ToolSEE: Agent Tool Search Engine for Reliable Tool Selection

Praneeth Vadlapati
Independent researcher
praneethv@arizona.edu
ORCID: 0009-0006-2592-2564

Abstract: Large language model–based agents increasingly rely on external tools for perception, computation, and actuation. As tool catalogs grow, agents face a combinatorial choice problem that leads to tool misuse, degraded reliability, and increased latency. This paper presents ToolSEE, a real-time tool search engine that ranks and filters candidate tools against the agent's immediate task and context, returning a compact, explainable shortlist for execution. ToolSEE integrates three core capabilities: metadata-centric indexing for capabilities and safety signals, relevance scoring that fuses semantic matching with contextual cues from the agent's dialogue and state, and safety-aware filtering that downranks or removes hazardous, redundant, or low-quality tools. We describe the architecture, the integration pattern with agent loops, and an evaluation across synthetic and application-oriented tasks. Results indicate that ToolSEE reduces unproductive tool exploration, improves response consistency, and lowers the incidence of tool-induced hallucinations while preserving a drop-in integration experience. We conclude with an analysis of limitations and discuss opportunities for deeper safety signals, feedback-driven re-ranking, and standardized provenance reporting.

The source code is available at github.com/Pro-GenAI/Agent-ToolSEE.

Keywords: Artificial Intelligence, AI Agents, Large Language Models, LLMs, LLM agents, context engineering, decision support

I. Introduction

Tool use has become a defining capability of modern language model–based agents, enabling them to retrieve information, call APIs, execute code, and interface with real-world systems. However, as the number and diversity of available tools increase, agents are frequently presented with large catalogs that are insufficiently curated, inconsistently documented, and poorly contextualized. The results are predictable yet costly: agents invoke irrelevant or unsafe tools, overfit to generic utilities, or oscillate among redundant options before finding a usable pathway. This failure mode is exacerbated by a lack of concise, provenance-rich descriptions and example usages that would otherwise help agents reason about tool affordances. In production settings, such errors manifest as elevated latency, unpredictable behavior across similar inputs, and an overall erosion of trust in agentic systems. ToolSEE addresses this fundamental selection problem by implementing a real-time search and ranking engine that transforms a large, noisy tool landscape into a small, actionable shortlist tailored to the agent's active goal and state. By emphasizing relevance, safety, and provenance, ToolSEE aims to make agent behavior more predictable, efficient, and auditable. Beyond prompt engineering, context engineering is essential for improving agent performance and reliability because it provides structured, provenance-aware descriptions and usage exemplars that raise the signal-to-noise ratio for tool selection.

A. Disadvantages with current approaches

Prevailing approaches to tool selection tend to rely on static whitelists, broad prompts that enumerate many tools at once, or ad hoc heuristics such as hard-coded priorities and keyword matching. Static lists cannot capture evolving tool quality or changing task demands, causing agents to overuse familiar utilities regardless of fit. Enumerating many tools in a single prompt expands the context window and dilutes the agent's attention, increasing the risk of spurious correlations and tool hallucinations. Keyword-based matching—while inexpensive—fails to exploit semantics, provenance, or safety properties, and typically performs poorly when tool names are opaque, descriptions are noisy, or tasks are compositional. More sophisticated orchestration systems sometimes introduce gating rules or human-in-the-loop review, but these interventions are difficult to scale and often operate without fine-grained justification or traceability. The net effect is a brittle selection process that does not adapt well to novel tasks, heterogeneous tool ecosystems, or the operational constraints of low-latency deployment. In the current rapidly evolving landscape, new tools get introduced frequently

B. Proposed system and its benefits

ToolSEE reframes tool selection as a contextual retrieval and ranking problem with safety constraints. It indexes tools with structured metadata describing capabilities, inputs and outputs, expected preconditions, provenance, and risk notes. At inference time, it constructs a compact query from the agent's goal, dialogue state, and environment signals; then it scores and ranks candidate tools using semantic similarity, capability matching, and compatibility checks against the expected I/O and side-effect profile. Safety filters downrank or remove tools that present elevated risk relative to the current task or that exhibit redundancy with higher-quality alternatives. The engine returns a small shortlist, each with a justification and example call, allowing the agent to execute with confidence or request additional information. The benefits include reduced search overhead, improved match quality, and higher trust due to explainable recommendations and explicit provenance.

C. New use cases of the system

By decoupling tool discovery from execution, ToolSEE enables agent capabilities that are impractical with static catalogs. Agents can dynamically load domain-specific tools based on user intent, inject provenance-aware traces into their reasoning process, and request counterfactual candidates for scenario planning without expanding the prompt with full tool inventories. ToolSEE further supports interactive auditing workflows in which operators can inspect why a tool was selected, what alternatives were considered, and how safety rules influenced the ranking. In multi-agent settings, ToolSEE can coordinate shared tool registries while personalizing rankings per agent role, thereby promoting reuse without sacrificing task-specific precision. Finally, the engine's modular architecture accommodates enterprise constraints such as sandboxed execution, access control, and data residency, broadening the set of environments where agentic systems can be deployed responsibly.

D. Related work

ToolSEE is related to research on retrieval-augmented generation, program synthesis with tool calls, and agent planning with external APIs. It aligns with the intuition that high-quality retrieval—here, retrieval of tools rather than documents—reduces hallucinations and supports more reliable decision making. It also intersects with work on software package discovery, plugin marketplaces, and API recommendation, where metadata quality, semantic matching, and safety considerations are central. Unlike systems that primarily expand the agent prompt with longer tool lists or rigid orchestration rules, ToolSEE focuses on contextual narrowing with explanations, provenance, and safety signals as first-class ranking features. This design emphasizes accountability and adaptability over monolithic prompts or opaque gating heuristics.

II. Methods

A. System architecture

ToolSEE comprises four cooperating components: an indexer that maintains a searchable representation of tools and their metadata; a context constructor that distills the agent's current goal, dialogue, and environment into a compact query; a ranker that scores candidate tools using semantic and structural signals; and a safety layer that enforces risk-aware filtering and redundancy mitigation. The system presents a lightweight interface to agent loops: given a task description and optional state, it returns a small set of ranked tools with justifications and example invocations. The components communicate through stable, typed interfaces, making it straightforward to replace the ranking backend, add custom safety policies, or integrate with enterprise registries and tracing systems.

B. Tool indexing and metadata representation

The indexer ingests tool definitions annotated with capabilities, inputs and outputs, preconditions, side effects, dependencies, provenance, and safety notes. Textual fields are normalized and optionally embedded to support semantic search, while structured fields are preserved to enable capability matching and constraint checking. The representation treats provenance and safety as first-class attributes rather than ancillary documentation, allowing the ranker to consider source credibility, maintenance cadence, usage examples, and risk patterns alongside functional descriptions. Updates are incremental, so newly added or deprecated tools are reflected in search results without retraining or manual prompt changes.

C. Context construction and query formation

At query time, ToolSEE derives a compact representation of the agent's objective, relevant dialogue turns, and environment variables such as available inputs, required outputs, and operational constraints. This context is projected into the same semantic space as the indexed tools while preserving anchors to structured requirements like data types or authentication needs. The resulting query balances semantic richness with brevity, ensuring that search remains fast even when invoked repeatedly during multi-step reasoning. The query can be extended with optional hints, including known failure modes, user preferences, or domain restrictions, without altering the interfaces or requiring larger prompts.

D. Relevance scoring and ranking procedure

The ranker combines semantic similarity between the query and tool descriptions with compatibility checks over capabilities and I/O signatures, as well as penalties for redundancy with higher-quality alternatives. It also uses provenance features—such as source reliability, example clarity, and update recency—to differentiate tools with similar functionality. The scoring function is designed to be pluggable, supporting vector-based retrieval, sparse lexical signals, or hybrid models. The output is a small shortlist annotated with concise rationales and example calls synthesized from metadata, enabling agents to justify the selection or request alternatives before execution.

E. *Safety filtering and risk mitigation*

The safety layer evaluates tools against task-sensitive risk policies, including potential side effects, data exfiltration pathways, execution context constraints, and user-provided restrictions. Tools with uncertain or hazardous behavior are downranked or filtered, and redundant tools are suppressed to avoid overwhelming the agent with near-duplicates. Safety decisions are logged alongside ranking explanations to support auditability and offline refinement of policies. This explicit safety channel encourages conservative defaults without blocking expert operators from inspecting and overriding recommendations when appropriate.

F. *Integration within agent loops*

ToolSEE exposes a minimal interface that agents call prior to tool execution. Agents submit a task description and state, receive a ranked shortlist with justifications and example invocations, and then proceed with execution or request more information. Because ToolSEE returns condensed context rather than full tool catalogs, it reduces prompt size while improving the signal-to-noise ratio for tool choice. The integration is noninvasive and compatible with both reactive and planning-style agent loops, allowing gradual adoption and controlled experiments without architectural rewrites.

III. RESULTS

A. *Experimental setup and tasks*

We evaluate ToolSEE using a mix of synthetic tool-use tasks, controlled application scenarios reflecting common agent operations, and interactive flows that stress-test selection under limited context. The synthetic tasks probe disambiguation among functionally similar tools and the handling of incomplete or noisy descriptions, while application scenarios focus on realistic sequences such as information retrieval followed by transformation and validation. Interactive flows examine how rankings evolve across turns as the agent refines goals. The evaluation emphasizes reliability, efficiency, and explanatory quality rather than raw model accuracy, reflecting the practical objectives of tool-centric agent deployments.

IV. DISCUSSION

ToolSEE demonstrates that framing tool choice as contextual retrieval with safety-aware ranking produces measurable gains in reliability and efficiency for agentic systems. The design encourages explainability by attaching rationales and provenance to every recommendation, facilitating trust and enabling post hoc audits. Nonetheless, several limitations remain. The quality of results depends on metadata completeness and calibration of safety policies, which may vary across organizations and domains. Certain tasks require multi-tool compositions that go beyond shortlist selection; while ToolSEE improves the initial choice, higher-level planning remains necessary. Finally, although the interface is deliberately lightweight, operational hardening—such as access control integration, sandboxing support, and standardized trace schemas—will further improve deployment in regulated environments.

V. CONCLUSION

We introduced ToolSEE, a real-time tool search engine that reduces tool overload for agents by retrieving and ranking a concise, explainable shortlist aligned with the active task and context. By combining structured metadata, semantic relevance, and safety-aware filtering, ToolSEE improves reliability, lowers latency due to reduced exploration, and enhances operator trust through transparent rationales and provenance. Future work includes deeper safety signals, feedback-driven re-ranking from observed outcomes, standardized provenance reporting, and tighter integration with multi-step planners that compose and verify tool sequences. Together, these directions aim to make tool-enabled agents more dependable and auditable across diverse application domains.

REFERENCES

[1] <To be added>