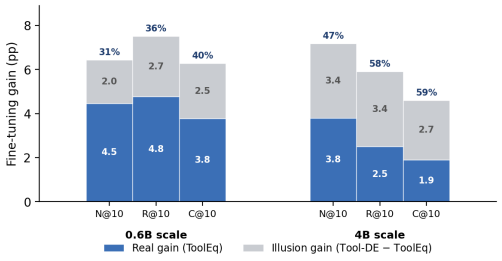


Figure 3: Training gain illusion: The gray excess over blue is the illusion gain — 47% of the reported improvement at 4B is an evaluation artifact.

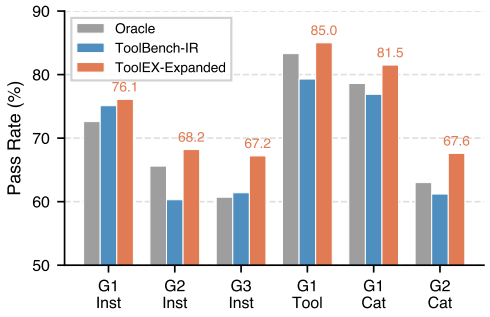


Figure 4: Downstream task pass rates (%) on ToolBench across six instruction categories.

Table 3: Skill retrieval on the full library (17,810 skills), pairing SKILLRET (one-to-one) against **SKILLEQ** (one-to-many). All values are in %; each metric cell reports *old / new*, i.e., SKILLRET (gray) / SKILLEQ (black). $\Delta$ is the mean gain SKILLEQ−SKILLRET averaged over all nine N@$\{5, 10, 20\}$/R@$\{5, 10, 20\}$/C@$\{5, 10, 20\}$ cells (pp).

| | NDCG | | | Recall | | | Comprehensiveness | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | @5 | @10 | @20 | @5 | @10 | @20 | @5 | @10 | @20 | $\Delta$ |
| BM25s | 39.9/51.2 | 42.0/53.9 | 43.8/55.8 | 43.5/55.4 | 49.4/62.6 | 55.6/68.8 | 29.2/38.8 | 34.3/46.0 | 40.1/52.7 | **+11.9** |
| gte-Qwen2-1.5B | 46.6/59.6 | 49.1/62.2 | 50.8/63.9 | 50.4/62.6 | 57.1/69.3 | 63.0/74.9 | 35.3/45.4 | 41.0/52.0 | 46.5/58.5 | **+12.1** |
| e5-mistral-7b | 40.8/51.3 | 43.0/53.8 | 44.8/55.6 | 44.7/55.2 | 50.8/61.7 | 56.8/67.9 | 30.0/38.6 | 35.3/44.9 | 40.8/51.2 | **+10.4** |
| GritLM-7B | 51.5/64.1 | 54.1/66.6 | 55.8/68.2 | 55.4/66.7 | 62.3/73.4 | 68.2/78.5 | 39.7/49.8 | 46.1/57.1 | 52.3/63.9 | **+11.4** |
| NV-Embed-v1 | 55.0/63.6 | 57.6/66.0 | 59.4/67.7 | 58.2/66.2 | 65.2/72.6 | 71.0/77.9 | 41.8/48.8 | 48.7/55.9 | 54.7/62.2 | **+7.7** |
| Qwen3-Embedding-0.6B | 47.7/58.7 | 49.9/60.8 | 51.5/62.5 | 50.7/60.6 | 56.7/66.1 | 62.1/71.8 | 34.8/42.3 | 39.7/48.3 | 45.1/54.6 | **+9.7** |
| Qwen3-Embedding-4B | 46.2/61.2 | 48.6/63.2 | 50.3/64.7 | 50.6/63.4 | 57.0/68.8 | 62.9/73.6 | 35.4/45.2 | 40.2/50.2 | 45.4/55.8 | **+12.2** |
| Qwen3-Embedding-8B | 48.0/59.8 | 50.2/61.9 | 51.8/63.4 | 51.1/61.5 | 57.1/66.9 | 62.4/72.3 | 34.4/43.2 | 39.9/49.0 | 45.5/55.5 | **+10.4** |
| SkillRet-0.6B | 50.7/61.9 | 53.2/64.6 | 54.9/66.3 | 54.4/65.4 | 61.2/72.4 | 67.0/78.0 | 38.6/48.3 | 45.3/56.7 | 51.8/63.8 | **+11.2** |
| SkillRet-8B | 72.0/80.6 | 74.6/82.9 | 76.3/84.1 | 76.3/83.5 | 83.2/89.3 | 88.8/93.3 | 62.9/71.1 | 72.7/80.8 | 80.7/87.6 | **+7.3** |

**Observation 4: 30–47% of the apparent fine-tuning gain is an evaluation artifact.** When evaluated on the original TOOL-DE benchmark, fine-tuning yields apparent gains: Tool-Embed-0.6B improves over Qwen3-Embedding-0.6B by 6.4 pp in NDCG@10 (averaged over Code/Web/Customized), and the 4B model by 7.2 pp. However, under TOOLEQ the same gaps shrink to 4.5 pp at 0.6B and 3.8 pp at 4B (Figure 3). The inflation is most pronounced at 4B, where 47% of the reported gain vanishes once equivalents are credited; on Code the sign even reverses (Tool-DE +2.1 pp vs. ToolEq −0.3 pp). Notably, both models still score *higher in absolute terms* under TOOLEQ (Tool-Embed-4B reaches 57.1 vs. 52.0 on TOOL-DE), so the expansion does not penalize fine-tuned models; it only removes the spurious share of their advantage. This pattern indicates that a non-trivial fraction of the measured gain comes from alignment to the single annotated target that TOOL-DE privileges.

**Observation 5: Equivalent tools are already retrieved but left uncredited.** In the Qwen3-Embedding-4B ranking, the best-matching equivalent tool appears in the top-10 for 79.4% of queries with at least one verified equivalent — slightly above the GT tool itself (76.1%), versus 9.2% for a random negative — and for 14.5% of these queries the GT tool falls outside the top-10 while an equivalent appears within it. These are exactly the retrievals that TOOL-DE counts as failures but TOOLEQ correctly credits.

### 4.3 EXTENDED EXPERIMENT: SKILL RETRIEVAL

To verify generality beyond tools, we apply the TOOLEX pipeline (Section 3.2) to skill retrieval on SKILLRET (Cho et al., 2026); the expansion statistics appear in Table 1. We evaluate SkillRet-Embedding-0.6B, 8B, fine-tuned on SKILLRET training data, alongside the eight retrieval models of Section 4.1. Table 3 pairs SKILLRET (one-to-one) against **SKILLEQ** (one-to-many).

**Observation 6: The one-to-one annotation problem generalizes to skill retrieval.** SKILLEQ yields consistent gains over the original SKILLRET across all models; e.g., SkillRet-0.6B improves from 53.2 to 64.6 NDCG@10 (+11.4 pp) and zero-shot Qwen3-Embedding-0.6B from 49.9 to 60.8 (+10.9 pp), confirming that the one-to-one annotation problem is not limited to tool retrieval.

**Observation 7: Skill fine-tuning provides genuine improvement, and documentation complexity explains the contrast.** Unlike tool retrieval, the fine-tuned SkillRet-8B still substantially outperforms zero-shot Qwen3-8B *after* expansion: 82.93 vs. 61.89 NDCG@10 (+21.0 pp). This contrast stands in sharp relief against the tool-retrieval result, where the analogous gap collapses to +3.0 pp. We hypothesize that the difference stems from documentation length (Figure 2): skill documents contain rich, structured descriptions of capabilities, invocation patterns, and examples, so fine-tuning on them teaches task-relevant distinctions that survive annotation expansion, whereas the shorter and more uniform tool API documents leave fine-tuning to mainly memorize the annotated API identity itself.

### 4.4 DOWNSTREAM TASK EVALUATION ON TOOLBENCH

We evaluate downstream pass rates on 568 ToolBench tasks in three settings: **Oracle**, **ToolBench-IR**, and **ToolEX-expanded**, where the provided APIs are replaced by functionally equivalent combinations from our pipeline; all settings use DeepSeek-V3.2 as the resolver agent, and the six instruction categories follow the G1/G2/G3 splits of ToolBench (Qin et al., 2024). Figure 4 reports the results. ToolEX-expanded achieves the highest pass rate in all six categories, exceeding Oracle by +3.5–6.5 pp and ToolBench-IR by +1.0–6.9 pp (Figure 4). This is not paradoxical: some oracle APIs in the ToolBench snapshot have expired or become obsolete, so following them verbatim fails on those tasks, whereas an agent with multiple verified equivalents can fall back when one invocation fails. We view this as evidence that many discovered equivalents are practically usable, not as proof that ToolEX surpasses a fully specified execution oracle.

### 4.5 DISCUSSION: BENCHMARK IMPLICATIONS

TOOLEQ does not make evaluation more permissive; it restores credit for functionally valid solutions already present in the repository but omitted by one-to-one annotation, so the TOOL-DE–TOOLEQ gap measures annotation-induced underestimation rather than an arbitrary metric shift. Exact-reference evaluation remains useful for reproducibility, but TOOLEQ evaluates *functional sufficiency*, which better matches deployment. The consistent gains on both SKILLEQ and ToolBench suggest a broader mismatch between one-to-one annotation and open-world tool use; we therefore recommend that future tool-retrieval benchmarks report both exact-reference and equivalent-aware scores, so that retrievers are judged on the solutions they can actually deliver rather than on a single arbitrarily chosen annotation.

## 5 CONCLUSION

We identified a one-to-many annotation problem in tool retrieval benchmarks: functionally equivalent tools are common, yet existing benchmarks annotate a single ground-truth combination per query, so valid retrievals are scored as failures. We proposed TOOLEX, a three-stage pipeline that discovers and annotates functionally equivalent tool combinations automatically, and constructed TOOLEQ, which credits a retrieval as soon as it matches any pre-annotated valid combination. Re-evaluating existing retrievers on TOOLEQ showed that one-to-one annotation systematically underestimates them (mean +6.5 pp NDCG@10 across eight base models) and that 30–47% of the apparent fine-tuning gain on TOOL-DE is an evaluation artifact. Extending the framework to SkillRet confirmed that the problem is ecosystem-wide.

**Future Work.** Extending TOOLEX to tool-use benchmarks where tools can be practically executed would enable runtime verification of functional equivalence. Active-learning loops could scale the pipeline to larger libraries by prioritizing high-uncertainty candidates. Finally, designing retrievers that return diverse equivalent sets opens a new direction for tool-augmented agents (Shen et al., 2024; Patil et al., 2024).

**AI Use Statement.** In this work, we used generative AI tools for improving the readability of the text, LaTeX typesetting, and organizing the released code; we have not used generative AI tools for designing or running experiments, analyzing results, or formulating the claims, and the remaining tasks listed in the ICLR AI Policy for Authors are not applicable to this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

## REFERENCES


Shicheng Chang, Ailiang Wang, Sicheng Wang, Chuang Ju, Sui Mai, and Xiaofeng Zhang. Meta-Tool: A benchmark for large language models to determine what to call and how to call. *arXiv preprint arXiv:2404.00943*, 2024.

Hongcheol Cho, Ryangkyung Kang, and Youngeun Kim. Skillret: A large-scale benchmark for skill retrieval in llm agents. *arXiv preprint*, 2026.

Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *SIGIR*, 2009. doi: 10.1145/1571941.1572114.

DeepSeek-AI, Aixin Liu, Aoxue Mei, and et al. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

Wei Fang and James Glass. Beyond single-shot: Multi-step tool retrieval via query planning, 2026. URL https://arxiv.org/abs/2601.07782.

Linfeng Gao, Yaoxiang Wang, Minlong Peng, Jialong Tang, Yuzhe Shang, Mingming Sun, and Jinsong Su. Tool graph retriever: Exploring dependency graph-based tool retrieval for large language models, 2025. URL https://arxiv.org/abs/2508.05152.

Shouzheng Huang, Meishan Zhang, Baotian Hu, and Min Zhang. Toolomni: Enabling open-world tool use via agentic learning with proactive retrieval and grounded execution, 2026. URL https://arxiv.org/abs/2604.13787.

Tenghao Huang, Dongwon Jung, Vaibhav Kumar, Mohammad Kachuee, Xiang Li, Puyang Xu, and Muhao Chen. Planning and editing what you retrieve for enhanced tool learning. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 975–988, Mexico City, Mexico, June 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.61. URL https://aclanthology.org/2024.findings-naacl.61/.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. Metatool benchmark for large language models: Deciding whether to use tools and which to use, 2024b. URL https://arxiv.org/abs/2310.03128.

Vladimir Karpukhin, Barlas Oguz, Sewon Ye, Patrick Lewis, Sebastian Riedel, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, 2020.

C Lee et al. NV-Embed: Improved techniques for training LLMs as generalist embedding models. *arXiv preprint arXiv:2405.17428*, 2024.

Minghao Li, Yifan Zhao, Bowen Yu, Feifan Song, Hang Li, Haiyang Yu, and Zhoujun Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Z Li et al. gte-Qwen2: A generalist text embedding family. *arXiv preprint arXiv:2407.09261*, 2024.

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

Yanming Liu, Xinyue Peng, Jiannan Cao, Yuwei Zhang, Xuhong Zhang, Sheng Cheng, Xun Wang, Jianwei Yin, and Tianyu Du. Tool-planner: Task planning with clusters across multiple tools, 2025. URL https://arxiv.org/abs/2406.03807.

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

Xuan Lu, Haohang Huang, Rui Meng, Yaohui Jin, Wenjun Zeng, and Xiaoyu Shen. Tools are under-documented: Simple document expansion boosts tool retrieval. *arXiv preprint*, 2025.

Yining Mao, Yuxuan Shi, Fengran Zhang, Hongyu Huang, Zihan Liu, Yun Chen, and Yi Mao. GTA: A benchmark for general tool agents in real-world scenarios. *arXiv preprint arXiv:2407.04407*, 2024.

Suhong Moon, Siddharth Jha, Lutfi Eren Erdogan, Sehoon Kim, Woosang Lim, Kurt Keutzer, and Amir Gholami. Efficient and scalable estimation of tool representations in vector space, 2024. URL https://arxiv.org/abs/2409.02141.

N Muennighoff et al. GritLM: Generating text with generative retrieval information language modeling. *arXiv preprint arXiv:2402.09906*, 2024.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. In *ICLR*, 2024.

Shu Qiao, Xin Liu, Chen Lin, Shijian Yu, Wen Yao, Min Peng, Weiguang Lin, Detao Huang, and Junfeng Lu. ToolAlpaca: Generalizing tool learning of small language models. *arXiv preprint arXiv:2312.06244*, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunliang Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, and Bill Qian. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *ICLR*, 2024.

Saptarshi Sengupta, Zhengyu Zhou, Jun Araki, Xingbo Wang, Bingqing Wang, Suhang Wang, and Zhe Feng. ToolDreamer: Instilling LLM reasoning into tool retrievers. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5465–5482, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.254. URL https://aclanthology.org/2026.eacl-long.254/.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. In *NeurIPS*, 2024.

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. In *ACL*, 2025.

T Stefanović. Smaller flashlight, bigger accuracy: The power of bm25s. *arXiv preprint arXiv:2407.03618*, 2024.

L Wang et al. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2023. e5-mistral-7b-instruct model.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. Toolgen: Unified tool retrieval and calling via generation, 2025. URL https://arxiv.org/abs/2410.03439.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive estimation for dense text retrieval. In *ICLR*, 2021.

Zongxu Yang, Shuyu Wang, Miaoxuan Liu, Junlin Zhang, Boyu Dong, Hengyuan Wang, Fei Gao, Bo Zhang, Ying Li, Jun Liu, et al. AppBench: Aligning and delegating LLMs as versatile app actors. *arXiv preprint arXiv:2405.16340*, 2024.

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

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint*, 2025.

## A LLM PROMPT TEMPLATES

**Stage 1: Multi-Tool Query Decomposition.** We use DeepSeek-V3.2 for query decomposition during benchmark construction. The system prompt instructs the LLM to decompose a query into exactly $k$ atomic sub-queries, one per ground-truth tool, minimizing the semantic gap between sub-query text and tool documentation (Listing 1). Because this prompt receives the benchmark-provided tool count and reference tools, it is an oracle-assisted annotation component rather than a deployable test-time planner; the corresponding diagnostic rows in the main paper reuse it only as an oracle upper bound.

**Stage 1 --- Sub-query Decomposition (DeepSeek-V3.2)**

```
You are the Planner in a Plan-and-Execute system.
Given a user query and a set of relevant tools, your job is to
decompose the query into exactly N executable steps (subqueries),
one per relevant tool, so that executing all subqueries in order fully
answers the query.
Each subquery will later be used as a search query to retrieve its
corresponding tool from a large tool library.  Therefore, the primary
goal of each subquery description is to MINIMIZE THE SEMANTIC GAP
between the subquery text and the tool documentation, so that the
correct tool can be accurately retrieved.

Rules:  1.  Produce exactly N subqueries where N equals the number
of relevant tools provided.  2.  Each subquery must map to exactly
one relevant tool via its "relevant_tool_id".  3.  Write subquery
descriptions with enough detail to enable accurate retrieval.  Focus
on two things only:  (a) FUNCTIONALITY --- the precise operation
the tool performs (e.g., retrieve, calculate, search, filter,
authenticate, convert) (b) ENTITY --- the specific data subject or
domain the operation acts on (e.g., stock ticker, historical events on
a date, user credentials, ...)  Use domain-relevant terminology that
likely appears in the tool documentation.  Do NOT describe specific
parameter names, parameter types, or return value types.  4.  Do
NOT mention any tool name, API name, or function name.  Describe the
operation and data, not the implementation.
Return ONLY a valid JSON array --- no markdown fences, no explanation
outside the JSON.
```

**Stage 2: Tool Verification.** We use GPT-4o-mini to verify tool functionality for the sub-query. The system prompt instructs the LLM to judge whether a candidate tool can be directly invoked to complete a given atomic sub-query, using the ground-truth tool as a reference benchmark (Listing 2).

**Stage 2 --- Tool Verification (GPT-4o-mini)**

```
Your task is to judge whether a given tool can be directly invoked to
complete a given sub-query.
The sub-query is atomic --- it describes a single operation
completable by calling one tool.  You are also given a Reference Tool
--- a ground-truth tool that can already complete this sub-query.  If
the tool under evaluation performs the same core operation or has
equivalent functionality, answer "yes".
```

```
Rules: Answer "yes" when ALL of the following hold: 1. The tool's
primary function directly performs the same operation as the Reference
Tool. 2. The tool's output covers what the sub-query needs (same
type of result as the Reference Tool).
Answer "no" when ANY of the following apply: 1. The tool's function
does not match the sub-query's required operation. 2. The tool's
output only partially satisfies the sub-query and significant extra
steps are needed.
When genuinely uncertain, answer "no".
Return ONLY a valid JSON object: {"verdict": "yes" or "no",
"reason": "<one concise sentence>"}
```

**Stage 3: Combination Audit.** We use Claude-Sonnet-5 to audit tool combinations for the specific query and instruction. The system prompt instructs the LLM to determine whether a set of available tools is sufficient to fully complete the user's query, using a sub-query breakdown as a coverage checklist and the ground-truth tool set as reference. In this stage, the LLM audits the provided tool combinations under the query context. (Listing 3).

**Stage 3 --- Combination Audit**

```
You are a tool-use agent. Your task is to determine whether a given
set of available tools is sufficient to fully complete the user's
query.
You will receive the user's query, an instruction providing sub-query
context, and the available tools. You may also receive a subquery
breakdown --- use it as a checklist to verify each required step
is covered. If a reference tool set is provided, treat it as a
known-correct benchmark: tools that perform the same core operations
as the reference tools count as sufficient, even if they differ in
name, interface details, or parameter formats.

Rules: 1. Answer "yes" when the available tools collectively
cover every operation needed to fulfill the query and can produce
the result the user wants. Note that intermediate values (such as
authentication tokens, resource IDs, or session handles) are assumed
to be passed between tools at runtime --- you do not need to verify
that explicitly; focus on whether the required operations exist.
2. Answer "no" when an essential operation is entirely absent from
the available tools --- meaning no tool can perform a required
action at all (a functional mismatch, not a minor format or naming
difference).
Return ONLY a valid JSON object: {"verdict": "yes" or "no",
"reason": "<one concise sentence>"}
```

## B HUMAN VALIDATION GUIDELINES

This appendix specifies the protocol for the human validation used to check automatically added positives from TOOLEX, and reports its outcome.

**Sampling unit.** We audit only the *final equivalent tool-combination annotations*, i.e., accepted query-level combinations after Stage 3. In our implementation, these items are stored in `query_combo_verified.jsonl`. Audit items were sampled with 1,000 combinations per benchmark, stratified by domain (Code/Web/Customized, tool side) and by query complexity (number of required tools / number of sub-queries, both sides).

**Reviewer inputs.** Each reviewer receives the original query, the task instruction, the sub-query breakdown (if available), the candidate tool combination, the reference ground-truth combination, and the model-produced rationale stored with that combination.

**Decision rule.** Mark a candidate combination as **valid** iff both conditions hold: (1) the combination jointly covers all required operations in the original query, and (2) the tools are dependency-

compatible whenever one tool's output must feed another (e.g., shared platform, compatible identifiers, or session state). Otherwise mark it **invalid**.

**Adjudication.** Each item is judged independently by two reviewers. Disagreements are resolved by a third reviewer who sees both rationales but not model scores.

**Audit results and failure modes.** Combination-level precision is 85.9% for TOOLEQ and 95.7% for SKILLEQ. Table 4 breaks the invalid combinations down by failure mode.

Table 4: Failure-mode distribution of invalid combinations in the human validation, reported as a share of all invalid items per benchmark.

| Failure mode | ToolEq | SkillEq |
|---|---|---|
| Cross-platform dependency mismatch | 98.7% | 67.8% |
| Near-synonym, non-substitutable | 1.3% | 32.2% |
| Decomposition ambiguity | 0% | 0% |
| Incomplete functional coverage | 0% | 0% |

For TOOLEQ, failures are almost entirely **cross-platform dependency mismatch**: the pipeline assembles individually valid tools that cannot cooperate at run time. A representative case is an instruction to add a reminder to a memo service, which requires logging in before writing — feasible only when the login API and the reminder API come from the same provider, so a combination mixing providers fails dependency consistency even though each tool covers its sub-query. For SKILLEQ, inter-skill dependencies are much weaker — skills rarely exchange run-time values such as tokens or session handles — so the dependency failure share drops to 67.8% and near-synonym but non-substitutable skills account for the remaining 32.2%, which explains the higher skill-side pass rate. No invalid item in either benchmark was attributed to decomposition ambiguity or incomplete functional coverage.

## C JUSTIFICATION OF BEST-MATCH EVALUATION

This section formalizes why best-match evaluation is a principled way to extend a standard IR metric from one to many ground-truth combinations, and why the gap $\Delta_M$ admits a clean interpretation.

**Setup.** Fix a query $q$, the retrieval list $L_q$ with per-tool scores $s_q(t)$, a cutoff $K$, and the expanded set $\hat{\mathcal{C}}_q = \{C_1, \ldots, C_m\}$ of valid equivalent combinations. Let $M(L_q, C_j) \in [0, 1]$ denote any of $\mathrm{NDCG@}K$, $\mathrm{Recall@}K$, $\mathrm{Comp@}K$ computed against the single reference $C_j$ under the same single-reference definitions used in the main paper. All combinations in $\hat{\mathcal{C}}_q$ are by construction *equally valid* solutions to $q$: the pipeline verifies each to be functionally sufficient, and none is privileged over another. The one-to-one benchmark resolves this equivalence arbitrarily by selecting one $C^* \in \hat{\mathcal{C}}_q$ as the sole reference; TOOLEQ instead asks the label-agnostic question: *among all valid solutions, which one does the retrieval list actually find, and how well does it rank it?*

**Principled multi-reference extension.** Each $M(L_q, C_j)$ is the score the retrieval earns if $C_j$ were the reference. Because every $C_j$ is an equally valid reference, a score that does not depend on which combination the annotator happened to record must select the reference on the basis of the retrieval itself, not the labels. Equation 3 does so through the alignment score $\sum_{t \in C} s_q(t)$: since all combinations share the same cardinality (one tool per sub-query), this selects the valid solution whose tools the retrieval collectively ranks highest — i.e., the valid solution the system actually found — and scores the standard metrics against it. This realizes the semantics "a retrieval is correct if it matches *any* valid equivalent combination," and reduces to the standard single-reference metric when $|\hat{\mathcal{C}}_q| = 1$. Aggregates such as the mean over references would instead discount retrievals that match some but not all equivalents, re-introducing a false-negative penalty; best-match credits each valid match in full.

**Coherence across metrics.** Under best-match, $\mathrm{NDCG@}K$, $\mathrm{Recall@}K$, and $\mathrm{Comp@}K$ are all computed against the same selected reference $C_q^\dagger$, so a reported metric triple describes one coherent matching solution rather than mixing references across metrics.

**The gap and its interpretation.** Stage 3 guarantees $C^* \in \hat{\mathcal{C}}_q$, so the original one-to-one reference always competes in the selection. The reported $\Delta$ is the macro-averaged gap $\Delta_M = \mathbb{E}_q\big[M(L_q, C_q^\dagger) - M(L_q, C^*)\big]$, which is positive for every model and category we evaluate; it measures the credit TOOLEQ restores by recognizing retrieved equivalents that the one-to-one convention counted as misses. We note that, unlike a per-metric maximum over references, best-match does not *pointwise* dominate $M(L_q, C^*)$: the best-aligned reference can occasionally rank slightly below $C^*$ on an individual metric. The dominance we rely on is aggregate-level, which is the level at which all reported comparisons are made.