

# Retrieval-Governed Context: Scope-Gated Selection of Instructions and Tools for LLMs and Intelligent Agents

SCOTT N. HWANG, The Pennsylvania State University College of Medicine, USA

Dynamically assembling system prompts (selecting instructions, tools, and safety policies at runtime) is increasingly common in production LLM systems [1, 11]. We present *retrieval-governed context*, an architecture that attaches structured governance metadata (scope gates, priority weights, conflict resolution, and mandatory injection) directly to instructions and tools as first-class schema fields, then layers this governance on top of standard retrieval backends. Instructions and tools are unified in a single typed corpus, enabling a shared pipeline for scope-gated retrieval, composition, and safety enforcement. We position BEAR as a systematic substrate for context engineering, comparable in behavioral output to careful hand-authored prompt engineering, but organized around scalable authoring, governance, and unified tool selection.

We evaluate on three corpora of increasing metadata richness. On the ToolBench benchmark (3,225 APIs from the benchmark evaluation split, 47 categories), governance significantly improves every retrieval backend tested ( $p < 0.0001$ ), with effect sizes inversely related to embedding quality (Cohen's  $d = 0.50$  for BM25 down to  $d = 0.21$  for the strongest dense model). On MetaTool (199 tools without category metadata), governance has negligible effect on recall ( $d < 0.08$ , n.s.), confirming that the benefit scales with metadata richness. When an LLM generates category tags from tool descriptions in a single offline pass (MetaTool+Tags), governance improves Recall@5 by 18–30 percentage points across backends (all  $p < 0.0001$ ). When tags are instead inferred from query text alone at runtime, governance hurts retrieval ( $d = -0.20$  to  $-0.59$ ), demonstrating that tag alignment between corpus and query is required. On a purpose-built 58-instruction behavioral corpus with rich multi-tag metadata, governance is the dominant factor. Removing scope gates degrades F1 by 33%, and retrieval-governed prompting matches conditional prompt assembly on anticipated contexts while achieving 0.917–1.000 recall on semantically discoverable instructions (vs. 0.000; McNemar  $p < 0.003$ ), with 90% token savings on tool retrieval and zero cross-domain tool leakage.

Our contribution is not a new retrieval algorithm but a backend-agnostic governance layer that makes context engineering systematic, with evidence that governance metadata quality and embedding quality are orthogonal contributors to retrieval-based context engineering.

CCS Concepts: • **Computing methodologies** → **Intelligent agents**; • **Information systems** → *Document filtering*; Retrieval effectiveness.

Additional Key Words and Phrases: large language models, intelligent agents, tool use, context engineering, retrieval, governance, safety, prompt construction

## ACM Reference Format:

Scott N. Hwang. 2026. Retrieval-Governed Context: Scope-Gated Selection of Instructions and Tools for LLMs and Intelligent Agents. *ACM Trans. Intell. Syst. Technol.* 1, 1 (April 2026), 22 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

---

Author's Contact Information: Scott N. Hwang, snh5391@psu.edu, The Pennsylvania State University College of Medicine, Radiology, Hershey, Pennsylvania, USA.

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM 2157-6912/2026/4-ART

<https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

## 1 Introduction

As LLM-controlled agents move from prototypes to production, the context that governs their behavior grows in size and complexity. For example, a mid-sized customer-support operation might maintain dozens of behavioral instructions across several departments. A large enterprise with hundreds of agents can easily accumulate hundreds of instructions and tens of thousands of tokens of context. Practitioners typically manage this complexity through manual prompt or context engineering by concatenating all instructions into a monolithic system message, wiring conditional lookup tables that map known contexts to instruction subsets, or relying on short role descriptions.

Each approach has well-understood failure modes. Monolithic prompts waste tokens and dilute relevant guidance in noise. Conditional prompt assembly (CPA), a standard production pattern [1, 11], is precise on anticipated contexts but requires manual updates as contexts evolve. Adding a new department requires modifying code, and instructions discoverable only by semantic similarity are invisible. Role prompting is token-efficient but behaviorally imprecise [24], offering little control over which specific behaviors and tools are available in a given situation.

Tool use compounds these problems [16, 22]. As the number of available tools grows, injecting all tool schemas into every prompt wastes context tokens and can degrade model performance [6], while selecting the right tools for a given situation requires either manual wiring or a retrieval mechanism. Existing retrieval-based approaches to tools [12, 23] typically maintain tools in a separate registry from behavioral instructions. Scope and safety are encoded informally in prose and routing logic rather than as queryable metadata. Instruction-Tool Retrieval (ITR) [5] applies retrieval to instruction and tool selection, reporting up to 95% token reduction and improved tool routing accuracy, validating retrieval-based instruction selection as a viable pattern.

We instead treat scope and governance as first-class entities in a context-engineering schema. Building on retrieval-based instruction selection [5], retrieval-governed context layers a governance pipeline over standard retrieval backends (dense, sparse, or hybrid) and unifies tools and behavioral instructions under a single typed schema. We implement this architecture in BEAR (Behavioral Evolution and Retrieval; <https://github.com/snhwang/bear>), which provides a broader behavioral substrate for LLM agents. The same governed corpus architecture supports behavioral evolution, inheritance, and multi-agent knowledge flow (manuscripts in preparation). This paper evaluates only BEAR's retrieval and governance components. In BEAR, tools are represented as a subtype of behavioral instruction. They share the same schema and governance pipeline as text instructions but add an action schema and are subject to hard scope gating rather than soft semantic retrieval. Scope filtering with `required_tags` prevents cross-domain tool leakage, priority weighting ranks competing instructions, conflict resolution handles mutually exclusive behaviors, and mandatory injection guarantees that safety and compliance directives are always present, independent of the retrieval backend.

We evaluate this architecture against four common context-construction alternatives (monolithic injection, role prompting, conditional prompt assembly (CPA), and random- $k$  retrieval) on a purpose-built multi-tag behavioral corpus (Pet Simulation), a procedurally generated multi-department customer-support corpus, and two external benchmarks (ToolBench [15] and MetaTool [7]). We measure retrieval quality on anticipated, novel, and semantically discoverable instructions, token efficiency, and cross-domain tool leakage. Our empirical findings are summarized in Section 6.

## 2 Governed Context Architecture

### 2.1 Context as a Governed Corpus

We treat an agent's "context" as the full control surface presented to the LLM, including system-level behavioral instructions, tool schemas, safety and compliance policies, and their associated metadata,

```

99  # Behavioral directive
100 - id: greeting-empathy
101   type: protocol
102   priority: 80
103   tags: [customer-service]
104   required_tags: [upset_customer]
105   text: >
106     Acknowledge the customer's frustration
107     before addressing the issue...
108
109 # Tool instruction
110 - id: process-refund
111   type: tool
112   priority: 70
113   tags: [customer-service, billing]
114   required_tags: [billing, refund_requested]
115   actions:
116     - name: process_refund
117       parameters:
118         order_id: {type: string, required: true}
119         amount: {type: number, required: true}
120   text: >
121     Process a refund for a customer order.

```

Fig. 1. Two YAML instructions from a customer-service corpus. Both share the same schema; the tool adds an actions field. For tools, required\_tags are a hard gate: a tool surfaces only when all prerequisite context tags are present. For text instructions, satisfying required\_tags guarantees inclusion, but high similarity may also surface them through the soft-retrieval path.

not just a single prompt string. Context engineering in this setting is the problem of deciding which pieces of this corpus should be visible to the model for a given situation, and under what constraints (scope, prerequisites, mandatory inclusion, conflicts). Our BEAR architecture instantiates this as a governed retrieval pipeline over a unified instruction/tool corpus.

Each behavioral element is a typed instruction in a shared corpus (YAML in our implementation), specifying what it does and when it applies. Figure 1 shows examples of a behavioral directive and a tool.

All instructions share a common typed schema:

- **Type:** one of {persona, constraint, protocol, directive, tool}.
- **Scope:** tags (soft), required\_tags (hard), user roles, domains, and trigger patterns.
- **Priority:** integer [0, 100] used in scoring and conflict resolution.
- **Relationships:** fields for linking instructions to each other: conflicts\_with (mutually exclusive instructions), requires (dependencies), and supersedes (updated instructions that replace older ones). In this work we primarily exercise conflicts\_with; requires and supersedes are included for completeness but not deeply evaluated.
- **Actions** (tool-only): function name and parameter schema (types and required flags).

Table 1 summarizes the instruction schema and groups fields by their role in governance and retrieval.

The key design choice is that tools are first-class instructions in the same corpus and schema, differing only by having an actions field and stricter scope gating. This unification allows tools

Table 1. Instruction schema overview. Fields are grouped by their role in governance and retrieval.

Category	Field	Purpose
Type	type	One of {persona, constraint, protocol, directive, tool}. Tools share all other fields but add actions and stricter scope gating.
Scope	tags	Soft labels (domains, departments, user states) used for grouping and optional filtering.
	required_tags	Hard conjunctive gates. All must be present in the context for an instruction to be eligible. For tools this is an absolute requirement. For text instructions, it guarantees inclusion and can be supplemented by semantic similarity.
Priority	priority	Integer [0, 100] used to rank candidates and break ties in conflicts; combined with similarity in the scoring function.
Relationships	conflicts_with	List of instruction IDs that cannot co-occur; lower-priority items are removed at retrieval time.
	requires	Optional dependencies on other instructions (e.g., a policy that requires a base constraint). Included for completeness. Not deeply evaluated here.
	supersedes	Optional override relationships (e.g., updated policy superseding an older one). Included for completeness. Not deeply evaluated here.
Actions	actions (tool-only)	Function name and parameter schema (types and required flags) for tools; emitted as JSON tool definitions in the final context.

and behavioral directives to share a single governance pipeline, instead of being managed by separate, ad-hoc wiring or registries. Our implementation stores instructions as YAML files for ease of authoring, inspection, and version control during development, but the governance layer is format-agnostic. Any backend that exposes the same typed schema (e.g., JSON documents, a database, or a configuration service) can serve as the instruction corpus. In future work we plan to add a GUI for creating and editing governed instructions. Such a UI would operate over the schema, with YAML relegated to an optional import/export format rather than a requirement.

*Agent executor.* BEAR is responsible for context composition. It determines which instructions are present, in what order of priority, and under what scope constraints. What the agent does with the composed context is the agent's concern. This separation allows BEAR to compose with any executor, including but not limited to an LLM that interprets instructions semantically, a marker parser that executes them literally, or a rule-based system that maps them to code paths. It also means that end-to-end behavioral outcomes depend on the executor as much as on the governance layer. LLMs in particular may not follow instructions precisely, may interpret ambiguous directives differently across model versions, or may produce inconsistent outputs at higher temperatures. BEAR's contribution is upstream of these behaviors. It ensures the right instructions are present and the wrong ones are excluded.

*Action markers.* A complementary lightweight mechanism is available for frequent, low-stakes actions: *action markers* embedded directly in text instruction content (e.g., `[!speed(sprint)]`, `[!animation(excited_bounce)]`, `[!mood(excited)]`). Unlike tools, markers require no separate actions field schema and fire directly from retrieved instruction text. The instruction is simultaneously a human-readable behavioral description and an executable action specification. Markers are subject only to the soft retrieval path and carry no conjunctive scope guarantee, making them appropriate for simulation-speed behavioral dispatch where actions are reversible and the cost of an errant instruction is low. They are relatively risky for actions with external consequences (financial transactions, API calls, data mutations), where the hard gate provides a higher level of safety. The BEAR Pet Simulation (<https://github.com/snhwang/bear>) demonstrates action markers in a running implementation. Pets respond to stimuli, players, and moods through marker-driven animation, movement, and state transitions, with the LLM contributing contextual enrichment rather than load-bearing action dispatch. Action markers are not evaluated in a separate controlled experiment. The Pet Simulation serves as a qualitative demonstration of the mechanism.

## 2.2 Governed Retrieval Pipeline

Given a query and a set of context tags (e.g., department, user state, safety mode), BEAR constructs the agent's context in four governance stages (with a final composition stage), as illustrated in Figure 2.

We briefly describe each stage, emphasizing where BEAR's governance differs from typical retrieval or CPA pipelines.

1. *Scope filtering with required\_tags.* Each instruction may declare a set of `required_tags`, a per-instruction conjunctive scope condition. For tools, this is a hard gate: a tool surfaces only when all required tags are present in the context, regardless of embedding similarity. For text instructions (persona, constraint, protocol, directive), satisfying `required_tags` guarantees inclusion; in addition, an instruction can be admitted via semantic similarity if its embedding similarity to the query exceeds a threshold  $\theta$  (default 0.3), even when tags do not match.

Thus, scope gating is type-aware. Tools must satisfy hard conjunctive scope, while text instructions can be discovered semantically when the author's tags are too coarse. This type-differentiated gating is an architectural safety decision, not merely a retrieval optimization. Retrieving an extra behavioral guideline is low-risk (the model may ignore it), while retrieving an out-of-scope tool can trigger irreversible real-world actions (API calls, database mutations). Hard tool gating, combined with mandatory safety injection, ensures that the governance layer provides a deterministic safety floor independent of embedding quality or query phrasing.

2. *Priority-weighted scoring.* Candidates that pass scope filtering are scored by

$$\text{score} = (1 - \alpha) \text{sim}(q, d) + \alpha \frac{\text{priority}}{100}$$

where  $\text{sim}(q, d)$  is cosine similarity between the query embedding and the instruction embedding, and  $\alpha$  (default 0.3) controls the priority-similarity blend. This lets authors encode governance preferences (e.g., emphasize safety or department-specific policies) directly in metadata, independent of the retrieval backend.

3. *Conflict resolution.* Instructions can declare `conflicts_with` edges to other instructions. When two retrieved instructions conflict, the lower-priority item is removed. This applies to tools (e.g., mutually exclusive actions) and text directives (e.g., conservative vs. experimental policies), giving a declarative mechanism for resolving incompatible behaviors at retrieval time.

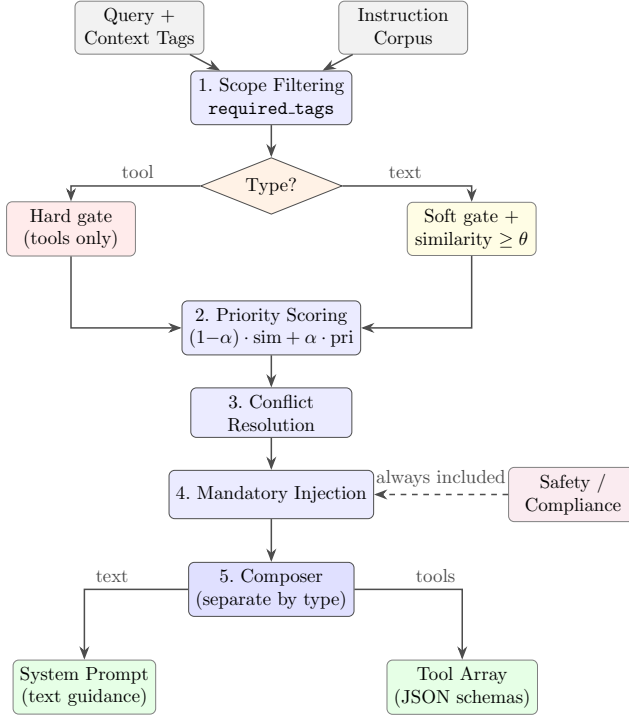


Fig. 2. Retrieval-governed context pipeline. Context tags and a query select candidates from the instruction corpus via scope filtering. Tools require all `required_tags` (hard gate); text instructions may also surface via similarity (soft gate). Candidates are scored, conflicts resolved, and mandatory safety/compliance items are always included. The Composer then separates results by type: text instructions become system-prompt guidance, tool instructions become structured JSON schemas.

**4. Mandatory injection.** Separately from the per-instruction scope, the system defines a set of global mandatory tags (e.g., `mandatory_tags: [safety]`). Any instruction carrying a mandatory tag is injected into every context, bypassing similarity scoring and scope filtering entirely. This guarantees that safety and compliance directives are present regardless of query, tags, or retrieval backend. Unlike `required_tags`, which control when an instruction may appear, mandatory tags control which instructions must always appear.

### 2.3 Context Composition and Emission

After governance, BEAR emits a structured context in two channels:

- **Text guidance:** persona, constraints, protocols, and directives are concatenated into a guidance string used as the system message.
- **Tool array:** tool instructions are emitted as JSON schemas in the underlying API's tool-calling format (e.g., OpenAI function calls).

This separation enforces a clean interface. Tools never appear in the guidance text, and behavioral text never appears in the tool array. From the model's perspective, the "prompt" it receives is a governed context, a curated system message plus a curated set of tools, both derived from the same instruction corpus and governance metadata. Because governance is factored from retrieval,

the same pipeline can sit atop dense-embedding retrievers, sparse keyword search, hybrid RAG systems, or graph/knowledge-graph retrieval. In our experiments, we instantiate it with four dense models, BM25, and a simple hash-based backend to show that most gains come from governance rather than from the choice of retriever.

### 3 Evaluation

We evaluate retrieval-governed context along four axes: (1) retrieval quality on an application-like behavioral corpus; (2) tool retrieval precision as the tool corpus scales; (3) comparison with CPA on anticipated, novel, and semantically discoverable instructions; and (4) a live demonstration in the BEAR Pet Simulation showing context-sensitive behavior through action markers (<https://github.com/snhwang/bear>).

*Baselines.* We compare against four context-construction strategies: monolithic injection (all instructions in every prompt), role prompting (a short natural-language persona description) [24], conditional prompt assembly (CPA, hand-written lookup tables mapping context keys to instruction subsets) [1, 11], and random- $k$  selection ( $k$  instructions sampled uniformly, serving as a lower bound). Unless otherwise noted, experiments use the BAAI/bge-base-en-v1.5 [21] sentence-transformer (hereafter *BGE*, 768-dim) for dense embeddings. Section 3.6 defines all embedding model abbreviations and evaluates additional backends. All retrieval metrics are fully deterministic (no LLM-as-judge), and 95% confidence intervals are bootstrap CIs (10,000 iterations).

*Retrieval metrics.* For each query, we have a ground-truth set of expected instructions (typically 4–6 out of 58), and the retriever returns the top  $k$  instructions. We report **Precision@ $k$**  (fraction of retrieved instructions that are expected), **Recall@ $k$**  (fraction of expected instructions that are retrieved), and **F1@ $k$**  (their harmonic mean). Unless otherwise noted,  $k=10$ . The larger procedural corpus uses  $k=25$ . We report strict metrics against tag-determined ground truth and relaxed metrics that expand the expected set with semantically plausible instructions. Relaxed sets were created once by the author before running any retrieval experiments. Strict metrics are our primary results. The relaxed metrics provide an upper bound.

#### 3.1 External Benchmark Validation

We begin by validating on two established external benchmarks before presenting detailed experiments on purpose-built corpora in subsequent sections. A key concern for any governance-aware retrieval system is whether its benefits transfer to independently authored tool collections. We evaluate BEAR on ToolBench [15] and MetaTool [7], which differ sharply in the availability of structured category metadata, providing both a positive validation and a negative control. We test five governed backends: BGE-M3 (BAAI/bge-m3, 1024-dim), Qwen3-0.6B (Qwen/Qwen3-Embedding-0.6B, 1024-dim), Qwen3-4B (Qwen/Qwen3-Embedding-4B, 2560-dim), BM25 (sparse lexical), and BGE; each is also evaluated without governance.

*ToolBench.* ToolBench [15] (ICLR 2024 Spotlight) provides 16,464 real-world REST APIs from RapidAPI across 49 categories. We use the publicly available benchmark evaluation split, which contains 3,225 APIs spanning 47 categories (e.g., “Data,” “Business,” “Communication”). We use 1,100 benchmark queries spanning six difficulty levels (g1\_instruction, g1\_category, g1\_tool, g2\_instruction, g2\_category, g3\_instruction). RapidAPI categories are mapped directly to required\_tags in BEAR’s governance layer, requiring no manual curation beyond the category labels already present in the benchmark. Because ToolBench was authored independently of this work, it provides external validation that is free of the single-author circularity present in the Pet Simulation corpus.

*MetaTool*. MetaTool [7] (ICLR 2024) provides 199 tools from the OpenAI plugin store with ground-truth tool assignments. We sample 500 queries for evaluation. Unlike ToolBench, MetaTool tools lack a meaningful category hierarchy. Tags are derived from tool names only, providing no useful scope metadata for governance. MetaTool therefore serves as a negative control. If governance adds value only when metadata is informative, it should have no effect here.

*Results.* On ToolBench (Table 2), governance yields significant improvements for every backend (all  $p < 0.0001$ ). BEAR+BGE achieves Recall@5 of 0.679, an 18% relative improvement over ungoverned BGE (0.574;  $t=13.73$ ,  $d=0.41$ ). BEAR+BGE-M3 achieves 0.683, a 13% improvement over ungoverned BGE-M3 (0.602;  $t=12.12$ ,  $d=0.37$ ). BEAR+BM25 achieves 0.583, a 25% improvement over ungoverned BM25 (0.468;  $t=16.49$ ,  $d=0.50$ ) and the largest governance effect of any backend. Instruction-tuned embeddings lead: Qwen3-0.6B (0.740) and Qwen3-4B (0.734) significantly outperform general-purpose BGE (0.679;  $p < 0.0001$ ,  $d=0.21$  and  $d=0.17$  respectively), while BGE and BGE-M3 are indistinguishable under governance ( $p=0.61$ , n.s.). Notably, BM25 with governance (0.583) slightly exceeds BGE without governance (0.574), though the difference is not significant ( $p=0.40$ , n.s.), showing that governance can partially compensate for a weaker backend.

The ungoverned Qwen3 baselines reveal an inverse relationship between embedding quality and governance benefit: BM25 gains +25% ( $d=0.50$ ), BGE +18% ( $d=0.41$ ), BGE-M3 +13% ( $d=0.37$ ), Qwen3-4B +7% ( $d=0.24$ ), and Qwen3-0.6B +6% ( $d=0.21$ ). All differences are significant ( $p < 0.0001$ ), and the pattern is generally inverse. Stronger embedding models benefit less from governance, though the two Qwen3 variants show similar effect sizes. This suggests governance and embedding quality are partially substitutable, with better embeddings already capturing some of what governance metadata provides. Notably, ungoverned Qwen3-0.6B (0.696) and governed BGE-M3 (0.683) have overlapping confidence intervals, suggesting that on corpora with coarse metadata, the choice of embedding model can matter as much as adding governance.

On MetaTool (Table 3), governance has negligible effect at the recall level ( $d < 0.08$ , all  $p > 0.10$ ), though NDCG shows a small but significant effect for BGE-M3 ( $p=0.002$ ,  $d=0.14$ ). The practical magnitude is minimal, confirming MetaTool's role as an approximate negative control. Governance adds little when meaningful category metadata is absent. Table 4 shows that this changes sharply when metadata is supplied: LLM-generated tool tags improve governance by 18–30 percentage points, while runtime-inferred query tags actively harm it. Backend choice dominates entirely. Qwen3-4B achieves 0.912 Recall@5, far exceeding BGE's 0.710, a 28% gap driven entirely by embedding quality.

The contrast between the two benchmarks, together with the Pet Sim results, reveals a key insight: governance value scales with metadata richness. On Pet Sim (rich multi-tag metadata), governance is the dominant driver (up to 5× F1 improvement). On ToolBench (single category tag per API), governance helps significantly but moderately (+4–12% absolute,  $d=0.21$ –0.50). On MetaTool (no meaningful tags), governance has negligible effect on recall ( $d < 0.08$ , n.s.). Meanwhile, embedding quality matters more as metadata richness decreases. On ToolBench, instruction-tuned embeddings outperform general-purpose models by a significant margin. On MetaTool, backend choice is the sole determinant of performance. The practical recommendation generalizes. Invest in governance metadata first (biggest ROI), then choose the best embedding model for the domain. Both are orthogonal contributors.

*MetaTool with LLM-generated tags.* To test whether governance value can be bootstrapped onto an untagged corpus, we run two additional conditions on MetaTool. In **MetaTool+Tags**, an LLM generates required\_tags from tool descriptions in a single offline pass; query context is derived from the same tool tags at runtime. In **MetaTool+QueryTags**, tags are instead inferred from query text alone at runtime, without knowledge of the target tool. Table 4 shows the results.

Table 2. ToolBench results ( $k=5$ , 1,100 queries, 3,225 APIs from benchmark split, 47 categories, 95% bootstrap CIs). Governance significantly improves every backend (all  $p<0.0001$ , paired  $t$ -test + Wilcoxon, 1,100 queries). The  $\Delta$  column shows the governance effect: absolute Recall@5 improvement and Cohen's  $d$  ( $d=0.2$  small, 0.5 medium, 0.8 large [3]).

Backend	Recall@5 [95% CI]	NDCG@5 [95% CI]	$\Delta$ Recall ( $d$ , $p$ )
<i>With governance (required_tags = API category)</i>			
BEAR+Qwen3-0.6B	<b>0.740</b> [0.720, 0.759]	<b>0.713</b> [0.694, 0.732]	+0.044 ( $d=0.21$ , $p<0.0001$ )
BEAR+Qwen3-4B	0.734 [0.714, 0.753]	0.695 [0.675, 0.713]	+0.051 ( $d=0.24$ , $p<0.0001$ )
BEAR+BGE-M3	0.683 [0.661, 0.704]	0.648 [0.627, 0.668]	+0.081 ( $d=0.37$ , $p<0.0001$ )
BEAR+BGE	0.679 [0.658, 0.699]	0.644 [0.624, 0.663]	+0.105 ( $d=0.41$ , $p<0.0001$ )
BEAR+BM25	0.583 [0.561, 0.605]	0.564 [0.543, 0.586]	+0.115 ( $d=0.50$ , $p<0.0001$ )
<i>Without governance (embedding similarity only)</i>			
Qwen3-0.6B	0.696 [0.675, 0.717]	0.663 [0.643, 0.682]	—
Qwen3-4B	0.683 [0.663, 0.704]	0.640 [0.619, 0.659]	—
BGE-M3	0.602 [0.579, 0.624]	0.572 [0.550, 0.593]	—
BGE	0.574 [0.552, 0.596]	0.545 [0.524, 0.566]	—
BM25	0.468 [0.445, 0.491]	0.447 [0.425, 0.469]	—

Table 3. MetaTool results ( $k=5$ , 500 queries, 199 tools, no categories, 95% bootstrap CIs). Without meaningful category metadata, governance has negligible effect on recall ( $d<0.08$ , n.s.); NDCG shows a small significant effect for BGE-M3 ( $p=0.002$ ,  $d=0.14$ ). Backend quality independently determines retrieval performance.

Condition	Recall@5 [95% CI]	NDCG@5 [95% CI]
<i>Governed backends</i>		
BEAR+Qwen3-4B (gov)	<b>0.912</b> [0.886, 0.935]	<b>0.837</b> [0.809, 0.863]
BEAR+Qwen3-0.6B (gov)	0.885 [0.857, 0.911]	0.792 [0.762, 0.820]
BEAR+BGE-M3 (gov)	0.743 [0.704, 0.781]	0.635 [0.598, 0.671]
BEAR+BGE (gov)	0.710 [0.669, 0.749]	0.596 [0.558, 0.633]
BEAR+BM25 (gov)	0.406 [0.364, 0.449]	0.346 [0.308, 0.385]
<i>Ungoverned baselines</i>		
Qwen3-4B (no gov)	0.916 [0.891, 0.939]	0.839 [0.812, 0.864]
Qwen3-0.6B (no gov)	0.884 [0.856, 0.910]	0.792 [0.762, 0.820]
BGE-M3 (no gov)	0.734 [0.694, 0.772]	0.621 [0.584, 0.657]
BGE (no gov)	0.718 [0.678, 0.757]	0.603 [0.566, 0.640]
BM25 (no gov)	0.406 [0.364, 0.449]	0.350 [0.311, 0.389]
<b>MetaTool:</b> Recall governed vs. ungoverned all $d<0.08$ , n.s.; NDCG BGE-M3 $p=0.002$ , $d=0.14$		

MetaTool+Tags is evaluated on the  $n=10,051$  queries whose target tool received LLM-generated tags; MetaTool+QueryTags uses all  $n=21,111$  queries.

*End-to-end tool selection.* To verify that retrieval-level improvements translate to downstream task performance, we run the full ToolBench pipeline end-to-end. For each query, BEAR retrieves the top- $k$  tool schemas, composes them into a system prompt, and a local LLM (mistralai/Mistral-Nemo-Instruct-2407, 12B parameters, served via vLLM) selects which tool to call. We measure tool accuracy (correct tool name selected) and exact accuracy (correct tool name and argument values) across all 1,100 queries. Table 5 shows that governance improves tool selection for every

Table 4. MetaTool with LLM-generated tags ( $k=5$ , 199 tools, 95% bootstrap CIs). **MetaTool+Tags**: required\_tags generated offline from tool descriptions; governance improves all backends significantly. **MetaTool+QueryTags**: tags generated from query text at runtime; governance hurts all semantic backends.

Condition	Backend	Recall@5 [95% CI]	NDCG@5 [95% CI]
<i>MetaTool+Tags (LLM tool tags, <math>n=10,051</math> queries with tagged tools)</i>			
	BEAR+Qwen3-4B (gov)	<b>0.915</b> [0.907, 0.923]	<b>0.840</b> [0.831, 0.849]
	BEAR+BGE (gov)	0.861 [0.851, 0.871]	0.756 [0.745, 0.767]
	BEAR+BM25 (gov)	0.603 [0.589, 0.617]	0.519 [0.505, 0.532]
	BEAR+Hash (gov)	0.370 [0.355, 0.384]	0.280 [0.267, 0.293]
	BGE (no gov)	0.678 [0.666, 0.690]	0.567 [0.555, 0.579]
<i>MetaTool+QueryTags (LLM query tags at runtime, <math>n=21,111</math>)</i>			
	BEAR+Qwen3-4B (gov)	0.615 [0.608, 0.621]	0.507 [0.501, 0.513]
	BEAR+BGE (gov)	0.516 [0.510, 0.523]	0.411 [0.405, 0.416]
	BEAR+BM25 (gov)	0.340 [0.334, 0.346]	0.296 [0.290, 0.302]
	BEAR+Hash (gov)	0.180 [0.175, 0.185]	0.117 [0.113, 0.121]
	Qwen3-4B (no gov)	0.905 [0.901, 0.909]	0.819 [0.815, 0.823]
	BGE (no gov)	0.678 [0.671, 0.684]	0.561 [0.556, 0.567]
<b>+Tags</b> : all gov vs. no-gov $p<0.0001$ , $d=0.48-1.17$ ; <b>+QueryTags</b> : gov hurts semantic backends ( $d=-0.20$ to $-0.59$ , $p<0.0001$ )			

Table 5. End-to-end tool selection accuracy on ToolBench (1,100 queries, mistralai/Mistral-Nemo-Instruct-2407 12B via vLLM,  $k=5$ , 95% bootstrap CIs). All governed vs. ungoverned pairs are significant ( $p<0.0001$ , paired  $t$ -test).

Backend	Tool Acc [95% CI]	Exact Acc [95% CI]	$\Delta$ Tool ( $d$ )
<i>With governance</i>			
BEAR+Qwen3-0.6B	<b>0.785</b> [0.760, 0.808]	<b>0.707</b> [0.680, 0.734]	+0.080 ( $d=0.21$ )
BEAR+Qwen3-4B	0.768 [0.744, 0.792]	0.681 [0.654, 0.708]	+0.064 ( $d=0.17$ )
BEAR+BGE	0.761 [0.735, 0.785]	0.683 [0.655, 0.710]	+0.119 ( $d=0.30$ )
BEAR+BGE-M3	0.750 [0.724, 0.775]	0.689 [0.662, 0.716]	+0.102 ( $d=0.27$ )
BEAR+BM25	0.675 [0.647, 0.703]	0.607 [0.579, 0.636]	+0.140 ( $d=0.33$ )
<i>Without governance</i>			
Qwen3-0.6B	0.705 [0.678, 0.731]	0.646 [0.618, 0.675]	—
Qwen3-4B	0.704 [0.676, 0.731]	0.616 [0.587, 0.645]	—
BGE-M3	0.648 [0.619, 0.675]	0.591 [0.561, 0.619]	—
BGE	0.642 [0.614, 0.670]	0.582 [0.553, 0.610]	—
BM25	0.535 [0.505, 0.565]	0.485 [0.456, 0.515]	—
Monolithic	0.023 [0.015, 0.032]	0.021 [0.013, 0.029]	—

backend (all  $p<0.0001$ , paired  $t$ -test,  $n=1,100$ ). The inverse pattern from retrieval-level evaluation persists: BM25 benefits most (+0.140,  $d=0.33$ ) and Qwen3-0.6B least (+0.080,  $d=0.21$ ). Monolithic injection (all 3,225 tool schemas) is effectively unusable: 0.023 accuracy with 363/1,100 LLM errors, 34 $\times$  lower than the best governed system (0.785). At production corpus scale, retrieval is not an optimization but a requirement for functional tool selection. Note that absolute accuracy figures are LLM-dependent. A different model would produce different numbers. BEAR's contribution is upstream of the LLM's selection decision and holds regardless of which model is used: it ensures the correct tool schema is present in context and cross-domain tools are excluded.

Table 6. Retrieval quality on Pet Simulation corpus (60 queries,  $k = 10$ , 95% bootstrap CIs). Hash = token-hash embeddings; Semantic = BAAI/bge-base-en-v1.5. Full BEAR vs. Pure Similarity: BGE  $t=26.15$ ,  $p<0.0001$ ,  $d=3.38$ ; Hash  $t=10.39$ ,  $p<0.0001$ ,  $d=1.34$ . Full BEAR vs. Priority-Heavy: both n.s. ( $p>0.19$ ).

Configuration	Hash		Semantic (BGE)	
	Strict F1	Relaxed F1	Strict F1	Relaxed F1
Full BEAR ( $\alpha = 0.3$ )	0.835 [0.800, 0.869]	0.886 [0.867, 0.904]	0.780 [0.756, 0.806]	0.866 [0.840, 0.892]
Pure Similarity ( $\alpha = 0$ )	0.481 [0.430, 0.532]	0.404 [0.353, 0.456]	0.155 [0.111, 0.202]	0.214 [0.166, 0.262]
Priority-Heavy ( $\alpha = 0.7$ )	0.837 [0.804, 0.870]	0.886 [0.867, 0.904]	0.757 [0.725, 0.789]	0.866 [0.840, 0.892]

### 3.2 Retrieval Quality

Having validated governance on independently authored benchmarks, we turn to more detailed experiments on the purpose-built Pet Simulation corpus, which supports richer query types and direct comparison against conditional prompt assembly. The corpus models a virtual pet simulation environment in which agents respond to stimuli, manage mood states, interact with players, and encounter other pets. We first evaluate retrieval quality on the Pet Simulation corpus, which consists of 58 YAML instructions across 8 files with 60 queries. This corpus was designed as a controlled precursor to a more complex Evolutionary Ecosystem simulation (BEAR; <https://github.com/snhwang/bear>), in which agents face dynamic environmental changes, predator threats, and emergent social contexts that make manual context wiring impractical. The pet simulation’s multi-domain structure was chosen deliberately to stress-test governance mechanisms across varied, overlapping input types in a tractable setting before applying them to the full ecosystem. The 58 instructions were authored by a single researcher as part of the simulation design. The 60 queries were manually constructed to cover the space of simulation scenarios across all eight instruction files. Ground truth is determined by tag matching: an instruction is relevant to a query if its tags overlap with the query’s context tags. We acknowledge that the same author designed both the corpus and the governance system, which introduces a potential circularity; the relaxed metric set was constructed to account for legitimate near-misses where a semantically appropriate instruction was not explicitly tagged for a given scenario. Because tag-determined ground truth penalizes some useful retrievals (e.g., retrieving an “excited” mood instruction for a ball stimulus that was not explicitly tagged), we report both strict and relaxed metrics as described above.

Table 6 reveals a striking finding: removing governance ( $\alpha=0$ ) collapses strict F1 from 0.780 to 0.155 for semantic embeddings (5× reduction) and from 0.835 to 0.481 for hash. In contrast, varying  $\alpha$  between 0.3 and 0.7 changes F1 by less than 0.03, showing the system is robust to the exact weighting once governance is engaged. Governance metadata is the dominant factor; the embedding model is secondary. The choice of backend changes absolute scores far less than enabling or disabling governance.

### 3.3 Tool Scaling

*Setup.* We construct a corpus of 50 tool instructions spanning 8 domains (food/survival, danger/combat, social, exploration, crafting, inventory, communication, environment). Each tool uses `required_tags` for hard scope gating. We test 8 scenarios (e.g., “predator attack”, “social encounter”, “hungry near food”) with ground-truth expected and not-expected tool sets, at corpus sizes 5, 10, 25, and 50 tools.

Table 7 shows that recall remains perfect (1.000) at all corpus sizes, indicating every expected tool is retrieved. Precision decreases slightly from 1.000 to 0.912 as the corpus grows. Additional

Table 7. Tool retrieval quality as corpus size scales from 5 to 50 tools across 8 domains ( $\alpha = 0.3$ , BAAI/bge-base-en-v1.5). Zero cross-domain leakage confirms `required_tags` prevents out-of-scope tools from surfacing.

Corpus Size	Precision	Recall	Cross-Domain Leaks	Status
5	1.000	1.000	0	PASS
10	1.000	1.000	0	PASS
25	0.900	1.000	0	PASS
50	0.912	1.000	0	PASS

Table 8. Prompt token savings from scoped tool retrieval vs. injecting all 50 tools (2,928 tokens baseline).

Scenario	Tools	Tokens	Savings
Predator attack	4	214	93%
Social encounter	7	450	85%
Hungry near food	3	172	94%
Building w/ materials	3	164	94%
Combat encounter	6	346	88%
Near water	3	155	95%
Item management	5	288	90%
Territory dispute	10	629	79%
<b>Average</b>			<b>90%</b>

scope-matched tools from related domains (e.g., greet and teach both matching a social context) are retrieved but are not false positives in the cross-domain sense. Critically, zero cross-domain leaks are observed at any scale, since `required_tags` provides hard isolation between domains.

*Token savings.* Scoped retrieval achieves substantial token reduction compared to injecting all 50 tool schemas (2,928 tokens). Average token savings reach 90% (Table 8), with savings ranging from 79% (territory dispute, the broadest context with 10 matching tools) to 95% (near water, 3 tools). This savings grows with corpus size: at 500 tools, static injection would consume ~29,000 tokens while scoped retrieval remains bounded by the number of scope-matched tools.

### 3.4 Comparison with Conditional Prompt Assembly

*Setup.* We compare governed retrieval against CPA, a deterministic lookup-table baseline that assembles prompts from pre-defined template slots keyed by context tags. CPA represents standard production practice: if/else blocks or tag-to-template registries [11, 13]. Both systems receive identical instruction content. The difference is how each selects instructions. We use a procedurally generated multi-department customer support corpus, scaling from 10 to 500 agents with 28–675 instructions.

*Standard queries.* On anticipated contexts with explicit tag mappings, both BEAR and CPA achieve perfect recall across all instruction categories and scale points ( $N = 10\text{--}500$ , 28–675 instructions,  $\alpha = 0.3$ ,  $\theta = 0.3$ ,  $k = 25$ ), confirming that governed retrieval matches CPA’s deterministic correctness without sacrificing accuracy.

Table 9. Token efficiency as agent population scales. Both BEAR and CPA maintain roughly constant per-query token counts, while static injection scales linearly with corpus size.

$N$	BEAR	CPA	Static	BEAR/Static
10	952	952	1,454	0.655
50	985	950	4,344	0.227
100	995	952	8,021	0.124
500	953	944	34,721	0.027

Table 10. Semantic recall: instructions discoverable only through embedding similarity ( $n=12$  queries per scale point, 95% CIs shown). By construction, CPA's lookup tables cannot reference these instructions. CPA upper CI is Clopper–Pearson exact bound for  $k=0$ . McNemar  $p=0.001$ – $0.003$ ; Cohen's  $h=2.56$ – $3.14$  (large).

$N$	$n$	BEAR [95% CI]	CPA [95% CI]	$\Delta$
10	12	1.000 [1.000, 1.000]	0.000 [0.000, 0.265]	+1.000
50	12	1.000 [1.000, 1.000]	0.000 [0.000, 0.265]	+1.000
100	12	0.917 [0.750, 1.000]	0.000 [0.000, 0.265]	+0.917
500	12	0.917 [0.750, 1.000]	0.000 [0.000, 0.265]	+0.917

### 3.5 System-Level Scaling and Maintenance

We next study how BEAR behaves as the number of tools and instructions grows in a multi-department customer-support corpus, comparing against static monolithic injection and CPA. We vary the corpus size  $N$  from 10 to 500 agents (28 to 675 instructions, up to 34,721 tokens) and measure per-query context size and cross-domain tool leakage.

Table 9 shows that BEAR and CPA achieve similar per-query token counts (~950–995 tokens) independent of corpus size, while the static baseline grows linearly to 34,721 tokens at  $N=500$  (a ~36 $\times$  compression ratio for BEAR). Retrieval latency remains in the 4.2–10.4 ms range (p95) even at  $N=500$ . CPA's maintenance cost grows with corpus size: each new department requires additional conditional wiring code, whereas BEAR requires no per-context wiring since new instructions and tools become available through their metadata without modifying code.

To test adaptability, we evaluate BEAR and CPA on (1) novel departments that CPA has no explicit lookup entry for and (2) instructions that are discoverable only via semantic similarity. On novel contexts, BEAR achieves 0.944–1.000 recall while CPA plateaus at 0.888–0.912 across all scale points; the differences are directional but do not reach statistical significance given the small per-condition sample sizes ( $n=19$ – $24$ , McNemar  $p>0.48$ ). The effect is structural: CPA cannot retrieve instructions for contexts not explicitly mapped at authoring time, whereas BEAR discovers them via tag matching as new departments are added.

Table 10 quantifies the benefit of combining governance with semantic retrieval: CPA attains zero recall on semantically discoverable instructions by design, while BEAR achieves 0.917–1.000 recall across corpus sizes. This is a fundamental capability gap: no system that relies solely on explicit mappings can surface these instructions, regardless of how carefully the mappings are maintained.

### 3.6 Retrieval Backend Comparison and Governance Ablation

A natural question is whether BEAR's retrieval quality stems primarily from its governance mechanisms or from the choice of retrieval backend. We compare seven retrieval backends under three

Table 11. Token efficiency on Pet Simulation corpus (58 instructions, 5,029 monolithic tokens). Governed retrieval (BEAR) achieves 85–88% savings; Random- $k$  achieves similar savings but far lower retrieval quality (Table 12(a)).

Strategy	Tokens/query	Instr.	Savings
Monolithic	5,029	58.0	—
Random- $k$	868	10.0	83%
BEAR + BGE	716	8.1	86%
BEAR + Qwen3-0.6B	716	8.2	86%
BEAR + BGE-M3	679	7.7	87%
BEAR + Qwen3-4B	679	7.7	87%
BEAR + Hash	628	6.9	88%
BEAR + BM25	593	6.5	88%
Role prompting	46	—	99%

governance conditions. Backends include four dense embedding models: BGE (BAAI/bge-base-en-v1.5, 768-dim), BGE-M3 (BAAI/bge-m3, 1024-dim), Qwen3-0.6B (Qwen/Qwen3-Embedding-0.6B, 1024-dim), and Qwen3-4B (Qwen/Qwen3-Embedding-4B, 2560-dim), alongside BM25 (sparse lexical), and a deterministic hash embedding (no semantic signal). All backends pass through the same governance pipeline; only their similarity scores differ.

We do not include ITR [5] as an evaluated backend because we were unable to reproduce its full reported pipeline from the released implementation. Governance mechanisms are orthogonal to the retrieval backend and could in principle be layered on top of ITR or any other retriever.

*Query conditions.* We evaluate four query sets of increasing difficulty: (a) *standard* ( $n=60$ ) with full context tags that satisfy `required_tags`; (b) *paraphrase* ( $n=60$ ) with minimal tags (species only) and natural-language descriptions that avoid instruction keywords; (c) *no-tag* ( $n=54$ ) with empty context tags, where no `required_tags` gates fire and only mandatory injection and similarity contribute; and (d) *complex* ( $n=60$ ) with multi-intent scenarios, temporal narratives, indirect/figurative language, and adversarial distractors designed to mislead similarity-based retrieval. The no-tag condition excludes the six species-specific personality queries (cat, dog), since their expected matches require `required_tags` on species that are not supplied with empty context\_tags. This leaves  $n=54$  (5 moods  $\times$  6 + 6 safety + 12 mixed + 6 ambiguous).

We ablate governance with three configurations: (1) *full governance* (`required_tags` + mandatory injection), (2) *mandatory only* (all `required_tags` removed), and (3) *no governance* (both mechanisms disabled).

Table 12(a) shows F1 under full governance across all backends and query types, along with per-backend retrieval latency (ms/query).

*Lower-bound baselines and token efficiency.* Random- $k$  selection ( $k=10$ , averaged over 1,000 trials) yields  $F1 \approx 0.10$ – $0.12$  regardless of query type, confirming that semantic and governed retrieval contribute beyond merely reducing the instruction set. Table 11 compares token efficiency: all retrieval-based methods achieve 83–88% savings over monolithic injection, with governed retrieval (BEAR+BM25) the most efficient (593 vs. 846 tokens) because `required_tags` select a narrower set than the full  $k=10$  budget.

*Governance ablation.* Table 12 compares full governance, mandatory-only, and no-governance across backends and query types. Three patterns emerge.

**1. required\_tags are the main driver.** Comparing full governance to mandatory-only for the same backend shows large gains: Hash improves from 0.431 to 0.835 (+0.404) and BM25 from 0.478 to 0.872 (+0.394) on standard queries. Enabling required\_tags typically adds 0.25–0.40 absolute F1 across backends and query types. This is the largest single intervention in the paper. Complex queries achieve F1 comparable to standard queries under full governance, indicating robustness to query formulation.

**2. Backend ranking flips without governance.** This is a striking result: the best backend under full governance (BM25, 0.872 F1) becomes one of the worst without governance (0.249 F1 in panel c), a 3.5× collapse. With full governance, required\_tags perform most of the selection, making the embedding signal less important. With only mandatory injection, dense models provide the best F1 since semantic similarity partially compensates for missing scope gates. Without any governance, hash embeddings exploit structured metadata via n-gram overlap and outperform dense models. This is not because they are intrinsically better, but because lexical matching can carry most of the load when metadata is available.

*Practical backend selection.* In a mature, well-tagged corpus, hash or BM25 backends are often sufficient: required\_tags handle most selection and dense embeddings add noise more than signal. The practical recommendation is to prioritize high-quality required\_tags and choose the lightest backend that meets latency and coverage needs.

*Reproducibility.* All evaluation scripts, datasets, and configuration files are available in the artifacts repository (<https://github.com/snhwang/paper-retrieval-governed-context-artifacts>).

## 4 Related Work

*Retrieval-augmented generation and context engineering.* RAG [9] established the pattern of retrieving relevant documents to augment LLM inputs, typically for factual grounding. Recent work generalizes this to context engineering, assembling the right instructions, tools, and data for each model invocation via retrieval and orchestration frameworks. In this spirit, we apply retrieval not only to knowledge documents but also to behavioral instructions and tools that shape how the model responds, not just what it knows.

*System instruction retrieval.* ITR [5] applies hybrid dense+sparse retrieval with budget-aware selection to instruction and tool corpora, reporting large token reductions and improved tool routing accuracy. ID-RAG [14] uses retrieval over a knowledge-graph identity model of beliefs, traits, and values to ground agent behavior, demonstrating that retrieval can govern persona coherence as well as factual grounding, but does not address tool governance, scope gating, or conflict resolution. Our work extends the instruction-retrieval direction by unifying tools and instructions in a single typed schema, adding governance mechanisms (required\_tags, conflict resolution, mandatory injection), and comparing against common context-construction alternatives.

*Tool retrieval and selection.* CRAFT [23] creates and retrieves from specialized toolsets but does not unify tools with behavioral instructions. Gorilla [12] trains models to generate API calls but uses a separate retrieval index. ToolkenGPT [6] integrates tool tokens into the model vocabulary, a complementary approach to our retrieval-based selection. AnyTool [4] uses hierarchical API trees, which differs from our flat unified corpus with metadata-based scope filtering. These systems improve the retrieval backend itself, and any could serve as a BEAR backend with governance layered on top. BEAR addresses the complementary question of how to select tools and behavioral directives together under a shared governance schema. Tool-DE [10] shows that LLM-generated document expansion (including synthetic tags, use-case descriptions, and limitations) improves retrieval across all embedding models. BEAR’s MetaTool experiment validates a complementary

Table 12. Governance ablation (F1@10 with 95% bootstrap CI). Query types. Standard ( $n=60$ ): full context tags; Paraphrase ( $n=60$ ): minimal tags; No-tag ( $n=54$ ): empty tags; Complex ( $n=60$ ): multi-intent, temporal, indirect, adversarial. Panel (a) includes ms/query retrieval latency. Random- $k$  operates without governance and is shown in each panel for comparison.

(a) Full governance (required_tags + mandatory injection)						
Backend	Std	Para	No-tag	Complex	ms/query	
BM25	<b>0.872</b> [0.84,0.91]	<b>0.898</b> [0.88,0.92]	0.718 [0.69,0.75]	0.870 [0.84,0.90]	0.2	
Hash	0.835 [0.80,0.87]	0.886 [0.87,0.91]	<b>0.837</b> [0.81,0.86]	<b>0.892</b> [0.86,0.92]	<1	
BGE-M3	0.796 [0.77,0.83]	0.860 [0.84,0.88]	0.801 [0.78,0.82]	0.823 [0.79,0.85]	9	
Qwen3-4B	0.793 [0.76,0.83]	0.835 [0.81,0.86]	0.754 [0.73,0.78]	0.807 [0.78,0.83]	25	
BGE	0.780 [0.76,0.81]	0.836 [0.81,0.86]	0.701 [0.68,0.72]	0.797 [0.77,0.82]	5	
Qwen3-0.6B	0.758 [0.73,0.78]	0.795 [0.78,0.81]	0.715 [0.70,0.74]	0.779 [0.75,0.81]	20	
Random- $k$	0.116 [0.11,0.12]	0.102 [0.10,0.10]	0.10 [0.10,0.10]	0.115 [0.11,0.12]	—	

(b) Mandatory only (required_tags stripped)					
Backend	Standard	Paraphrase	No-tag	Complex	
BM25	0.478 [0.45,0.50]	0.384 [0.35,0.42]	0.378 [0.34,0.41]	0.480 [0.45,0.51]	
Hash	0.431 [0.40,0.46]	0.416 [0.39,0.44]	0.447 [0.42,0.47]	0.405 [0.38,0.43]	
BGE-M3	<b>0.538</b> [0.52,0.56]	<b>0.467</b> [0.45,0.49]	0.432 [0.43,0.44]	0.528 [0.51,0.55]	
Qwen3-4B	0.531 [0.51,0.55]	0.458 [0.44,0.48]	0.440 [0.43,0.45]	<b>0.534</b> [0.51,0.56]	
BGE	0.522 [0.50,0.54]	0.439 [0.42,0.45]	<b>0.451</b> [0.44,0.46]	0.508 [0.49,0.53]	
Qwen3-0.6B	0.529 [0.51,0.55]	0.417 [0.40,0.44]	0.445 [0.43,0.46]	0.481 [0.45,0.51]	
Random- $k$	0.116 [0.11,0.12]	0.102 [0.10,0.10]	0.10 [0.10,0.10]	0.115 [0.11,0.12]	

(c) No governance (both mechanisms disabled)					
Backend	Standard	Paraphrase	No-tag	Complex	
BM25	0.249 [0.22,0.28]	0.117 [0.09,0.15]	0.201 [0.16,0.24]	0.232 [0.20,0.27]	
Hash	<b>0.304</b> [0.28,0.33]	<b>0.254</b> [0.23,0.28]	<b>0.289</b> [0.25,0.33]	<b>0.318</b> [0.28,0.35]	
Qwen3-4B	0.232 [0.21,0.26]	0.129 [0.10,0.16]	0.114 [0.08,0.15]	0.238 [0.21,0.27]	
Qwen3-0.6B	0.213 [0.19,0.23]	0.096 [0.07,0.12]	0.114 [0.08,0.15]	0.215 [0.19,0.24]	
BGE-M3	0.158 [0.14,0.18]	0.077 [0.06,0.10]	0.022 [0.01,0.04]	0.151 [0.13,0.17]	
BGE	0.155 [0.13,0.18]	0.073 [0.05,0.10]	0.105 [0.08,0.14]	0.144 [0.12,0.17]	
Random- $k$	0.116 [0.11,0.12]	0.102 [0.10,0.10]	0.10 [0.10,0.10]	0.115 [0.11,0.12]	

finding: LLM-generated required\_tags improve governance-based scope gating on the same corpus, yielding NDCG@5 = 0.840 without fine-tuning or outcome data.

*Conditional prompt construction.* Universal Conditional Logic [11] formalizes conditional prompt assembly rules. ProPS [13] explores conditional and compositional prompting. DSPy [8] compiles declarative prompt programs. These systems rely on deterministic condition matching where scope and prerequisites are encoded in code rather than as queryable metadata. Our approach augments deterministic scope filtering with semantic retrieval and explicit governance metadata, enabling discovery of instructions not anticipated in condition tables while still providing hard scope guarantees for tools.

Table 13. Summary comparison across context-construction strategies. ✓ = strong, ~ = moderate or degenerate, × = weak.

	Monolithic	Role	CPA	Random- $k$	BEAR
Precision	×	N/A	✓	×	✓
Recall	✓	N/A	✓	×	✓
Novel contexts	~	~	×	×	✓
Semantic discovery	~	×	×	×	✓
Token efficiency	×	✓	✓	~	✓
Behavioral specificity	×	×	✓	×	✓
Tool governance	×	×	~	×	✓
Mandatory safety	×	×	~	×	✓

*Row definitions.* Precision/Recall: instruction retrieval accuracy. Novel contexts: handling unanticipated contexts without code changes. Semantic discovery: surfacing instructions via similarity when no explicit mapping exists. Token efficiency: prompt/context size relative to corpus size. Behavioral specificity: context-specific behavioral detail. Tool governance: scope-gated tool selection preventing cross-domain leakage. Mandatory safety: guaranteed injection of safety/compliance directives regardless of query. For Monolithic, novelty and semantic coverage reflect indiscriminate inclusion of the entire corpus.

*Prompt engineering and agent frameworks.* Chain-of-thought [19], Self-Instruct [18], and least-to-most prompting [25] optimize individual prompt structure. ReAct [22] and Toolformer [16] focus on how agents reason about and invoke tools. Voyager [17] demonstrates retrieval-based behavioral composition in an agent setting, storing executable code indexed by embedding similarity and retrieved for task reuse. Its skills are executable code rather than governed natural-language instructions, retrieval is purely similarity-based with no scope gating or safety constraints, and the library grows through task execution rather than structured authoring. These distinctions motivate the governance layer that BEAR adds. Our work is complementary to all of these. We address which tools and instructions are selected for a given context.

## 5 Discussion

### 5.1 Paradigm Comparison

Table 13 synthesizes the comparison. No single method dominates under all conditions. When the instruction corpus is small ( $N < 20$ ), the monolithic approach is simplest and sufficient. Retrieval adds latency without changing the composed output. Role prompting is appropriate when behavioral specificity is not required. CPA remains the right choice when all contexts are known in advance, latency is critical, and the corpus is stable.

Retrieval-governed context (BEAR) becomes attractive when: (a) the corpus is large or growing, (b) novel contexts must be handled without code changes, or (c) semantic relationships between instructions matter. These conditions are increasingly common in production deployments. The key distinction is not that retrieval is more efficient (CPA achieves comparable token counts) but that governed retrieval is more adaptive, treating context construction as an open-world problem rather than a closed enumeration.

### 5.2 BEAR as Systematic Context Engineering

Retrieval-governed composition, as implemented in BEAR, is best understood as an organized and systematic substrate for context engineering. The architecture and empirical evaluations in this paper do not claim that retrieval-composition generates behavior a careful human prompt engineer cannot approximate on a small corpus; they claim that it systematizes the engineering process in

ways that matter as the corpus grows, as tool selection becomes a governance question, and as multiple stakeholders contribute to and maintain the specification. Four aspects of the systematic character of BEAR are worth making explicit.

First, authoring cost scales with corpus axes, not with their product. Each instruction in the corpus expresses one piece of behavior (one mood, one stimulus response, one interaction rule), with its own scope metadata. Adding a new axis, such as a new mood, a new relationship tier, or a new interaction type, is an append operation, one new YAML file, retrieved whenever its scope tags are present. A monolithic persona, by contrast, must describe how every new axis interacts with every axis already present, a combinatorial authoring burden. Our system-level scaling analysis (Section 3.5) quantifies the token-side consequence of this difference; the authoring-side consequence is structural.

Second, governance is a mechanism, not a convention. Scope gates, mandatory injection, priority-based ranking, and conflict resolution are first-class operations of the governed pipeline (Section 2). Our ablation (Section 3.6) shows that removing governance from any retrieval backend degrades performance into the range of unguided baselines. The ToolBench/MetaTool experiments (Section 3.1) show that the value of this governance scales with metadata richness. Corpora with structured tags benefit significantly from governance across all backends tested, while corpora without them do not. Governance is the contribution, and retrieval is the substrate it sits on.

Third, retrieval enables semantic discovery and novel-context adaptability that conditional or persona-based specifications cannot. Section 3.2 shows 0.917–1.000 recall on semantically discoverable instructions against 0.000 for conditional methods (Table 10), and directionally higher recall (0.944–1.000) on novel contexts unseen at design time. These are open-world capabilities unavailable to methods that fire only on explicitly enumerated conditions.

Fourth, instructions as data (individually authored YAML files) enables operational practices that monolithic persona text does not. Individual instructions can be edited by non-developers, versioned independently, and logged per-query for runtime observability of which specific rules fired. Automated authoring aids, such as LLM-assisted tag suggestion, corpus-level tag induction, or runtime gap detection, are all applicable at the instruction level and are the subject of a companion manuscript on behavioral evolution.

The BEAR Pet Simulation demonstrates this substrate producing context-sensitive behavior through action markers, with the LLM contributing enrichment rather than load-bearing action dispatch. The systematic properties above are where the paradigm’s distinctive value lies, and they become load-bearing as the corpus grows and as safety and compliance requirements harden.

### 5.3 Why BEAR Outperforms Fine-Tuned Retrievers on ToolBench

BEAR+BGE achieves Recall@5 = 0.679 on ToolBench without fine-tuning and without LLM involvement in retrieval, surpassing the fine-tuned ToolLLM retriever (Recall@5 = 0.456) reported by Qin et al. [15] and approaching the fine-tuned ToolBench-IR baseline (0.709). The advantage stems from scope gating reducing the effective retrieval space. Rather than searching all 3,225 APIs simultaneously, governance first restricts to the relevant category via `required_tags`, then runs embedding similarity within that subset. The signal-to-noise ratio improves because the embedding model competes against roughly 50–100 domain-matched tools rather than the full corpus. BEAR+Qwen3-4B (Recall@5 = 0.740) surpasses ToolBench-IR entirely, demonstrating that strong embeddings combined with governance metadata can exceed what fine-tuning alone achieves. Metadata encodes domain knowledge that would otherwise require labeled training examples, making BEAR practical in new domains where training data do not exist.

These results are not directly comparable to outcome-aware methods such as GRETEL [20] and OATS [2], which address a different layer of the problem. GRETEL validates candidate tools

through sandboxed execution trials at retrieval time; OATS refines tool embeddings offline toward the centroid of historically successful queries. Both improve similarity search quality after the candidate set is formed. BEAR operates upstream. Scope gating determines which tools are eligible before similarity is computed. The two layers are complementary. GRETEL or OATS could be applied within the scope-gated candidate set that BEAR produces, potentially combining the precision of governance with the recall improvements of execution-aware validation or outcome-based embedding refinement.

#### 5.4 Metadata as the Active Ingredient: MetaTool Experiment

The MetaTool experiment isolates the causal role of metadata. MetaTool tools originally lack category tags; with no metadata, governance is neutral ( $d < 0.08$ , n.s.) and BEAR reduces to pure embedding retrieval. When an LLM generates `required_tags` from tool descriptions in a single offline pass, governance improves Recall@5 by 18–30 percentage points across backends (all  $p < 0.0001$ ), with BEAR+Qwen3-4B reaching Recall@5 = 0.915 and NDCG@5 = 0.840. The mandatory-only condition collapses to baseline (BEAR+BGE: Recall@5 = 0.392), confirming that `required_tags`, not mandatory injection, drives the improvement.

A complementary condition in which query context tags are inferred from query text alone, without knowing the target tool, produces negative governance effects ( $d = -0.20$  to  $-0.70$ , all  $p < 0.0001$ ,  $n = 21,111$ ), because independently generated query and tool tags are not guaranteed to share vocabulary. Together these results show that BEAR’s governance value depends on metadata quality and alignment. High-quality tags that are consistent between corpus and runtime context reliably improve retrieval. Absent or misaligned tags provide no benefit or actively harm it. LLM-generated tags can bootstrap governance on untagged corpora, but a shared taxonomy enforced at authoring time is necessary for full benefit.

This finding is complementary to Tool-DE [10], which shows that LLM-generated document expansion improves retrieval by enriching embeddings. BEAR’s MetaTool experiment shows that the same generated content improves governance-based scope gating independently of embedding quality.

#### 5.5 Scope Gating vs. Similarity-Only Retrieval

Our ablations quantify the value of structured scope metadata. Removing `required_tags` while retaining mandatory injection and ranking degrades F1 by roughly 0.25–0.40 across backends and query types (Section 3.6). A similarity-only configuration ( $\alpha = 0$ ) that ranks purely by embeddings drops further: to F1 = 0.155 with semantic embeddings and 0.481 with hash (Table 6). For tool retrieval the stakes are higher than for text instructions, since retrieving an incorrect tool can trigger real-world actions (API calls, database mutations), making false-positive prevention critical. Accordingly, `required_tags` are enforced as a hard gate for tools: no amount of embedding noise can surface a tool whose prerequisite context is absent. For text instructions, the dual-path gate (Section 2.2) permits semantic discovery as a secondary path, reflecting the lower risk of surfacing an unexpected guideline versus an incorrect tool.

One might object that our strong results simply reflect high-quality hand-authored scope metadata. We view this as a feature, not a flaw. The architecture is explicitly designed so that structured metadata provides a deterministic safety and relevance floor, while semantic retrieval extends coverage to unanticipated contexts. The semantic component’s value is isolated in Table 10, where BEAR attains 0.917–1.000 recall on instructions unreachable by any purely metadata-based system.

## 5.6 Limitations

Our study has several limitations. First, CPA represents standard production practice (if/else context assembly) rather than a tuned ML baseline. We deliberately do not update CPA's lookup tables after adding the "fraud" department because this reflects the real-world failure mode where lookup tables fall behind as corpora evolve. Second, while our end-to-end experiment (Table 5) confirms that governance improves LLM tool selection accuracy, it measures tool name correctness rather than full task completion (e.g., whether the selected tool achieves the user's goal with correct arguments). Human evaluation of selection appropriateness remains future work. Third, the Pet Simulation and procedural customer-support corpora are synthetic by design. Controlled corpora let us isolate governance effects without confounding deployment-specific factors. We mitigate this with external validation on the ToolBench benchmark split (3,225 APIs from the 16,464-API corpus) and MetaTool (199 tools). Fourth, `required_tags` require authors to specify prerequisite tags, which adds authoring effort per instruction or tool. A missing or misspelled tag causes silent exclusion rather than a loud error, making tag quality critical. Reducing this burden is tractable with current technology. LLMs can suggest `required_tags` from instruction text at authoring time, corpus-level tag induction can bootstrap a consistent vocabulary from an existing instruction set, and runtime observability (logging which instructions were gated in or out per query) can flag tags that never fire or always fire, surfacing authoring errors as operational signals rather than silent failures. The full BEAR system's behavioral evolution module is designed around this feedback loop. LLMs assist in authoring, refining, and updating the instruction corpus, including tag suggestion and gap detection. Evaluation of behavioral evolution and inheritance is presented in a companion manuscript targeting Artificial Intelligence Review, and multi-agent knowledge governance in a manuscript targeting ACM TiiS.

Fifth, the safety guarantees provided by mandatory injection and hard tool gating are architectural, not behavioral. Governance ensures that safety instructions are always present in the composed context and that out-of-scope tools are never retrieved. Whether the LLM actually follows those instructions is a general property of LLMs, not a limitation of this architecture: LLMs may interpret instructions inconsistently, ignore directives under adversarial prompting, or produce outputs that deviate from specification. This problem is not specific to BEAR and applies to any system that uses LLMs as executors. An additional open question is whether adversarial queries can exploit the soft-retrieval path for text instructions to surface unintended behavioral guidance.

To support reproducibility, we release the corpora, procedural generator, query sets, and scoring scripts (<https://github.com/snhwang/paper-retrieval-governed-context-artifacts>). Future work should include task-level evaluations (e.g., full task completion rates with governed vs. ungoverned tool selection), human evaluation of tool-selection appropriateness (subject to IRB approval), and authoring tools with controlled tag vocabularies, conflict browsers, and authoring-time validation to reduce the metadata maintenance burden in team settings.

## 6 Conclusion

We presented a systematic comparison of context-construction strategies and a governed tool-instruction retrieval architecture that unifies tools and behavioral instructions under a single typed, scope-gated, priority-weighted retrieval pipeline. Empirical evaluation supports six main findings:

- (1) Retrieval-governed context offers adaptability to novel contexts and semantic discovery. It attains 0.917–1.000 recall on semantically discoverable instructions (McNemar  $p < 0.003$ ,  $h = 2.56$ – $3.14$ ) while CPA scores 0.000 by construction, and achieves directionally higher recall on novel contexts across all corpus scales.

- (2) Scope-gated retrieval eliminates cross-domain tool leakage while achieving roughly 90% token savings over static tool injection, and it scales to hundreds of instructions without increasing per-query context length.
- (3) Structured scope metadata (`required_tags`) is the dominant contributor to retrieval quality: ablating `required_tags` reduces F1 by roughly 0.25–0.40 across backends and query types, whereas changing embedding models within a fixed governance configuration alters F1 by less than 0.07 among dense backends on standard queries.
- (4) The BEAR Pet Simulation demonstrates context-sensitive behavior in a live implementation: pets respond differently to the same stimulus depending on their governed context (bonded vs. wary player, excited vs. cautious mood), with action markers in retrieved instructions driving animation, movement, and state transitions directly without requiring an LLM in the critical path. A running demonstration is available at <https://github.com/snhwang/bear>.
- (5) The choice of retrieval backend is secondary to governance on rich-metadata corpora: multiple backends all achieve strong results under full governance, while removing governance from any backend degrades performance into the range of unguided baselines.
- (6) External benchmarks show that governance value scales with metadata richness. On ToolBench (3,225 APIs from the benchmark evaluation split), governance significantly improves every backend (all  $p < 0.0001$ ,  $d = 0.21$ – $0.50$ ), with effect sizes inversely related to embedding quality. On MetaTool (199 tools without category metadata), governance has negligible effect ( $d < 0.08$ , n.s.). When LLM-generated tags are added offline (MetaTool+Tags), governance improves Recall@5 by 18–30 percentage points (all  $p < 0.0001$ ), demonstrating that BEAR’s value can be bootstrapped onto untagged corpora. When tags are instead inferred from query text alone at runtime (MetaTool+QueryTags), governance hurts retrieval ( $d = -0.20$  to  $-0.59$ ), establishing that tag alignment between corpus and query is required. Governance and embedding quality are orthogonal contributors.

The shift from “engineer the prompt” to “engineer and govern the context” mirrors the broader RAG insight, applied here to behavioral governance rather than factual grounding. BEAR provides a backend-agnostic context-engineering layer in which tools and behavioral directives share a unified corpus and governance pipeline; it selects not just which tools exist, but which instructions, tools, and safety constraints are appropriate for an agent’s current context. Future work includes task-level and human evaluations in real deployments and extending governed context to longer-horizon planning and multi-agent systems.

## References

- [1] Harrison Chase. 2022. LangChain. <https://github.com/langchain-ai/langchain>. Open-source framework for LLM application development.
- [2] Huamin Chen, Xunzhuo Liu, Junchen Jiang, Bowei He, and Xue Liu. 2026. Outcome-Aware Tool Selection for Semantic Routers: Latency-Constrained Learning Without LLM Inference. arXiv preprint arXiv:2603.13426. <https://arxiv.org/abs/2603.13426>.
- [3] Jacob Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences* (2nd ed.). Lawrence Erlbaum Associates, Hillsdale, NJ.
- [4] Yu Du, Fangyun Wei, and Hongyang Zhang. 2024. AnyTool: Self-Reflective, Hierarchical Agents for Large-Scale API Calls. In *International Conference on Machine Learning (ICML)*. PMLR, Vienna, Austria, 11812–11829.
- [5] Uria Franko. 2025. Dynamic System Instructions and Tool Exposure for Efficient Agentic LLMs. arXiv preprint arXiv:2602.17046. <https://arxiv.org/abs/2602.17046>.
- [6] Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. ToolkenGPT: Augmenting Frozen Language Models with Massive Tools via Tool Embeddings. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 36. Curran Associates, Inc., Red Hook, NY, USA. Oral presentation.
- [7] Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. 2024. MetaTool Benchmark for Large Language Models: Deciding Whether to Use Tools and

- Which to Use. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*. OpenReview.net, Vienna, Austria. Poster.
- [8] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Mober, et al. 2024. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*. OpenReview.net, Vienna, Austria. Poster.
- [9] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33. Curran Associates, Inc., Red Hook, NY, USA, 9459–9474.
- [10] Xuan Lu, Haohang Huang, Rui Meng, Yaohui Jin, Wenjun Zeng, and Xiaoyu Shen. 2025. Tools Are Under-Documented: Simple Document Expansion Boosts Tool Retrieval. arXiv preprint arXiv:2510.22670. <https://arxiv.org/abs/2510.22670>.
- [11] Anthony Mikinka. 2025. Universal Conditional Logic: A Formal Language for Prompt Engineering. arXiv preprint arXiv:2601.00880. <https://arxiv.org/abs/2601.00880>.
- [12] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large Language Model Connected with Massive APIs. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 37. Curran Associates, Inc., Red Hook, NY, USA, 126544–126565.
- [13] Jonathan Pilault, Can Liu, Mohit Bansal, and Markus Dreyer. 2023. On Conditional and Compositional Language Model Differentiable Prompting. In *International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI Organization, Macao, SAR China, 5225–5233.
- [14] Jacob Platinick et al. 2025. ID-RAG: Identity Retrieval-Augmented Generation for Long-Horizon Persona Coherence in Generative Agents. arXiv preprint arXiv:2509.25299. <https://arxiv.org/abs/2509.25299>.
- [15] 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 Gerber, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*. OpenReview.net, Vienna, Austria. Spotlight presentation.
- [16] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 36. Curran Associates, Inc., Red Hook, NY, USA. Oral presentation.
- [17] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research* (2024). Article ID ehfRiF0R3a. <https://openreview.net/forum?id=ehfRiF0R3a>.
- [18] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, Toronto, Canada, 13484–13508.
- [19] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 35. Curran Associates, Inc., Red Hook, NY, USA, 24824–24837.
- [20] Zongze Wu, Yani Guo, Churong Liang, and Runnan Li. 2025. GRETEL: A Goal-driven Retrieval and Execution-based Trial Framework for LLM Tool Selection Enhancing. arXiv preprint arXiv:2510.17843. <https://arxiv.org/abs/2510.17843>.
- [21] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-Pack: Packed Resources For General Chinese Embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, 641–649. doi:10.1145/3626772.3657878
- [22] 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)*. OpenReview.net, Kigali, Rwanda. Oral presentation.
- [23] Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R Fung, Hao Peng, and Heng Ji. 2024. CRAFT: Customizing LLMs by Creating and Retrieving from Specialized Toolsets. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, Vienna, Austria. Poster.
- [24] Mingqian Zheng, Jiaxin Pei, and David Jurgens. 2024. When “A Helpful Assistant” Is Not Really Helpful: Personas in System Prompts Do Not Improve Performances of Large Language Models. arXiv preprint arXiv:2311.10054. <https://arxiv.org/abs/2311.10054>.
- [25] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, Kigali, Rwanda. Poster.