ToolSense: A Diagnostic Framework for Auditing Parametric Tool Knowledge in LLMs

Anonymous ACL submission

Abstract

Large language models deployed as agents over large tool catalogs face a critical tool-retrieval bottleneck. As embedding-based retrieval approaches rely on compact encoders that may under-capture specialized tool semantics, parametric tool retrieval (Wang et al., 2025) addresses this by encoding each tool as a *virtual token* appended to the LLM vocabulary, fine-tuned in two stages (memorization then retrieval SFT) to use the LLM as a retriever, achieving strong performance on standard ToolBench retrieval benchmarks. Yet these benchmarks use verbose, fully-specified queries, and their evaluation applies constrained decoding that restricts outputs to valid token paths — neither reveals whether the model actually *understands* its tools. We introduce **ToolSense**, an open-source LLM-powered diagnostic framework that takes any tool catalog as input and automatically generates three benchmarks: a Realistic Retrieval Benchmark (RRB) with queries at three ambiguity tiers, an MCQ probing benchmark, and a QA probing benchmark. Applying ToolSense to ToolBench (∼47k tools) and evaluating five parametric model training configurations reveals a *knowledge-retrieval dissociation*: on RRB queries, several configurations collapse by **∼50-64 percentage points** compared to fully-specified ToolBench benchmarks, falling below embedding-model baseline. Additionally, despite strong retrieval performance, some models score near-random on factual probes, suggesting a knowledge-retrieval dissociation. We open-source the ToolSense framework and the ToolBench diagnostic benchmarks to encourage further research. (see supplementary material)

1 Introduction

Large language models are increasingly deployed as autonomous agents over large software ecosystems, where selecting the right tool from catalogs of thousands of APIs is a core bottleneck (Schick et al., 2023; Yao et al., 2023; Qin et al., 2024a). The dominant approach encodes tool descriptions as dense vectors and retrieves top-k candidates via approximate nearest-neighbor search (Karpukhin et al., 2020). This paradigm has known limitations: small retrieval encoders may underspecify the semantics of complex, overlapping APIs at scale (Wang et al., 2025); retrieved descriptions must still be injected into the LLM context, compounding overhead across multi-step agent loops; and retriever and generator are trained with separate objectives, preventing joint optimization toward task success (Tay et al., 2022).

Parametric Tool Retrieval. Wang et al. (2025) propose ToolGen, an alternative strategy to encode the entire tool catalog directly into LLM parameters. Each tool is assigned a unique *virtual token* appended to the model vocabulary as a flat atomic identifier representing the full tool identity (e.g., «`TIKTOK&&GET_TRENDING_VIDEOS`»). In Stage 1 (memorization), the model is fine-tuned to associate each tool's metadata with its virtual token. In Stage 2 (retrieval), the model is further fine-tuned on (query → virtual token) pairs, learning to generate the correct token given a natural language query. At inference, a *DisjunctiveTrie* (Cao et al., 2021) constrains beam search to valid token sequences, guaranteeing every output maps to a real tool. Applied to ToolBench (Qin et al., 2024b) (∼47k tools), this paradigm achieves over ≈ 0.90 recall on standard ToolBench benchmarks.

The Diagnostic Gap. This naturally raises a question: *does parametric training produce a model that understands its tools, or does it simply learn to match query patterns to token identifiers?* A model that has learned only surface-level mappings may appear to work on distribution-matched benchmarks while failing the moment a user phrases their query differently. Yet existing evaluation is not designed to reveal this, for two reasons. First, the queries in ToolBench's standard

evaluation splits (G1/G2/G3) are verbose and fully specified, unlike how a real user would phrase them (examples in Appendix A). Second, constrained decoding means the model need only rank paths through a fixed trie rather than freely recall a tool token, masking whether genuine memorization occurred. This matters practically: downstream agentic fine-tuning typically requires free-form token generation without trie support, making undetected memorization gaps a silent failure risk. Together, verbose queries and constrained decoding make it difficult to distinguish pattern matching from genuine tool knowledge (Petroni et al., 2019), motivating a more diagnostic evaluation protocol.

We introduce **ToolSense**, an open-source LLM-powered diagnostic framework that takes any tool catalog as input and automatically generates three diagnostic benchmarks: (1) a *Realistic Retrieval Benchmark* (RRB) with queries in three ambiguity tiers reflecting how human users actually write queries; (2) an *MCQ probing benchmark* testing discriminative factual knowledge about tool capabilities; and (3) a *QA probing benchmark* testing inferential factual knowledge about tool properties. Alongside the benchmarks, we establish a free-form evaluation protocol that reports an *Internalization Score* (IS@k = free@k / constrained@k) as a trie-dependency diagnostic.

Applying ToolSense to ToolBench reveals a striking gap: configurations that appear highly capable on standard benchmarks collapse dramatically on realistic queries, falling below non-parametric baselines. Factual probing further reveals that Stage 2 training (responsible for the retrieval gains) nearly universally destroys the tool knowledge acquired in Stage 1.

To summarize, our contributions include: **(i) ToolSense**, an open-source diagnostic framework that auto-generates RRB, MCQ, and QA benchmarks from any tool catalog; **(ii) ToolBench diagnostic benchmarks** RRB, MCQ, and QA released openly; and **(iii)** an empirical diagnosis of five parametric configurations on ToolBench, revealing the training choices that govern the knowledge-retrieval dissociation.

2 Background and Related Work

Generative and parametric retrieval. Tay et al. (2022) introduced Differentiable Search Index (DSI), encoding document IDs into transformer parameters and retrieving via constrained beam search. Cao et al. (2021) applied autoregressive entity retrieval to knowledge-grounded generation. Wang et al. (2022) extended this to dense document corpora. Wang et al. (2025) adapt the paradigm to tool catalogs with virtual tokens and two-stage training. All of these systems evaluate exclusively with constrained decoding; our free-form IS protocol introduces a new diagnostic practice for this entire class of systems.

Tool learning in LLMs. Qin et al. (2024b) build ToolBench, a benchmark covering ∼16k real-world APIs (∼47k tools). Patil et al. (2024) fine-tune LLMs to generate tool-calls with retrieval augmentation. Schick et al. (2023) train LLMs to self-insert tool-calls in context. We focus specifically on the *retrieval* stage of parametric tool systems and ask whether that retrieval mechanism encodes tool semantics, a question orthogonal to downstream task performance studied by these works. A concurrent line of work (Anonymous, 2026) improves dense encoder retrieval for tool search via iterative co-training of a contrastive encoder and an LLM rewriter, an approach orthogonal to ToolSense's diagnostic focus on parametric retrieval.

Probing and interpretability. Petroni et al. (2019) introduce LAMA, probing pre-trained LMs for factual knowledge via cloze-style queries. Tenney et al. (2019) probe BERT's representations for linguistic structure. Our MCQ and QA probes extend this tradition to a novel setting: tokens *learned during fine-tuning* (virtual tokens), not pre-trained representations. We probe whether fine-tuning creates genuinely semantic representations or only task-relevant pointers.

Knowledge retention in fine-tuning. McCloskey and Cohen (1989) identify catastrophic interference as a fundamental failure mode when neural networks are trained sequentially on new tasks. Hu et al. (2022) propose LoRA to mitigate this by freezing backbone weights and learning low-rank updates, preserving pre-trained representations. Biderman et al. (2024) empirically show that LoRA retains more prior task performance than full fine-tuning. Our approach extends this line of work to the parametric retrieval setting.

3 The ToolSense Framework

The framework takes as input a **tool catalog**

$$\mathcal{C} = \{\, \tau_i = (\text{name}_i,\ \text{desc}_i) \,\}_{i=1}^{|\mathcal{C}|} \qquad (1)$$

and automatically generates three diagnostic benchmark datasets. Figure 1 illustrates the pipeline. In the following subsections, we explain the benchmark generation methodology in detail.

3.1 Realistic Retrieval Benchmark (RRB)

RRB tests whether a retrieval system generalises beyond verbose-style queries by presenting it with short, intent-focused requests at three ambiguity levels. Generation proceeds in four stages: seed sampling, hard-negative pool construction, parallel generation tiers, and dual validation.

Stratified seed sampling. Rather than generating queries for every tool in $\mathcal{C}$, we sample N_{RRB} anchor tools stratified by domain to ensure broad catalog coverage, yielding $\mathcal{C}_{\text{RRB}} \subset \mathcal{C}$ with $|\mathcal{C}_{\text{RRB}}| = N_{\text{RRB}}$.

Hard-negative pool. For each anchor $\tau \in \mathcal{C}_{\text{RRB}}$, let $\phi : \mathcal{C} \to \mathbb{R}^d$ be a sentence encoder. We retrieve the K nearest neighbours by cosine similarity, $\mathcal{H}_K(\tau) = \arg\text{top-}K_{\tau' \in \mathcal{C} \setminus \{\tau\}} \langle \phi(\tau), \phi(\tau') \rangle$, and form the candidate pool $\mathcal{P}(\tau) = \{\tau\} \cup \mathcal{H}_K(\tau)$ with $|\mathcal{P}(\tau)| = K+1$. The pool defines a hard-negative context shown to the generator and the labels from which ground-truth answers A are drawn.

Parallel generation tiers. For each anchor, three sub-pipelines run in parallel, $t \in \{\text{easy}, \text{medium}, \text{hard}\}$, each generating a batch of (q, A) pairs from the same candidate pool $\mathcal{P}(\tau)$:

- **Easy**: $|A|$=1; a concise query pointing to exactly one tool.
- **Medium**: $|A| \in \{2, 3\}$; a cross-functional request genuinely satisfiable by 2–3 tools.
- **Hard**: $|A| \geq 4$; a high-level, ambiguous goal spanning 4 or more tools.

The generator LLM θ_{RRB} receives $(\tau,\ \mathcal{P}(\tau),\ t,\ \mathcal{E})$, where $\mathcal{E} = \{\hat{q}_i^{(e)}\}_{i=1}^{n_e}$ are few-shot query examples provided to guide the style and tone of intent-focused query generation.

Dual validation with feedback. Each generated batch passes two sequential filters. The programmatic filter $\mathcal{V}_{RRB}(q, A, \tau, \mathcal{P}(\tau))$ rejects entries failing either of two checks: answers must be grounded in the pool ($A \subseteq \mathcal{P}(\tau)$), and no answer tool name may appear verbatim in the query ($\forall \tau' \in A : \text{name}(\tau') \notin q$). Entries passing $\mathcal{V}_{RRB}$ are then scored by an LLM judge $\mathcal{J}_{RRB}$ for naturalness, tier-compliance, and label correctness. Rejected entries are dropped, yielding the validated dataset $\mathcal{D}_{\text{RRB}} = \{(q_j,\ A_j,\ t_j)\}_{j=1}^{N_{\text{RRB}}}$.

3.2 Probing Benchmarks (MCQ and QA)

Both probing benchmarks follow the same pipeline structure: sample a seed set of tools, generate question–answer pairs via an LLM generator, validate with an LLM judge $\mathcal{J}$, and filter entries for which no unambiguous question can be formed. In all accepted entries, the question refers to the tool only as "this tool", never by name, so the model must answer given the virtual token v_τ.

MCQ (N_{MCQ} tools). For each τ, θ_{MCQ} produces a factual question z about the tool's functionality, one correct answer $o_{y^\star}$, and three plausible-but-wrong distractors, yielding $\mathcal{D}_{\text{MCQ}} = \{(v_{\tau_i},\ z_i,\ \{o_k^i\},\ y_i^\star)\}$. At evaluation the model receives $(v_\tau,\ z,\ \{o_k\}_{k=1}^4)$; constrained decoding restricts output to option letters (A/B/C/D) and the prediction $\hat{y}$ is compared with $y^\star \in \{0, 1, 2, 3\}$.

QA (N_{QA} tools). For each τ, θ_{QA} receives the description alongside a pre-specified target $b \in \{\text{Yes}, \text{No}\}$, alternated across tools to ensure label balance, and produces a binary question w about a specific, verifiable tool property (e.g., supported modality, domain, etc.), yielding $\mathcal{D}_{\text{QA}} = \{(v_{\tau_i},\ w_i,\ b_i)\}$. At evaluation the model receives $(v_\tau,\ w)$; constrained decoding restricts output to Yes/No and $\hat{b}$ is compared with b.

3.3 Free-form Evaluation and Internalization Score

Let $\hat{A}_c(q, k)$ and $\hat{A}_f(q, k)$ denote the top-k outputs under *constrained* (trie-guided) and *free-form* (unconstrained) decoding respectively using beam search with beam width B. Constrained and free-form Recall@k over query set $\mathcal{Q}$ are defined as $R_c@k = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbb{1}[A(q) \cap \hat{A}_c(q, k) \neq \emptyset]$ and $R_f@k = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbb{1}[A(q) \cap \hat{A}_f(q, k) \neq \emptyset]$. The **Internalization Score** (IS) is then:

$$\text{IS@k} = \begin{cases} R_f@k \ / \ R_c@k & \text{if } R_c@k > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

IS ≈ 1 indicates the model generates correct tokens equally well without the trie (retrieval is internalized); IS ≈ 0 indicates complete trie-dependence. IS is a ratio of two standard measurements, which enables us to *always* report $R_f@k$ alongside $R_c@k$, making trie-dependence transparent. (see Appendix F for design rationale for IS.)

4 Experimental Setup

We instantiate ToolSense on the ToolBench (Qin et al., 2024b) tool catalog to empirically diagnose

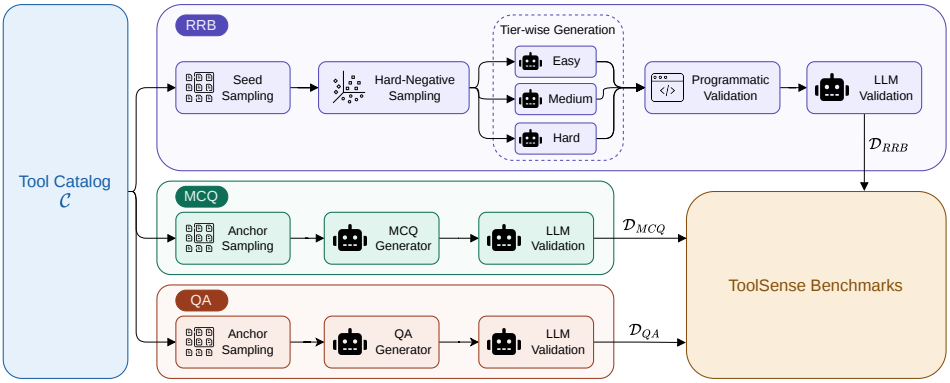


Figure 1: The ToolSense diagnostic framework generates $\mathcal{D}_{\text{RRB}}$, $\mathcal{D}_{\text{MCQ}}$, and $\mathcal{D}_{\text{QA}}$ from a tool catalog $\mathcal{C}$.

parametric tool retrieval across three dimensions: OOD generalization beyond training-distribution queries, dependence on constrained decoding, and correspondence between retrieval performance and semantic tool knowledge. All trained models follow the two-stage ToolGen paradigm (Wang et al., 2025): Stage 1 (memorization) fine-tunes on (tool metadata → virtual token) pairs; Stage 2 (retrieval) fine-tunes on (query → virtual token) pairs.

Training and evaluation data. Stage 1 uses 46,980 RapidAPI tools provided by ToolBench, each with a api name & description as well as endpoint name & description. Stage 2 uses ∼195k verbose (query, tool) pairs from the ToolBench training split (Qin et al., 2024b), generated by GPT-4 (OpenAI et al., 2024). For standard evaluation, we retain the ToolGen evaluation splits: G1 (593 queries), G2 (399), and G3 (100) (Wang et al., 2025); these match the verbose training distribution and serve as the in-distribution baseline (see Appendix A for a query style comparison).

Diagnostic benchmarks. Applying the ToolSense generation pipelines (§3.1–§3.2) to this catalog yields: an RRB of 500 queries across three difficulty tiers (167 easy / 167 medium / 166 hard), an MCQ probe of 496 samples testing discriminative factual knowledge, and a QA probe of 500 items testing inferential knowledge. Together, these evaluate retrieval generalization (RRB) and semantic understanding (MCQ/QA) as orthogonal dimensions of tool knowledge.

4.1 Model Configurations

We primarily use open-source instruction-tuned **Gemma3-4B** (Team et al., 2025), with **Qwen3.5-4B** (Yang et al., 2025) for cross-architecture validation and **Gemma3-12B** for scale ablations. For each configuration we train both a full fine-tuned (FFT) and a LoRA (Hu et al., 2022) variant; training details are in Appendix D. In all LoRA variants, adapters are attached to linear layers only, while the full embedding layer remains trainable.

We define five primary training configurations in Table 1, each isolating one variable along three axes: *token format* (flat single-token vs. hierarchical multi-token), *memorization format* (single-format vs. multi-format with reverse mappings and hard negatives), and *training method* (FFT vs. LoRA). Flat tokens follow the original ToolGen format (§1); hierarchical tokens decompose the identifier into a two-step sequence «`API`»«`Endpoint`» (e.g., «`TIKTOK`»«`GET_TRENDING_VIDEOS`») that the model should generate autoregressively.

TG replicates the original ToolGen setup (flat tokens, desc→tok only, no system prompt) and serves as the baseline. **TG-SP** adds a task-specific system prompt applied consistently across both training stages and inference. **TG-3FM** extends TG-SP with two additional Stage 1 formats (tok→desc and Multi-Choice Tool Selection (MCTS, Appendix B)), testing whether richer memorization objectives produce more robust representations. **TG-H** switches from flat to hierarchical tokens while keeping all else matched to TG-SP, isolating the effect of token structure. **TG-5FM** combines hierarchical tokens with all five memorization formats (Table 1), where the additional formats train each hierarchical generation step independently. Where applicable, we additionally train a **LoRA** variant for the above configurations. Full training details for each configuration are in Appendix C.

Stage 2 training uses identical (query → tok) data across all configurations; the only variation is

Config	Tokens	Tool Memo. Formats	SP	Train
TG	Flat	1 (desc→tok)	✗	FFT, LoRA
TG-SP	Flat	1 (desc→tok)	✓	FFT, LoRA
TG-3FM	Flat	3 (desc→tok, tok→desc, MCTS)	✓	FFT, LoRA
TG-H	Hier.	1 (desc→tok)	✓	FFT
TG-5FM	Hier.	5 (all*)	✓	FFT, LoRA

*desc→tok, desc→api_tok, desc+api_tok→endpoint_tok, tok→desc, MCTS

Table 1: Diagnostic model configurations. `tok` denotes the tool token encoding the full tool identity; `api_tok` and `endpoint_tok` are the two hierarchical tokens corresponding to API and endpoint respectively. SP = system prompt.

whether a system prompt is prepended, consistent with how each configuration was trained in Stage 1.

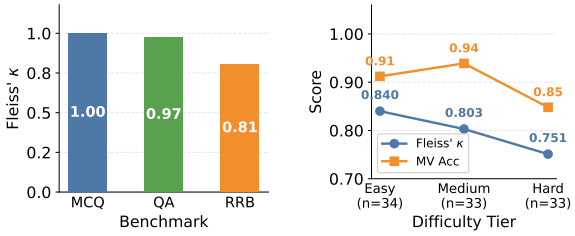
Retrieval baselines. We compare against **BM25** (Robertson and Zaragoza, 2009) (sparse lexical) and **text-embedding-3-large** (OpenAI, 2024) (te3l; OpenAI dense embeddings) on all evaluation splits.

Evaluation metrics. $R_c@k$, $R_f@k$, and IS$@k$ follow the formal definitions in §3.3 and are reported at beam width B=50 unless otherwise specified. MCQ and QA probing benchmarks are evaluated by exact-match accuracy using constrained decoding: MCQ restricts generation to option letters (A/B/C/D) and QA restricts to Yes/No; random baselines are 25% and 50%, respectively.

Benchmark quality validation. Three human annotators independently validated a stratified 100 samples from each benchmark: for RRB, annotators labeled from 14 candidate tools (ground truths + hard negatives) that matched each query's intent; for MCQ/QA, annotators answered each item given the full tool description. Figure 2a shows Fleiss' κ across all three benchmarks: MCQ achieves perfect agreement ($\kappa = 1.000$), QA near-perfect ($\kappa = 0.973$), and RRB substantial agreement ($\kappa = 0.805$). Figure 2b breaks down RRB agreement by difficulty tier: κ decreases gracefully from easy (0.840) to hard (0.751) while remaining above agreement threshold (0.70), confirming that the tier structure reflects genuine task difficulty rather than benchmark artifacts.

5 Results

We present findings across retrieval generalization, constrained decoding dependence, and semantic tool knowledge.



(a) Inter-annotator agreement Fleiss' κ per benchmark.

(b) RRB breakdown by difficulty tier.

Figure 2: Human annotation study ($N = 100$ items $\times$ 3 annotators). RRB agreement decreases gracefully with difficulty while majority-vote accuracy remains high (85–94%).

5.1 Generalization Collapse on Realistic Queries

Table 2 reports $R_c@50$ through both training stages and across evaluation splits. Before any retrieval SFT, Stage 1 memorization alone already reaches 38–47% on G1 and 27–41% on RRB for flat configurations, denoting how well the model has absorbed tool semantics during memorization. Stage 2 retrieval training then lifts G1 sharply into a 90–96% band, consistent with (Wang et al., 2025), yet performance on RRB advances only modestly: most configurations gain 2-18pp over their Stage 1 baselines, and TG-5FM (LoRA) actually *regresses* (41.0% → 31.9%). The absolute G1→RRB drop at Stage 2 is 50–52pp for flat and 61–64pp for hierarchical configurations, which suggests that parametric tool retrieval trained on synthetic verbose queries does not generalize to queries that differ in phrasing from the training distribution.

The ranking reverses on RRB: the best parametric model (TG-3FM, 44.4%) falls below retrieval baseline (te3l, 55.6%), and BM25 (32.4%) remains competitive with several parametric configurations. Both retrieval baselines actually *improve* from G1 to RRB (BM25: 27.8%→32.4%; te3l: 47.0%→55.6%), suggesting that the retrieval methods work better for natural-language queries.

The intermediate G2 and G3 splits trace the OOD gradient, both fall between G1 and RRB at Stage 2, confirming that the collapse is gradual rather than a sharp boundary. We examine the structural and training-dynamics reasons behind these findings in §5.2 and §5.4.

5.2 Trie-Dependency of Hierarchical Tokens

The Internalization Score quantifies whether a model generates valid tool tokens independently or

Config	G1 (%)		G2 (%)		G3 (%)		RRB (%)		Δpp
	S1	S2	S1	S2	S1	S2	S1	S2	
TG	47.0 [43.6,50.4]	**95.7 [94.2,97.0]**	37.0 [33.8,40.4]	84.5 [81.5,87.3]	**35.0 [27.9,42.7]**	84.6 [80.4,88.5]	37.8 [34.3,41.2]	43.8 [40.3,47.7]	−52
TG (LoRA)	38.8 [35.2,42.4]	90.4 [88.7,92.2]	28.4 [25.3,31.6]	82.8 [79.8,85.5]	26.8 [20.0,34.0]	84.9 [79.9,89.6]	31.8 [28.2,35.2]	37.0 [33.4,40.7]	−53
TG-SP	38.6 [35.0,42.4]	94.5 [92.9,96.0]	28.5 [25.4,31.5]	84.8 [81.5,87.6]	27.0 [20.7,33.7]	86.6 [81.9,90.8]	28.5 [25.2,32.0]	43.2 [39.6,46.7]	−51
TG-3FM	43.1 [39.3,46.4]	95.3 [93.9,96.6]	34.6 [31.2,38.3]	**85.3 [82.4,88.0]**	31.5 [24.5,38.9]	86.2 [82.3,90.3]	26.7 [23.4,30.1]	44.4 [40.7,48.3]	−51
TG-3FM (LoRA)	42.5 [38.8,46.1]	93.0 [91.4,94.6]	**37.4 [33.9,40.9]**	83.5 [80.6,86.4]	21.9 [15.7,28.4]	88.4 [84.3,92.0]	29.8 [26.5,33.3]	43.2 [39.7,46.6]	−50
TG-H	23.8 [20.5,27.1]	90.8 [88.9,92.6]	12.2 [9.9,14.4]	78.9 [75.5,82.0]	7.9 [4.5,11.8]	87.0 [82.9,90.8]	18.8 [15.9,21.9]	27.1 [24.2,30.5]	−64
TG-5FM	37.8 [34.2,41.6]	92.3 [90.6,93.9]	23.1 [20.3,25.9]	79.0 [76.0,82.1]	15.8 [10.8,21.1]	**89.2 [85.3,92.6]**	28.1 [24.5,31.5]	30.9 [27.9,34.2]	−61
TG-5FM (LoRA)	**48.5 [44.8,52.6]**	92.5 [90.7,94.0]	29.4 [26.0,32.5]	80.3 [77.4,83.4]	27.2 [20.4,34.0]	86.7 [81.4,91.2]	**41.0 [37.0,44.8]**	31.9 [28.4,35.5]	−61
BM25	—	27.8 [24.2,31.4]	—	23.6 [19.4, 27.8]	—	23.0 [14.8, 31.2]	—	32.4 [28.3, 36.5]	—
te3l	—	47.0 [43.0, 51.0]	—	40.6 [35.8, 45.4]	—	38.0 [28.5, 47.5]	—	**55.6 [51.2, 60.0]**	—

Table 2: R_c@50 (in %) for Gemma3-4B across all evaluation splits. **S1** = after Stage 1 memorization (no retrieval SFT); **S2** = after Stage 2 retrieval SFT (full pipeline). Δpp = G1→RRB drop at S2. BM25/te3l are retrieval baselines (Hard S2 only). **Bold** results are best; underlined are second best. Values in brackets are 95% CIs.

Config	G1 IS	RRB IS
TG Gemma3-4B	0.75 [0.72,0.79]	0.85 [0.74,0.96]
TG-SP Gemma3-4B	0.70 [0.66,0.74]	0.75 [0.65,0.85]
TG-3FM Gemma3-4B	0.69 [0.65,0.73]	0.82 [0.71,0.93]
TG-3FM (LoRA) Gemma3-4B	0.89 [0.86,0.93]	0.81 [0.71,0.91]
TG-H Gemma3-4B	0.42 [0.38,0.46]	0.33 [0.24,0.41]
TG-5FM Gemma3-4B	0.84 [0.80,0.87]	0.64 [0.53,0.76]
TG-5FM (LoRA) Gemma3-4B	0.66 [0.62,0.70]	0.79 [0.65,0.92]
TG-SP Qwen3.5-4B	0.90 [0.86,0.93]	0.98 [0.89,1.00]
TG-SP (LoRA) Qwen3.5-4B	**0.94 [0.92,0.97]**	**0.99 [0.89,1.00]**
TG-3FM Qwen3.5-4B	0.79 [0.75,0.83]	0.81 [0.73,0.89]
TG-3FM (LoRA) Qwen3.5-4B	0.68 [0.64,0.72]	0.82 [0.73,0.92]
TG-H Qwen3.5-4B	0.57 [0.53,0.61]	0.28 [0.21,0.35]
TG-SP Gemma3-12B	0.66 [0.62,0.70]	0.84 [0.75,0.93]
TG-3FM (LoRA) Gemma3-12B	0.76 [0.72,0.80]	0.86 [0.76,0.96]

Table 3: Internalization Score (IS@50) after Stage 2 training. 95% CI in brackets.

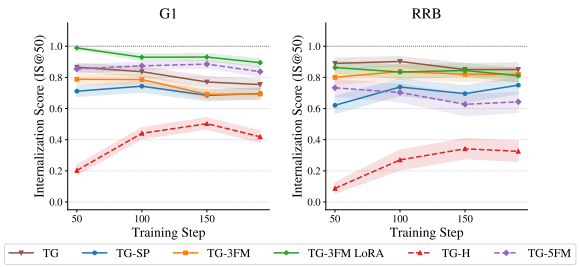


Figure 3: IS@50 across Stage 2 training steps for Gemma3-4B. Shaded bands are 95% bootstrap CIs. Full results across all splits & models are in Appendix G.

relies on trie guidance for heavy lifting (IS≈1.0: independent; IS≈0: trie-dependent).

Table 3 reports IS after Stage 2 ($B = 50$). The separation between token formats is large: flat configurations achieve RRB IS in the range 0.75–0.85, while hierarchical configurations fall to 0.33–0.79, a gap of Cohen's $d = 1.45$ (Cohen, 1988), indicating the two groups occupy fundamentally different regions of trie-dependency. The gap is *larger* on RRB than on G1 for hierarchical tokens (TG-H: G1=0.42, RRB=0.33), while flat tokens maintain or improve IS on RRB (TG: G1=0.75, RRB=0.85), suggesting that the distributional shift compounds with structural trie-dependency. The flat-over-hierarchical IS gap holds across architectures: Qwen3.5-4B flat configurations reach IS≥0.98 on RRB, while TG-H Qwen3.5-4B drops to 0.28; full IS results across all architectures are in Appendix G.

Figure 3 traces IS evolution: TG-H stays below IS=0.35 throughout Stage2 training, while TG-3FM(LoRA) declines only modestly (1.0→0.89), confirming LoRA preserves free-generation capacity from Stage 1.

5.3 Knowledge-Retrieval Dissociation

Table 4 reports MCQ and QA probing accuracy corresponding to Gemma3-4B model after Stage 1 and Stage 2 trainings. The headline finding is a striking dissociation: a model (TG) achieving ∼95% R_c@50 on G1 scores only 31.4% on 4-way MCQ (random=25%) and exactly 50.0% on binary QA (random=50%), suggesting the behavior to be more like a lookup table, not a knowledge base. At Stage 2, full-finetuned models score 20–31% on MCQ and 34–50% on QA; LoRA extends the upper end to 41.7% MCQ and 56.4% QA.

The picture differs sharply by token format. Flat configurations accumulate genuine knowledge during Stage 1 (TG enters retrieval training at 55.4% MCQ and TG-3FM at 32.9%), which Stage 2 then systematically erodes (TG: −24.0pp; TG-3FM: −3.1pp). Hierarchical tokens tell a different story: TG-H and TG-5FM score *below* random chance at Stage 1 itself (23.6% [19.8, 27.2] and 22.8% [18.9, 26.6]), before any retrieval training has occurred, suggesting that these configurations never established semantic grounding in the first place. LoRA substantially buffers Stage 2 destruction for configurations where knowledge does take hold: TG-

Config	MCQ (S1→S2)		QA (S1→S2)	
	S1	S2	S1	S2
TG	**55.4** [50.8,59.7]	31.4 [27.6,35.5]	57.8 [53.6,62.2]	50.0 [45.8,54.0]
TG-3FM	32.9 [28.6,36.9]	29.8 [25.6,33.9]	60.2 [55.8,64.6]	50.2 [45.8,54.8]
TG-3FM (LoRA)	44.8 [40.5,49.0]	**41.7** [37.7,46.2]	**79.8** [76.0,83.2]	**56.4** [52.2,60.6]
TG-H	23.6 [19.8,27.2]	20.4 [16.9,23.8]	50.6 [46.2,55.2]	34.4 [30.0,39.0]
TG-5FM	22.8 [18.9,26.6]	21.6 [17.7,25.2]	50.2 [46.0,54.8]	48.4 [44.2,52.6]
TG-5FM (LoRA)	30.4 [26.4,34.5]	28.8 [25.0,32.9]	56.2 [52.0,60.6]	52.2 [47.8,56.4]

Table 4: MCQ and QA probing accuracy (Gemma3-4B). Random: MCQ=25%, QA=50%. 95% CI in brackets.

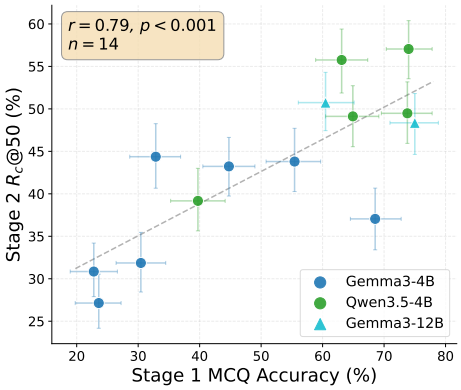


Figure 4: Correlation between Stage 1 MCQ accuracy and Stage 2 RRB R_c@50 across all model families ($n = 14$, $r = 0.79$, $p < 0.001$). Error bars show 95% CIs. All 14 model variants are listed in Table 10 (Appendix H).

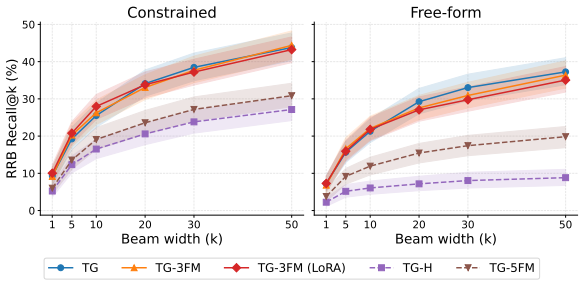


Figure 5: RRB R_c@k and R_f@k vs. beam width k for five Gemma3-4B Stage 2 configurations. Shaded bands are 95% CIs.

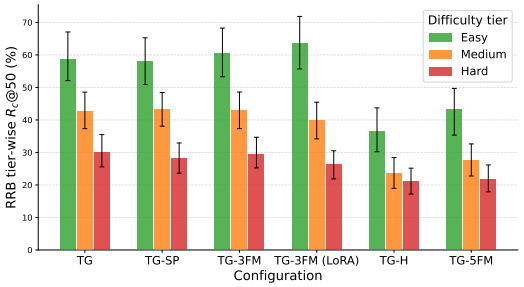


Figure 6: R_c@50 by RRB difficulty tier for five Gemma3-4B Stage 2 configurations. Error bars are 95% bootstrap CIs.

3FM (LoRA) retains 41.7% MCQ (CI [37.7, 46.2]) vs. FFT at 29.8% (CI [25.6, 33.9]) and achieves the highest QA score at Stage 2 (56.4% [52.2, 60.6]).

Across all configurations and model families, Stage 1 MCQ accuracy strongly correlates to Stage 2 RRB R_c@50 (Pearson $r = 0.79$, $p < 0.001$, $n = 14$; Figure 4): configurations that enter retrieval training with richer semantic representations generalize better to OOD queries, even after partial knowledge destruction. (cross-architecture results in Appendix H)

5.4 Design Choice Ablations

We isolate which design choices mitigate or exacerbate the three findings above.

Token format. Flat tokens (TG-SP, Gemma3-4B) achieve RRB R_c@50=43.2% (95% CI [39.6, 46.7]) and IS=0.75, vs. hierarchical (TG-H, Gemma3-4B) at RRB R_c@50=27.1% (CI [24.2, 30.5]) and IS=0.33 (Cohen's $d = 8.23$ on recall, $d = 1.45$ on IS (Cohen, 1988)). This replicates across architectures: Qwen3.5-4B flat R_c@50=55.8% vs. hier R_c@50=39.2% (+16.6pp). Flat tokens require a single decoding step; hierarchical tokens require multi-step composition where errors at the API level propagate to the endpoint level.

Figure 5 shows the gap persists across all beam widths: TG-H plateaus below 9% R_f@k regardless of beam budget, while flat configs scale to 37%.

Figure 6 shows flat configurations declining progressively across RRB difficulty tiers (Easy 58–64% → RRB 26–30%), while hierarchical configurations underperform at every level (Easy: 37–43%, RRB: 21–22%). Token format is the dominant separator, flat leads by 15–25 pp on Easy and Medium, with TG-3FM (LoRA) achieving the highest single-target retrievability (Easy: 63.5%).

LoRA as knowledge preservation. For Gemma3-4B TG-3FM, LoRA preserves MCQ at 41.7% (95% CI [37.7, 46.2]) vs. FFT at 29.8% (CI [25.6, 33.9]). For TG-5FM, the trend is 28.8% (CI [25.0, 32.9]) vs. 21.6% (CI [17.7, 25.2]). For hierarchical tokens, LoRA also improves IS: TG-5FM (LoRA) achieves RRB IS=0.79 vs. FFT at 0.64. The effect is most pronounced where FFT causes severe destruction; for Qwen3.5-4B, which already preserves knowledge without LoRA (TG-3FM: 72.8%), adding LoRA yields only marginal additional gain (74.2%).

Virtual tool token embedding drift. We measure relative L2 drift of virtual tool tokens vs. base-vocabulary tokens across Stage 1→2: vir-

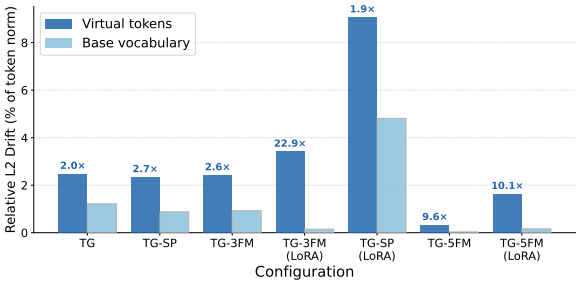


Figure 7: Relative L2 drift (d_{rel}) of virtual vs. randomly sampled vocabulary tokens across Stage 1→Stage 2 for Gemma3-4B configurations.

Model	Config	RRB Recall	IS	MCQ
Gemma3-4B	TG-SP	43.2	0.75	31.4
	TG-3FM	44.4	0.82	29.8
	TG-3FM (LoRA)	43.2	0.81	41.7
	TG-H	27.1	0.33	20.4
Qwen3.5-4B	TG-SP	55.8	**0.98**	61.7
	TG-3FM	**57.0**	0.81	72.8
	TG-3FM (LoRA)	49.5	0.82	74.2
	TG-H	39.2	0.28	31.6
Gemma3-12B	TG-SP	50.7	0.84	66.3
	TG-3FM (LoRA)	48.4	0.86	**76.4**

Table 5: Cross-architecture results (Stage 2, beam=50).

tual tool tokens drift 2–3× farther under FFT and 22.9× farther under LoRA with embedding layer fully-finetuned (Mann-Whitney $p < 2\times10^{-308}$), yet within-cluster cosine similarity barely changes ($|\Delta\text{cosim}| < 0.002$), suggesting tokens shift as coherent clusters rather than diverging to encode distinct knowledge content. Hierarchical tokens enter Stage 2 near-orthogonal to each other ($\text{cosim}_{S1}\approx0.038$ vs. 0.37 for flat tokens) and require 7–24× less repositioning, yet still fail MCQ probes just as severely — suggesting that token geometry is not the source of the knowledge-retrieval dissociation (full analysis in Appendix I)

Multi-format training. Multi-format memorization TG-3FM combined with LoRA achieves the highest Stage 2 MCQ accuracy across all architectures: 41.7% (Gemma3-4B), 74.2% (Qwen3.5-4B), 76.4% (Gemma3-12B). Seeing the same tool tokens in multiple different contexts during Stage 1 appears to build relatively more robust representations that are harder to overwrite during Stage 2 retrieval fine-tuning.

System prompt. During Stage 2, TG (no SP) achieves RRB R_c@50=43.8% (95% CI [40.3, 47.7]) vs. TG-SP at 43.2% (CI [39.6, 46.7]), fully overlapping intervals, suggesting that the system prompt has no measurable effect once retrieval training is complete.

Training data distribution. To test whether the knowledge-retrieval dissociation is a consequence of Stage 2 query distribution rather than the training objective itself, we trained a TG variant whose Stage 2 data consists entirely of RRB-style queries generated by ToolSense RRB generator over the same ∼47k tool catalog. Retraining TG Stage 2 on RRB-style queries (disjoint from evaluation set; see Appendix E) shows that RRB R_c@50 rises from 28.7% to 64.1% (+35.4pp), but G1 R_c@50 drops from 95.7% to 86.6%, suggesting that the model over-specializes to query style that appears in Stage 2 training. MCQ accuracy drops from 31.4% to 26.0%, almost near-random for a 4-choice task. The knowledge-retrieval dissociation thus persists regardless of Stage 2 training data distribution, pointing to the SFT objective itself as the deeper root cause.

5.5 Cross-Architecture Validation

Table 5 confirms both findings generalize: the RRB-recall collapse persists across all model families, and flat tokens consistently outperform hierarchical by 16–18 pp (IS: 0.75–0.98 flat vs. 0.28–0.33 hierarchical). Knowledge retention is not explained by model capacity alone: Qwen3.5-4B retains 62–73% MCQ despite matching Gemma3-4B in size (29–31%), and Gemma3-12B TG-3FM LoRA variant achieves the highest MCQ retention (76.4%) with competitive RRB recall (48.4%).

6 Conclusion

We introduced **ToolSense**, a diagnostic framework that auto-generates three benchmarks—RRB, MCQ, and QA—from any tool catalog to probe parametric retrieval systems along orthogonal axes: OOD generalization, trie-dependency, and factual knowledge retention. Applying ToolSense to ToolBench reveals that high constrained recall on verbose in-distribution queries is a poor proxy for real-world capability: performance collapses on realistic queries, hierarchical tokens exhibit deep trie-dependency, and Stage 2 retrieval fine-tuning systematically erodes the tool knowledge built during Stage 1, leaving models that behave more like lookup tables than knowledge bases. LoRA combined with multi-format Stage 1 memorization best mitigates these failure modes, and Stage 1 MCQ accuracy is a reliable early predictor of downstream OOD generalization. We release the ToolSense framework and the ToolBench diagnostic benchmarks to encourage further research.

Limitations

While ToolSense's benchmarks are designed to generalize to other parametric retrieval designs, our empirical study focuses on ToolGen's two-stage paradigm (Wang et al., 2025) applied to ToolBench; extending the diagnosis to other catalog-agnostic parametric designs is a natural next step. The RRB, MCQ, and QA benchmarks are LLM-generated and validated by human annotators on 100 samples each ($\kappa \geq 0.805$); scaling the human study beyond 100 samples per benchmark would further strengthen confidence in the benchmark quality, though doing so at catalog scale involves non-trivial annotation cost and time. Our analysis focuses on the retrieval stage in isolation; end-to-end agentic evaluation, where retrieval quality interacts with planning and execution, is a promising direction for future work. The open-source base models we study (Gemma3, Qwen) may have encountered RapidAPI tool documentation during pre-training, potentially inflating absolute Stage 1 knowledge scores relative to truly novel tools; the relative forgetting patterns across Stage 1→Stage 2 remain unaffected by this confound, though our findings may not directly transfer to private enterprise tool catalogs whose APIs never appear in any public pre-training corpus. The Internalization Score is a ratio metric and can exhibit high variance when $R_c@k$ is low; while we discuss and handle the degenerate $R_c@k = 0$ case explicitly and reported confidence intervals already capture this variance, future work should evaluate log-ratio or calibration-based formulations to confirm the stability of IS findings across catalog sizes. Finally, our model experiments cover the 4B–12B parameter range; studying very large models (≥30B) may reveal additional nuances in how model scale affects knowledge retention under sequential fine-tuning.

Ethical Considerations

We conducted experiments within the provisions of the ACL Ethics Policy and relevant research-integrity guidelines. There are, to the best of our knowledge, no remaining ethical risks that have not been addressed.

References

Anonymous. 2026. CoHyDE: Iterative co-training of LLM rewriter & dense encoder for tool retrieval. Under review.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John Patrick Cunningham. 2024. LoRA learns less and forgets less. *Transactions on Machine Learning Research*. Featured Certification.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

Jacob Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences*, 2nd edition. Lawrence Erlbaum Associates, Hillsdale, NJ.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 4582–4597, Online. Association for Computational Linguistics.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*.

OpenAI. 2024. New embedding models and API updates. Accessed: 2026-05-23.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive APIs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Aleksandar Petrov, Philip Torr, and Adel Bibi. 2024. When do prompting and prefix-tuning work? a theory of capabilities and limitations. In *The Twelfth International Conference on Learning Representations*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, and 24 others. 2024a. Tool learning with foundation models. *ACM Comput. Surv.*, 57(4).

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024b. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer memory as a differentiable search index. In *Advances in Neural Information Processing Systems*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

William Timkey and Marten van Schijndel. 2021. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4527–4546, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. 2025. Toolgen: Unified tool retrieval and calling via generation. In *The Thirteenth International Conference on Learning Representations*.

Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Sun, Weiwei Deng, Qi Zhang, and Mao Yang. 2022. A neural corpus indexer for document retrieval. In *Advances in Neural Information Processing Systems*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

A ToolBench Standard Splits vs. RRB: Query Style Comparison

Figure 8 illustrates a fundamental difference in query style between the ToolBench standard evaluation splits (G1/G2/G3) and our Realistic Retrieval Benchmark (RRB). For the same underlying tools, ToolBench queries are verbose narrative prompts that effectively paraphrase the API documentation. A model that has memorized tool descriptions at Stage 1 can retrieve the correct token by matching surface-level vocabulary, without genuine semantic understanding.

RRB queries, by contrast, are generated by θ_{RRB} with an explicit instruction to produce *intent-focused* natural language: a concise statement of what the user wants to accomplish, stripped of API-specific terminology. The four examples shown span distinct domains, yet in each case, the RRB query is how a real user would type the request into a chat interface: "Show me pre-game odds for today's matches," "Create a branded short link for our product page," "Find me streaming options for Stranger Things," and "What are the different marketplace categories I can browse for items?" This shift from documentation-paraphrase to user-intent query is precisely the distribution gap that causes retrieval collapse in our experiments (Section 5): a parametric model trained exclusively on Stage 2 (query $\rightarrow$ virtual token) pairs derived from verbose ToolBench-style prompts has not learned to bridge the lexical gap to short, colloquial, intent-focused user utterances.

B Multi-Choice Tool Selection (MCTS) Memorization Dataset

The standard Stage 1 memorization format trains the model on (tool metadata $\rightarrow$ virtual token) pairs (Wang et al., 2025). While effective at associating a tool with its token, this objective does not explicitly teach the model to *distinguish* semantically similar tools — a critical requirement in large catalogs where many tools share overlapping descriptions.

Format. MCTS is a discriminative Stage 1 objective. For each tool τ, we construct a multiple-choice instance by pairing its description $\text{desc}(\tau)$ with K hard-negative tool descriptions $\mathcal{H}_K(\tau) = \{\tau_1^-, \ldots, \tau_K^-\}$ retrieved by embedding-based similarity. At training time, the model receives the description of τ alongside $K{+}1$ tool tokens corresponding to the candidates (shuffled) and must generate the virtual token v_τ corresponding to the ground-truth tool. The training signal is therefore a next-token loss conditioned on a *contrast set*, forcing the model to resolve fine-grained semantic boundaries rather than simply recall a memorized association.

Hard-negative mining. Candidates are retrieved using a dense embedding model (`text-embedding-3-large`) indexed over the full tool catalog via chromadb. For each anchor tool, we over-fetch $K{+}5$ nearest neighbors by cosine similarity and remove the anchor itself, yielding a pool of K hard negatives. Because these tools are semantically proximate but functionally distinct (e.g., two overlapping enterprise search APIs, or two QR-code generation services), they represent the most plausible confusables — exactly the cases where a model relying on shallow lexical overlap would fail. For our experiments, we set $K = 4$.

Effect on Stage 1 representations. By requiring the model to correctly identify v_τ against its nearest semantic neighbors, MCTS encourages the embedding of v_τ to be positioned not just *near* its own description, but *farther* from confusable alternatives. This discriminative geometry is more resistant to the overwriting that occurs during Stage 2 retrieval SFT, as reflected in the higher Stage 2 MCQ scores observed for configurations that include MCTS in their Stage 1 training mix (TG-3FM, TG-5FM; Table 1).

C Training Configuration Details

C.1 Stage 1: Memorization Formats

Stage 1 fine-tunes the model to associate each tool's virtual token with its metadata. We define five memorization formats, each presenting the task from a different angle. All formats share the same tool catalog but differ in what is given as input and what the model is trained to generate. For applicable model configurations, exact system prompts are provided in Appendix K.1.

desc$\rightarrow$tok (standard). The model receives the full tool description and is trained to generate the virtual token. This is the format introduced by

Tool	ToolBench standard splits (verbose, fully-specified)	RRB (realistic, short)
API: Betcity **Endpoint:** Odds sports betting odds	"I am curious about the odds for the upcoming boxing matches. Can you provide me with the latest odds for the Tyson Fury vs. Anthony Joshua and Canelo Alvarez vs. Gennady Golovkin fights? Additionally, I would like to see the latest matches in the boxing world."	"Show me pre-game odds for today's matches."
API: URL Link Shortener **Endpoint:** Create a new link link shortening	"I'm planning a surprise baby shower for my best friend. Can you generate a short URL for the baby shower invitation? Also, suggest some adorable and creative slashtags for the event. Furthermore, provide me with a list of available domains."	"Create a branded short link for our product page."
API: Streaming Availability **Endpoint:** Search By Title streaming lookup	"I'm planning a weekend getaway with my family and we want to watch some movies during our trip. Can you recommend some popular movies available for streaming in Australia? Also, provide us with the streaming sources."	"Find me streaming options for Stranger Things."
API: Amazon Data_v2 **Endpoint:** MarketPlace List marketplace listing	"I'm organizing a charity event and I need to gather information about various marketplaces. Can you provide me with a list of marketplaces we can use to fetch data? Additionally, fetch the comments for a specific product with the ID B094NVF2V1..."	"What are the different marketplace categories I can browse for items?"

Figure 8: Side-by-side comparison of ToolBench standard evaluation split queries (G1/G2/G3) and RRB queries for the same tools. ToolBench queries are verbose, story-wrapped, and enumerate specific parameters. Whereas, RRB queries are short, intent-focused, and reflect how a real user would naturally phrase a request.

Wang et al. (2025) and used in *all* configurations, regardless of token scheme. The form of `tok` differs by scheme: for flat tokens it is a single atomic identifier encoding the full tool identity (`«API&&ENDPOINT»`); for hierarchical tokens it is the joint two-token sequence (`«API»«ENDPOINT»`), generated autoregressively as two successive decode steps.

> *Input:* "Manages and administers search queries within the Enterprise Search Administration system..."
> *Target (flat):* `«ESH_ADMIN_SRV_0001&&SearchQueries»`
> *Target (hierarchical):* `«ESH_ADMIN_SRV_0001»«SearchQueries»`

tok→desc (reverse). The input and target are swapped: the model receives the virtual token and is trained to generate the tool description. This forces bidirectional association—the model must encode not only *which description maps to which token* but also *which token maps to which description*—making the embedding more resistant to overwriting during Stage 2.

> *Input (flat):* `«ESH_ADMIN_SRV_0001&&SearchQueries»`
> *Input (hierarchical):* `«ESH_ADMIN_SRV_0001»«SearchQueries»`
> *Target:* "Manages and administers search queries..."

desc→api_tok (hierarchical only). The model receives the tool description and is trained to generate only the API-level token—the first step of the two-token hierarchical sequence. This format is specific to hierarchical token configuration TG-5FM and provides an additional supervised signal that trains the first generation step in isolation, supplementing the primary `desc→tok` format which already trains both tokens jointly end-to-end.

> *Input:* "Manages and administers search queries..."
> *Target:* `«ESH_ADMIN_SRV_0001»`

desc+api_tok→endpoint_tok (hierarchical only). The model receives the tool description with the API-level token and is trained to generate the endpoint-level token, the second step of the hierarchical sequence. Together with desc→api_tok, this provides two additional supervised signals that train each hierarchical step independently, complementing the joint `desc→tok` objective already present in TG-5FM. This decomposition reinforces each level of the hierarchy with a dedicated loss signal.

> *Input:* "Manages and administers search queries... `«ESH_ADMIN_SRV_0001»`"
> *Target:* `«SearchQueries»`

MCTS (discriminative). See Appendix B for a full description. In brief, the model must identify the correct virtual token from a set of $K+1$ tool tokens, which include ground truth tool token as well as K hard-negative candidates retrieved by embedding similarity, adding a discriminative objective alongside the generative formats above.

C.2 Stage 2: Retrieval Training Data

Stage 2 fine-tunes the model on (query → virtual token) pairs drawn from the ToolBench training split (Qin et al., 2024b), following Wang et al. (2025). The Stage 2 data is *identical* across all five

configurations; no format-specific data augmentation is applied at this stage. The only variation is whether a system prompt is prepended to each training instance: TG omits it entirely, while all other configurations (TG-SP, TG-3FM, TG-H, TG-5FM) include the same task-specific system prompt used during Stage 1, ensuring consistency between the training stages and inference. For applicable model configurations, exact system prompts are provided in Appendix K.2.

C.3 Design Rationale per Configuration

The five configurations isolate one variable at a time against a shared baseline:

- **TG vs. TG-SP**: isolates the effect of the system prompt, all else equal.
- **TG-SP vs. TG-3FM**: isolates the effect of enriching Stage 1 with reverse (tok→desc) and discriminative (MCTS) objectives, holding token format and system prompt constant.
- **TG-SP vs. TG-H**: isolates the effect of switching from flat to hierarchical tokens, holding memorization format (desc→tok only) and system prompt constant.
- **TG-H vs. TG-5FM**: isolates the effect of adding all five memorization formats to the hierarchical token baseline.
- **FFT vs. LoRA** (within any configuration): isolates the effect of constraining parameter update magnitude on knowledge retention across the sequential training stages (Biderman et al., 2024).

D Training Hyperparameters

All experiments were run on a single node with 8 NVIDIA H200 GPUs. Each model was trained on a single GPU, with the AdamW optimizer, bf16 mixed precision, and a maximum sequence length of 1024 tokens. The effective batch size is 1024 for all runs (achieved via gradient accumulation: 1 GPU $\times$ 2 samples $\times$ 512 accumulation steps for 4B models; 2 GPUs $\times$ 2 $\times$ 256 for 12B models).

LoRA Configuration

Adapters are applied to all linear projection layers (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`) with rank r=64, α=128, and zero dropout. The full embedding layer (`embed_tokens`) is kept fully trainable in all LoRA runs; weight tying between `embed_tokens` and `lm_head` is preserved. LoRA Stage 2 runs initialise from the merged Stage 1 LoRA checkpoint.

Hyperparameter	FFT	LoRA
Stage 1 — Memorization (2 epochs)		
Learning rate	2×10^{-5}	1×10^{-4}
LR scheduler	Cosine	Cosine
Min LR floor	2×10^{-6}	1×10^{-5}
Stage 2 — Retrieval SFT (1 epoch)		
Learning rate	2×10^{-5}	1×10^{-4}
LR scheduler	Linear	Linear
Shared		
Warmup ratio	0.03	
Effective batch	1024	

Table 6: Training hyperparameters for both stages.

E Realistic-Query Stage 2 Training

To test whether the generalization collapse is caused by query distribution shift or a deeper property of the Stage 2 training objective, we retrained the TG (Gemma3-4B) configuration using RRB-style queries in place of the standard verbose ToolBench queries.

Training data. We used ToolSense to generate realistic queries for all ∼47k tools in the ToolBench catalog. Since RRB queries can have multiple correct tools (Medium and Hard tiers require 2–3 and 4+ tools respectively), each generated sample $(q, [t_1, \ldots, t_n])$ is expanded into n individual training pairs $(q, t_1), \ldots, (q, t_n)$, yielding 284,567 training samples in total. The 500-query RRB evaluation set is disjoint from this training corpus. All other Stage 2 hyperparameters are identical to the standard TG run (see Appendix D).

Results. Table 7 compares TG trained on standard vs. RRB-style Stage 2 data. RRB R_c@50 rises by +44.0pp (43.8%→87.8%), demonstrating that the model has sufficient capacity to handle realistic queries when trained on them. The cost is a 9.1pp drop on G1 verbose recall (95.7%→86.6%), confirming that the two query styles induce complementary but non-overlapping retrieval behaviors and the model specializes to whichever distribution it trains on. Critically, MCQ accuracy drops from 31.5% to 26.0% (near-random), and QA from 50.0% to 46.6%, despite the substantial improvement in retrieval. This demonstrates that Stage 2 SFT erodes parametric tool knowledge regardless of whether training queries are verbose or realistic,

implicating the retrieval fine-tuning objective itself rather than the query distribution.

F Internalization Score: Rationale and Degenerate-Case Handling

What IS measures and what it does not. IS is a *conditional* metric: it asks, "the tool knowledge the model has already acquired (as measured by $R_c@k$), how much of it has been internalized into the weights rather than delegated to the trie?" It is *not* a measure of how much the model has learned overall, as that role belongs to $R_c@k$ itself. Consider the case where $R_c@50 = 0.10$ and $R_f@50 = 0.10$: recall scores are low, indicating the model has learned only about small fraction of tools; but whatever it did learn can be generated freely without constraint, giving IS $= 1.0$. This is not a failure of the metric, instead the correct reading would be: *whatever small amount was learned is fully internalized*. The learning deficit is plainly visible in R_c; IS tells us about the quality of that learning, not its quantity. For this reason we always report R_f, R_c, and IS together (see Tables 8 and 9) so the reader can distinguish low-recall-but-internalized from high-recall-but-trie-dependent regimes at a glance.

Degenerate case: $R_c@k = 0$. When constrained recall is exactly zero the ratio is undefined; we define IS $= 0$ in this case (reflected in Eq. 2). Operationally, $R_c = 0$ means the model cannot recall any correct tool even with full trie guidance, so there is nothing to internalize. Assigning IS $= 0$ rather than leaving it undefined prevents spurious high-IS readings caused by a model that is uniformly broken.

Relationship to a difference-based formulation. An alternative design would use $\Delta@k = R_f@k - R_c@k$. This directly measures the absolute gap but conflates two distinct phenomena: a model with $R_c = 0.90$, $R_f = 0.80$ (IS $= 0.89$, $\Delta = -0.10$) and one with $R_c = 0.10$, $R_f = 0.00$ (IS $= 0$, $\Delta = -0.10$) would produce the same difference score despite having very different failure modes. The ratio, always paired with R_c, is more interpretable: IS $= 0.89$ at $R_c = 0.90$ means a high-recall model is mostly but not fully internalized; IS $= 0$ at $R_c = 0.10$ means a low-recall model is entirely trie-dependent.

G Internalization Score: Full Results with Confidence Intervals

Tables 8 and 9 report IS@50 for all evaluated configurations, including 95% bootstrap confidence intervals on all four evaluation splits. CIs for IS are derived via the delta method from the bootstrap CIs of $R_f@50$ and $R_c@50$: letting $\hat{\sigma}_f$ and $\hat{\sigma}_c$ be the half-widths of the respective 95% bootstrap intervals divided by 1.96, $\sigma_{\text{IS}} \approx \text{IS} \cdot \sqrt{(\hat{\sigma}_f/R_f)^2 + (\hat{\sigma}_c/R_c)^2}$, and the CI is $[\text{IS} - 1.96\,\sigma_{\text{IS}},\ \text{IS} + 1.96\,\sigma_{\text{IS}}]$, clipped to $[0, 1]$.

The horizontal rule within each table separates flat-token configurations (top) from hierarchical-token configurations (bottom). Figures 9 and 10 show the IS@50 training dynamics for different configurations across all splits, complementing the main-body Figure 3.

Several patterns are consistent across architectures and sizes: (i) flat configurations typically achieve IS$>$0.65 on both G1 and RRB, with LoRA variants systematically higher (≥ 0.80); (ii) hierarchical configurations show a pronounced penalty on RRB—TG-H drops to IS=0.33 on RRB vs. 0.42 on G1 for Gemma3-4B, and this pattern holds for Qwen3.5-4B (0.28 on RRB); (iii) G3 IS values are frequently 1.00 or near-1.00 for flat configurations, suggesting that out-of-distribution single-API queries are handled without trie support; (iv) CI widths are wider on RRB than on ToolBench evaluation splits (G1/G2/G3), reflecting the models cause greater instability in consistently producing the tool tokens in organic manner.

H Cross-Architecture Probing Results

Table 11 reports MCQ and QA probing accuracy for Qwen3.5-4B and Gemma3-12B, complementing the Gemma3-4B results in Table 4. Table 10 lists all 14 model variants included in the Stage 1 MCQ vs. Stage 2 RRB correlation analysis (Figure 4).

The cross-architecture results reveal a consistent pattern: larger and more capable base models enter Stage 2 with substantially stronger parametric tool knowledge (Stage 1 MCQ of 73–75% for Qwen3.5-4B and Gemma3-12B, vs. 23–55% for Gemma3-4B) and largely *retain* that knowledge after retrieval fine-tuning. In contrast to Gemma3-4B FFT variants which suffer severe MCQ drops (e.g., TG drops from 55.4% to 31.4%), the Qwen3.5-4B and Gemma3-12B configurations maintain or even slightly improve MCQ scores — confirming

Training data	RRB R_c@50	G1 R_c@50	G2 R_c@50	G3 R_c@50	MCQ	QA
Standard (verbose)	43.8 [40.3, 47.7]	95.7 [94.2, 97.0]	84.5 [81.5, 87.3]	84.6 [80.4, 88.5]	31.5 [27.6, 35.5]	50.0 [45.8, 54.0]
RRB-style (realistic)	87.8 [85.6, 90.2]	86.6 [84.2, 89.1]	67.2 [63.6, 70.7]	61.8 [54.7, 69.6]	26.0 [22.4, 30.0]	46.6 [42.4, 50.8]

Table 7: TG Gemma3-4B: standard vs. RRB-style Stage 2 training (final checkpoint). 95% bootstrap CIs in brackets.

Config	G1 IS@50	G2 IS@50	G3 IS@50	Hard IS@50
TG Gemma3-4B	0.75 [0.72,0.79]	0.87 [0.82,0.93]	1.00 [0.93,1.00]	0.85 [0.74,0.96]
TG (LoRA) Gemma3-4B	0.55 [0.50,0.59]	0.70 [0.64,0.75]	0.95 [0.87,1.00]	0.80 [0.68,0.93]
TG-SP Gemma3-4B	0.70 [0.66,0.74]	0.70 [0.64,0.76]	0.65 [0.55,0.75]	0.75 [0.65,0.85]
TG-3FM Gemma3-4B	0.69 [0.65,0.73]	0.78 [0.73,0.84]	1.00 [0.93,1.00]	0.82 [0.71,0.93]
TG-3FM (LoRA) Gemma3-4B	0.89 [0.86,0.93]	0.92 [0.87,0.98]	1.00 [0.94,1.00]	0.81 [0.71,0.91]
TG-H Gemma3-4B	0.42 [0.38,0.46]	0.41 [0.36,0.46]	0.62 [0.53,0.70]	0.33 [0.24,0.41]
TG-5FM Gemma3-4B	0.84 [0.80,0.87]	0.78 [0.72,0.84]	0.52 [0.42,0.61]	0.64 [0.53,0.76]
TG-5FM (LoRA) Gemma3-4B	0.66 [0.62,0.70]	0.79 [0.73,0.85]	0.61 [0.50,0.72]	0.79 [0.65,0.92]

Table 8: IS@50 with 95% confidence intervals for training Gemma3-4B (Stage 2). Values in brackets are 95% CIs.

that knowledge destruction is more pronounced in smaller models fine-tuned with full parameter updates. Hierarchical tokenization (TG-H) remains the most destructive configuration across architectures: Qwen3.5-4B TG-H drops from 39.7% to 31.7% MCQ, reinforcing the earlier finding that collapsing tool tokens into path-structured representations impedes semantic retention regardless of model scale. LoRA consistently provides the best knowledge preservation for Qwen3.5-4B (TG-3FM LoRA retains 74.2% MCQ through Stage 2), confirming its role as the safest fine-tuning strategy when knowledge retention is required.

I Virtual Token Embedding Drift Analysis

To understand what Stage 2 SFT modifies mechanistically, we measure the *relative L2 drift* of virtual token embeddings between Stage 1 and Stage 2 checkpoints. For a token index i, define:

$$d_{\text{rel}}(i) = \frac{\|\mathbf{E}_{S2}[i] - \mathbf{E}_{S1}[i]\|_2}{\|\mathbf{E}_{S1}[i]\|_2} \tag{3}$$

where $\mathbf{E}_{S1}, \mathbf{E}_{S2} \in \mathbb{R}^{V \times d}$ are the embedding matrices at Stages 1 and 2 (d=3840 for Gemma-3-4B-IT). We compare virtual tokens against 5,000 randomly sampled base-vocabulary tokens using a one-sided Mann-Whitney U test. Results across all seven Gemma3-4B configurations are shown in Figure 7.

Finding 1: Stage 2 selectively repositions virtual tokens (all configs, $p < 2{\times}10^{-308}$). Drift ratios range from 1.9× to 22.9× depending on training regime. This rules out uniform global drift: if Stage 2 were broadly fine-tuning all weights, virtual and regular tokens would experience similar relative displacement. In FFT models (2–3× ratio), virtual tokens appear in *every* training example as generation targets while regular tokens appear sparsely in user queries; by the law of large numbers over mini-batches, virtual tokens accumulate proportionally larger gradient updates (Li and Liang, 2021; Lester et al., 2021).

Finding 2: LoRA creates extreme isolation (22.9×). Under LoRA (Hu et al., 2022), adapters are attached to linear layers while the full embedding layer remains trainable. Despite this, base vocabulary tokens drift by only $D_{\text{rel}}(\text{RAND})$=0.0015 — effectively numerical noise — because they appear solely in input queries and their gradients must back-propagate through the frozen backbone layers before reaching the embedding table, arriving with negligible magnitude. Virtual tokens, as generation targets in every Stage 2 training example, receive direct strong gradient from the output cross-entropy loss. This input/output asymmetry, amplified by backbone freezing, is responsible for the 22.9× drift ratio.

Finding 3: Hierarchical tokens occupy a near-isotropic geometric regime. Flat tokens have mean pairwise cosine similarity $\text{cosim}_{S1}{\approx}0.37$, a signature of embedding anisotropy produced when all tool tokens share similar Stage 1 generation objectives, pushing them into a common semantic cone (Gao et al., 2019; Ethayarajh, 2019; Timkey and van Schijndel, 2021). Hierarchical tokens have $\text{cosim}_{S1}{\approx}0.038$, approaching the isotropic baseline expected for randomly initialised unit vectors. For Gemma-3's weight-tied architecture ($\mathbf{W}_{\text{head}}$=$\mathbf{E}$), the trie-routing logit margin is

Config	G1 IS@50	G2 IS@50	G3 IS@50	Hard IS@50
TG-H Qwen3.5-4B	0.57 [0.53,0.61]	0.49 [0.43,0.54]	0.64 [0.55,0.73]	0.28 [0.21,0.35]
TG-3FM Qwen3.5-4B	0.79 [0.75,0.83]	0.78 [0.72,0.85]	1.00 [0.91,1.00]	0.81 [0.73,0.89]
TG-3FM (LoRA) Qwen3.5-4B	0.68 [0.64,0.72]	0.77 [0.71,0.83]	0.75 [0.66,0.85]	0.82 [0.73,0.92]
TG-SP Qwen3.5-4B	0.90 [0.86,0.93]	0.97 [0.91,1.00]	1.00 [0.92,1.00]	0.98 [0.89,1.00]
TG-SP (LoRA) Qwen3.5-4B	0.94 [0.92,0.97]	0.91 [0.85,0.96]	1.00 [0.92,1.00]	0.99 [0.89,1.00]
TG-3FM (LoRA) Gemma3-12B	0.76 [0.72,0.80]	0.85 [0.80,0.90]	0.96 [0.88,1.00]	0.86 [0.76,0.96]
TG-SP Gemma3-12B	0.66 [0.62,0.70]	0.85 [0.80,0.90]	0.65 [0.54,0.75]	0.84 [0.75,0.93]

Table 9: IS@50 with 95% confidence intervals for training Qwen3.5-4B and Gemma3-12B (Stage 2).

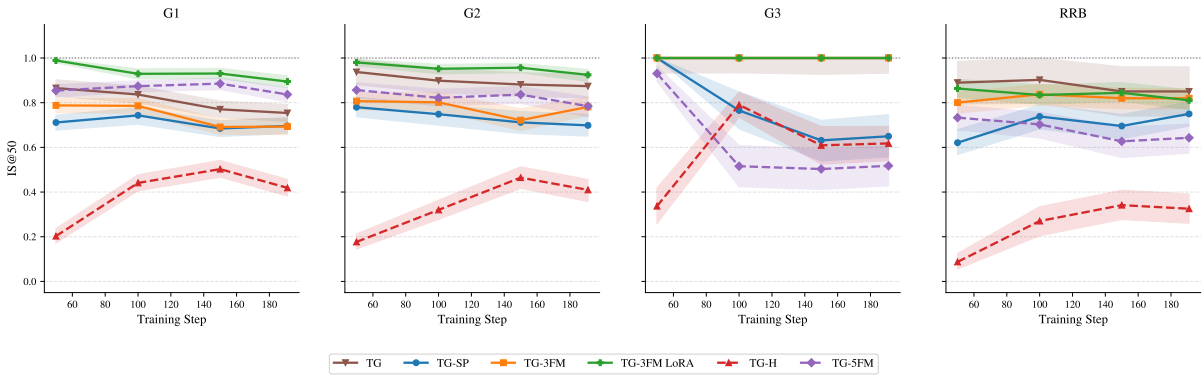


Figure 9: IS@50 over Stage 2 training for Gemma3-4B across all four evaluation splits. Shaded bands are 95% bootstrap CIs.

Model family	Configuration
Gemma3-4B	TG TG-SP TG-3FM TG-3FM (LoRA) TG-H TG-5FM TG-5FM (LoRA)
Qwen3.5-4B	TG-3FM TG-3FM (LoRA) TG-H TG-SP TG-SP (LoRA)
Gemma3-12B	TG-3FM (LoRA) TG-SP

Table 10: All 14 model variants included in the Stage 1 MCQ vs. Stage 2 RRB R_c@50 correlation plot (Figure 4).

$l_i - l_j = (\mathbf{E}[i] - \mathbf{E}[j])^\top \mathbf{h}$; near-orthogonal embeddings maximise this margin, so hierarchical tokens are better conditioned for constrained decoding from the start (Cao et al., 2021). Consequently, hierarchical tokens require 7–24× less Stage 2 repositioning (TG-3FM flat: d_{rel}=0.024; TG-5FM hier: d_{rel}=0.003).

Finding 4: No representation collapse ($|\Delta\text{cosim}| < 0.002$). Despite targeted repositioning, pairwise cosine similarity of virtual tokens is essentially unchanged by Stage 2 across all configurations. This rules out the hypothesis that Stage 2 collapses virtual tokens into undifferentiated routing codes. Each token retains its Stage-1-acquired geometric identity; what changes is absolute position (drift), not relative configuration (cosim). The trie objective structurally prevents collapse: if all tokens became identical, constrained beam search could not route to specific tools.

Model	Config	MCQ (S1→S2)		QA (S1→S2)	
		S1	S2	S1	S2
Qwen3.5-4B	TG-3FM	**74.0 [70.4,77.8]**	72.8 [69.2,76.6]	53.6 [49.2,58.0]	54.2 [49.8,58.8]
	TG-3FM (LoRA)	73.8 [69.6,77.8]	**74.2 [70.6,78.0]**	**64.2 [60.2,68.2]**	54.4 [50.0,58.8]
	TG-H	39.7 [35.3,44.2]	31.7 [27.6,36.1]	49.6 [44.8,54.0]	50.8 [46.6,55.2]
	TG-SP	63.1 [58.9,67.3]	61.7 [57.5,65.7]	51.6 [47.8,56.2]	48.0 [43.8,52.4]
	TG-SP (LoRA)	64.9 [60.5,69.2]	49.0 [44.6,53.8]	51.4 [47.0,55.6]	**51.6 [47.8,56.2]**
Gemma3-12B	TG-3FM (LoRA)	**75.0 [71.0,78.8]**	**76.4 [72.6,80.0]**	62.4 [57.8,66.8]	65.6 [61.4,69.6]
	TG-SP	60.5 [56.1,65.1]	66.3 [62.1,70.4]	**66.4 [62.0,70.6]**	**71.2 [67.0,75.2]**

Table 11: MCQ and QA probing accuracy for Qwen3.5-4B and Gemma3-12B. Random baselines: MCQ=25%, QA=50%. 95% CI in brackets.

Connection to knowledge-retrieval dissociation. Despite the targeted and statistically significant Stage 2 updates, MCQ/QA accuracy on tool knowledge degrades, confirming that Stage 2 optimizes virtual token positions as trie-routing codes rather than encoding semantic tool knowledge. Even hi-

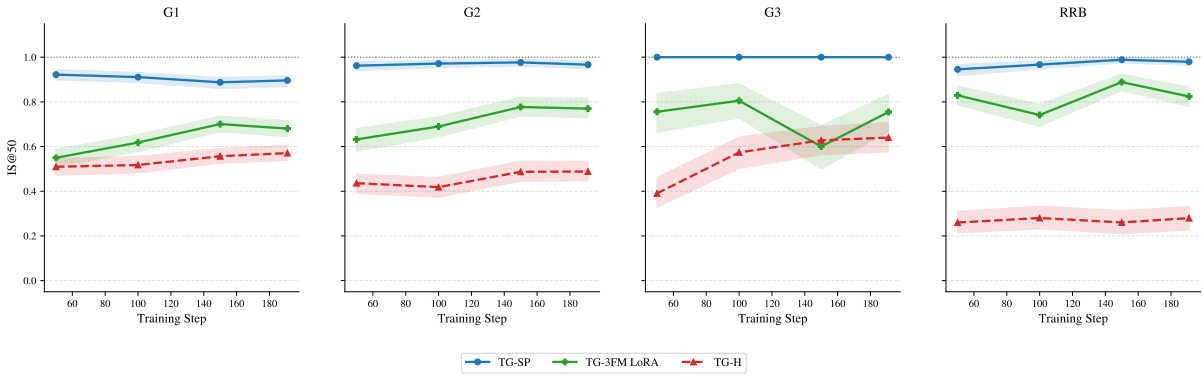


Figure 10: IS@50 over Stage 2 training for Qwen3.5-4B across all four evaluation splits. Shaded bands are 95% bootstrap CIs.

erarchical tokens, geometrically superior in every metric and requiring minimal Stage 2 adaptation, fail MCQ probes, closing the confound that poor token geometry explains the dissociation. This is consistent with causal analyses locating factual recall in MLP layers of mid-to-upper transformer blocks (Meng et al., 2022; Dai et al., 2022; Petroni et al., 2019): Stage 2 targets the input embedding layer, leaving factual recall pathways untouched (Petrov et al., 2024).

J Benchmark Generation Details

All three ToolSense benchmarks (RRB, MCQ, QA) were generated using **claude-4.5-sonnet** for both generation and judging calls, accessed via a LiteLLM proxy. Hard-negative retrieval for the RRB pipeline used OpenAI `text-embedding-3-large` embeddings stored in a ChromaDB vector database. Judge calls were made at temperature $= 0.0$; generation calls used the model's default sampling temperature.

J.1 Realistic Retrieval Benchmark (RRB) Generation Pipeline

Seed selection. Anchor tools are sampled from the ToolBench catalog using stratified sampling across service domains (parsed from the `Service&&Method` name format), with proportional allocation and a minimum of one seed per domain.

Hard-negative pool construction. For each anchor tool τ, 13 hard negatives are retrieved from the full catalog via cosine similarity over `text-embedding-3-large` embeddings of tool descriptions. The candidate pool is $\mathcal{P}(\tau) = \{\tau\} \cup \{13 \text{ hard negatives}\}$, giving 14 total candidates.

Generation and validation pipeline. Generation is implemented as a LangGraph `StateGraph` with three tier subgraphs (easy, medium, hard) running in parallel via fan-out/fan-in. Each subgraph follows: *generate* $\rightarrow$ *programmatic filter* $\rightarrow$ *LLM judge* $\rightarrow$ (accept or retry with feedback). Up to three retries are allowed per sample; on retry, the judge's rejection reason is injected into the next generation prompt. The programmatic filter checks that all answer tool names appear verbatim in the candidate pool, preventing hallucinated tool names.

Tier specifications. Easy tier queries must resolve to exactly 1 tool, remain under 20 words, and be phrased in business language. Medium tier queries must be genuinely ambiguous across 2–3 tools (overlapping functional domains), under 25 words. Hard tier queries describe a high-level business goal spanning 4+ tools, in 1–3 sentences. Default sample targets per seed: 3 (easy), 3 (medium), 2 (hard).

J.2 MCQ and QA Probing Benchmark Generation

Both benchmarks follow the same two-step generate-then-judge pattern: (1) a generation call produces a candidate item; (2) a judge call at temperature $= 0.0$ accepts or rejects it. Items failing the judge or flagged by the generator as unanswerable (`skip=true`) are discarded.

MCQ. For each tool, the generator produces a 4-way multiple-choice question: one correct answer and three plausible-but-wrong distractors, all as short phrases (2–8 words). Questions must use *"this tool"* as a placeholder and test a specific, verifiable property. The judge verifies all five criteria: specificity, correctness, distractor plausibility, op-

tion distinctiveness, and placeholder usage. Final dataset: 496 items.

QA. For each tool, the generator produces one binary yes/no question with a pre-specified target answer (alternated across tools for label balance). Questions must use *"this tool"* as a placeholder and test a verifiable property (modality, domain, input/output type, etc.). The judge verifies four criteria: specificity, correctness, placeholder usage, and verifiability. Final dataset: 500 items.

K System Prompts

We report the exact user-turn prompts used for experiments conducted in this study. Placeholders in `{braces}` are filled at runtime with the corresponding value.

K.1 Memorization Format Prompts

The `desc→tok` format (Figure 11), used in all configurations, asks the model to predict the virtual token from the tool's JSON description. For flat tokens the expected output is a single atomic identifier (`«API&&ENDPOINT»`); for hierarchical tokens it is the joint two-token sequence (`«API»«ENDPOINT»`) generated autoregressively.

The `tok→desc` format (Figure 12), used in TG-3FM and TG-5FM, reverses the mapping: the model receives the virtual token and is asked to generate the plain-language tool description. This bidirectional objective makes the association more resistant to overwriting during Stage 2.

The `desc→api_tok` format (Figure 13), used only in TG-5FM, restricts the target to the API-level token alone, providing an additional supervised signal that trains the first hierarchical generation step independently.

The `desc+api_tok→endpoint_tok` format (Figure 14), also TG-5FM only, gives the model both the tool description and the API-level token and asks it to predict the endpoint-level token, training the second hierarchical step independently.

The MCTS format (Figure 15), used in TG-3FM and TG-5FM, presents a discriminative selection task: the model must choose the correct virtual token from $K + 1$ hard-negative candidates retrieved by embedding similarity. See Appendix B for construction details.

K.2 Retrieval Training Prompt

Figure 16 shows the Stage 2 prompt shared by all configurations that use a system prompt (TG-SP, TG-3FM, TG-H, TG-5FM). The model receives a natural-language user query and must predict the corresponding virtual tool token. The no-prompt baseline TG omits this prompt entirely; all other configurations use it consistently across Stage 2 training and inference.

> You will be provided with the information about the tool in the JSON format. Your task is to predict the tool that corresponds to the given tool information. You only need to predict the tool. Please don't include any additional text in your response. The tool information is provided below:
>
> {tool_info}

Figure 11: `desc→tok` prompt (all configurations).

> You will be provided with a tool representation tokens. Your task is to describe what this tool does in plain language. Please don't include any additional text in your response other than the description. The tool representation is provided below:
>
> {tool_token}

Figure 12: `tok→desc` prompt (TG-3FM, TG-5FM).

K.3 Benchmark Generation Prompts

All benchmark generation prompts were used with **claude-4.5-sonnet** via a LiteLLM proxy. Figures 17 and 18 show the generation and judge prompts for the QA probing benchmark; Figures 19 and 20 show the corresponding prompts for MCQ. All judge prompts run at temperature $= 0.0$. Figure 21 shows the full RRB easy-tier generation prompt (Jinja2 template); the medium- and hard-tier prompts share the same structure with tier-specific rules substituted in. Figure 22 shows the shared RRB judge prompt, parameterised by the `{{complexity}}` variable.

> You are building a factual probing benchmark for AI tool retrieval research.
> Below is the description of an API or software tool:
> {description}
> Your task: generate one yes/no question that tests knowledge of a specific, verifiable functionality or capability of this tool. The correct answer MUST be "{target_answer}".

You will be provided with the information about the tool in the JSON format. Your task is to identify the API this tool belongs to and produce its corresponding API token. Only respond with the API token and nothing else. The tool information is provided below:

{tool_info}

Figure 13: `desc→api_tok` prompt (TG-5FM only).

You will be provided with the information about the tool in the JSON format along with its API. Your task is to identify the Endpoint this tool belongs to within that API and produce its corresponding Endpoint token. Only respond with the Endpoint token and nothing else. The tool information and API are provided below:

Tool Information: {tool_info}

API: {api_token}

Figure 14: `desc+api_tok→endpoint_tok` prompt (TG-5FM only).

Rules:
1. The answer to your question must be exactly "{target_answer}" based on the description.
2. Use "this tool" in the question — never include the actual tool name or service name.
(The model will see only a token at inference time, so the name must not be a hint.)
3. The question must be specific to THIS tool. "Does this tool provide an API?" is too generic.
Good examples for Yes: "Does this tool process image inputs?", "Is this tool designed for financial data?"
Good examples for No: "Does this tool support voice/audio input?", "Does this tool return results in XML?"
For No questions: ask about a plausible capability or data type that the description does NOT mention
(e.g. a different modality, format, or domain that a user might expect but this tool doesn't handle).
4. Set skip=true if you cannot form a specific, unambiguous question with the required answer.
{format_instructions}

Figure 17: QA probing benchmark: generation prompt.

You are validating a yes/no Q&A entry for an AI tool probing benchmark.
Tool description:
{description}
Generated entry:
Question : {question}
Answer : {answer}
Check ALL of the following:
1. The question is specific to THIS tool — not generically answerable for any API.
2. The answer ("{answer}") is directly and unambiguously supported by the description.
3. The question uses "this tool" as a placeholder — the actual tool name does not appear.
4. The question tests a verifiable property (domain, capability, input/output type, format, etc.).
Set accept=true only if ALL four checks pass. Otherwise set accept=false and state which check failed.
{format_instructions}

Figure 18: QA probing benchmark: judge prompt (temperature = 0.0).

You will be provided with the description of a tool along with line-separated multiple choice options for the tool tokens. Your task is to select the correct tool token that corresponds to the given tool description. Please respond with only the correct tool token without any additional text. The tool description and options are provided below:

[START OF TOOL DESCRIPTION]
{tool_description}
[END OF TOOL DESCRIPTION]
[START OF OPTIONS]
{tool_options}
[END OF OPTIONS]

From the above new line-separated options, select the correct tool. Only respond with the tool representation (sequence of tokens that represent tool) and nothing else.

Figure 15: MCTS discriminative prompt (TG-3FM, TG-5FM).

You will be provided with the user query. Your task is to predict the tool that can best fulfill the user's request. You only need to predict the tool token. Please don't include any additional text in your response. The user query is provided below:

{query}

Figure 16: Stage 2 retrieval training and inference prompt.

You are building a multiple-choice probing benchmark for AI tool retrieval research.
Below is the description of an API or software tool:
{description}
Your task: generate one specific factual question about this tool's properties and provide one correct short answer plus three wrong-but-plausible alternatives.
Rules:
1. Use "this tool" in the question — never include the actual tool name or service name.
(The model will see only a virtual token at inference time, so the name must not be a hint.)
2. The question must be specific to THIS tool — not answerable for a generic API.
Good topics: primary output type, domain/industry, key input, core capability, supported format.
Bad question: "Does this tool provide an API?" (too generic).
3. Each answer option must be a short phrase (2–8 words), not a full sentence.
4. The correct_answer must be directly supported by the description above.
5. The three wrong answers must be plausible alternatives — the kind of answer a user might
expect from a tool in the same domain — but clearly incorrect for THIS specific tool.
6. All four options (correct + wrong) must be meaningfully distinct from each other.
7. Set skip=true if you cannot form a specific, unambiguous factual question from this description.
{format_instructions}

Figure 19: MCQ probing benchmark: generation prompt.

You are validating a multiple-choice question entry for an AI tool probing benchmark.
Tool description:
{description}
Generated entry:
Question : {question}
Correct answer: {correct_answer}
Wrong answers : {wrong_answers}
Check ALL of the following:
1. The question is specific to THIS tool — not generically answerable for any API.
2. The correct answer is directly and unambiguously supported by the description.
3. Each of the three wrong answers is plausible for a tool in the same domain but clearly incorrect for THIS tool.
4. All four options are meaningfully distinct from each other.
5. The question uses "this tool" as a placeholder — the actual tool name does not appear.
Set accept=true only if ALL five checks pass. Otherwise set accept=false and state which check failed.
{format_instructions}

Figure 20: MCQ probing benchmark: judge prompt (temperature = 0.0).

You are a data generation expert creating an evaluation benchmark for a tool retrieval model.
Your task is to generate {{n_samples}} evaluation samples, each consisting of a concise enterprise user query that points to EXACTLY ONE specific tool.
[START OF TASK DESCRIPTION]
The goal is to simulate how real enterprise users phrase requests — NOT how developers write API calls.
Unlike typical benchmark queries (which are overly specific and technical), your queries should be:
- Concise: 1–2 sentences, ideally under 20 words
- Written in natural business language (NOT using API names, technical identifiers, or schema terms)
- Realistic: something a business analyst, manager, or power user would actually type
Each sample must contain:
1. A query — concise, business-language question pointing to the target tool
2. An answer — a list containing exactly ONE tool name (the target tool)
[END OF TASK DESCRIPTION]
[START OF GENERATION RULES]
1. Queries must be CONCISE — avoid verbose, over-specified phrasing
2. Use BUSINESS LANGUAGE — never include API method names, OData syntax, or system-specific technical identifiers in the query
3. The query should naturally lead to the target tool without explicitly naming it
4. The query should NOT trivially match every other tool in the pool — it must be specific enough to distinguish the target
5. Do NOT use OData query syntax ($filter, $select, $expand, etc.) in queries
6. Do NOT hallucinate tool names — only use names exactly as they appear in the tool list below
7. Every query must be distinct — vary phrasing and angle of attack
GOOD query examples (concise, business language):
- "Show me all open purchase orders"
- "Which customers have overdue invoices?"
- "I need to track employee time-off requests"
- "List products that are low on stock"
BAD query examples (too technical or verbose):
- "Retrieve all PurchaseOrders where Status eq 'Open' using the GetPurchaseOrders endpoint"
- "I need to call the employee leave management API to fetch pending time-off approval requests for the current fiscal quarter"
[END OF GENERATION RULES]
[START OF TARGET TOOL]
Tool Name: {{anchor_tool_name}}
Description: {{anchor_tool_description}}
[END OF TARGET TOOL]
[START OF TOOL LIST]
These are all available tools. Your answer must contain exactly one tool name from this list:
{{candidate_pool_str}}
[END OF TOOL LIST]
{% if previous_feedback and previous_feedback != "None" %}
[START OF FEEDBACK FROM PREVIOUS ATTEMPT]
{{previous_feedback}}
Address these issues in your new generation.
[END OF FEEDBACK FROM PREVIOUS ATTEMPT]
{% endif %}

[START OF OUTPUT FORMAT]
{{format_instructions}}
[END OF OUTPUT FORMAT]
Generate {{n_samples}} easy-tier eval samples. Each query must be concise (under 20 words), in business language, and point unambiguously to exactly one tool.

Figure 21: RRB easy-tier generation prompt (Jinja2 template; `{{var}}` = template variable, `{% ...%}` = control block). The medium-tier prompt is identical except: queries target 2–3 tools with genuinely overlapping functional domains, length limit is 25 words, and the answer is a list of 2–3 tool names. The hard-tier prompt targets 4+ tools spanning a high-level business goal, in 1–3 sentences.

You are a quality validator for an enterprise tool retrieval evaluation benchmark. Given a batch of generated (query, answer) samples, assess whether each sample is suitable as an evaluation example.
[START OF TASK DESCRIPTION]
Validate each sample and return one verdict per sample (in the same order).
For each sample, determine:
1. Is the query CONCISE and written in BUSINESS LANGUAGE (not technical API language)?
2. Does the query sound like something a real enterprise user would actually ask?
3. For {{complexity}} tier: is the number of answer tools and the ambiguity level appropriate?
- easy: query should be specific enough that one tool is the clear answer; the phrasing should not be technical
- medium: query should be genuinely ambiguous between 2–3 tools; ambiguity must arise from overlapping domains, not vague phrasing
- hard: query should describe a high-level business goal that legitimately spans 4+ tools
4. Are all answer tools plausibly relevant to the query?
[END OF TASK DESCRIPTION]
[START OF VALIDATION RULES]
A sample PASSES (validation_result: true) if ALL of the following hold:
- The query is CONCISE (1–3 sentences, not verbose or over-specified)
- The query uses BUSINESS LANGUAGE — no API method names, technical identifiers, OData operators ($filter, $select, $expand, $orderby, etc.)
- The query reads naturally — it sounds like a real enterprise user asking a question, not a developer writing a specification
- All answer tools are plausibly relevant to the stated query
- The ambiguity level matches the complexity tier
A sample FAILS (validation_result: false) if ANY of the following hold:
- The query is verbose or reads like a technical specification
- The query contains API names, OData syntax, or schema property names
- The query is too generic to be meaningful (e.g. "get data", "manage system entities")
- Answer tools are not justified by the query content
- Wrong complexity (e.g. an easy query matching 3+ tools, or a hard query where only 1–2 tools fit)
[END OF VALIDATION RULES]
[START OF SAMPLES TO VALIDATE]
{{samples_str}}
[END OF SAMPLES TO VALIDATE]
[START OF OUTPUT FORMAT]
{{format_instructions}}
[END OF OUTPUT FORMAT]
Validate each sample in order and return one verdict per sample.

Figure 22: RRB LLM judge prompt, shared across all three tiers (temperature $= 0.0$). The `{{complexity}}` variable is instantiated as `easy`, `medium`, or `hard` per tier.

L Artifact Licenses and Intended Use

Datasets used. ToolBench (Qin et al., 2024b) provides tool documentation metadata crawled from RapidAPI's publicly accessible API catalog. The ToolBench repository and dataset are released under the Apache-2.0 License. Our use conforms to these terms: we do not redistribute the raw metadata, instead releasing the generated diagnostic benchmarks (RRB, MCQ, QA) derived from it.

Opensource instruction-tuned models. Gemma 3 (Team et al., 2025) is released by Google under the Gemma Terms of Use, which permits research and commercial use subject to the stated conditions. Qwen3.5 (Yang et al., 2025) is released by Alibaba under the Apache 2.0 License.

Artifacts released by this work. The ToolSense benchmark generation framework and the ToolBench diagnostic benchmarks (RRB, MCQ, QA) are released under the **Apache 2.0 License**. The intended use is research on parametric tool-retrieval systems, multi-stage fine-tuning evaluation, and related NLP benchmarking tasks. We do not recommend using the benchmarks to make production deployment decisions without independent validation on the target tool catalog.

M Dataset Documentation

PII and offensive content. The ToolBench tool metadata consists entirely of API endpoint names, descriptions, and parameter schemas scraped from RapidAPI's public-facing documentation. This data contains no personally identifiable information (PII), no sensitive categories (health, finance, biometric, etc.), and no offensive content. The RRB, MCQ, and QA benchmarks are fully synthetic; all items were generated by an LLM conditioned on the above API documentation, with no user data or personal information involved.

Dataset statistics. Table 12 summarises the benchmarks created in this work. The ToolBench tool catalog used for training and evaluation contains ∼47k tools spanning ∼16k API services.

Benchmark	Split	Easy	Medium	Hard
RRB	eval	167	167	166
MCQ	eval		496 items	
QA	eval		500 items	

Table 12: Benchmark item counts. The RRB evaluation set (500 items total) is stratified across three difficulty tiers. MCQ and QA are single-split probing benchmarks.

N Computational Resources and Software

Hardware. All training and evaluation experiments were run on a single node equipped with 8 NVIDIA H200 SXM5 (141 GB) GPUs. All model configurations used a single GPU.

Software versions. All training code uses the following package versions: Python 3.13, PyTorch 2.10.0, HuggingFace Transformers 5.5.0, PEFT 0.19.0, Accelerate 1.11.0, TRL 1.2.0, Datasets 4.7.0, and DeepSpeed 0.18.2. Benchmark generation used LiteLLM ≥1.60, LangGraph ≥0.2.0, ChromaDB ≥0.5.0, and Pydantic ≥2.0.0.

O Human Annotation Study

Task instructions. Three annotators independently validated a stratified random sample of 100 items per benchmark following the instructions below.

For **RRB**: annotators were shown each query and a pool of 14 candidate tools (the ground-truth tool(s) plus hard-negative neighbors from the generation pipeline). They were asked: "Select all tools from the candidate list that you believe are a correct and relevant match for this query's intent." Annotators were instructed to rely solely on the tool names and descriptions provided and not to use external search.

For **MCQ**: annotators were shown the full tool description and the four answer options and asked to select the single correct option (A/B/C/D) based on the description.

For **QA**: annotators were shown the full tool description and the yes/no question and asked to answer Yes or No based solely on the description.

Recruitment and compensation. Annotators were full-time employees of the authoring organization; no additional compensation beyond their regular employment was provided.

Consent and data protection. Annotators were informed of the study's purpose before participating and provided voluntary consent to have their annotations used for research. No personal data belonging to the annotators or any third parties was collected or stored.

Ethical review. This annotation study involves expert evaluation of AI-generated benchmark items and does not constitute human subjects research involving personal data collection.

Annotator demographics. All three annotators are native or proficient English speakers with deep professional expertise in enterprise-scale tool catalogs and API ecosystems.

P Use of AI Assistants

Claude Code with claude-4.5-opus (Anthropic) was used to assist with prototyping parts of the experimental codebase and with refining the writing and presentation of this paper.