JSPLIT: A Taxonomy-based Solution for Prompt Bloating in Model Context Protocol

[1] Janea Systems

[2] BigFilter.ai

Emanuele Antonioni[1], Stefan Markovic[1], Anirudha Shankar[1], Jaime Bernardo[1], Lovro Markovic[1], Silvia Pareti[2], Benedetto Proietti[1]

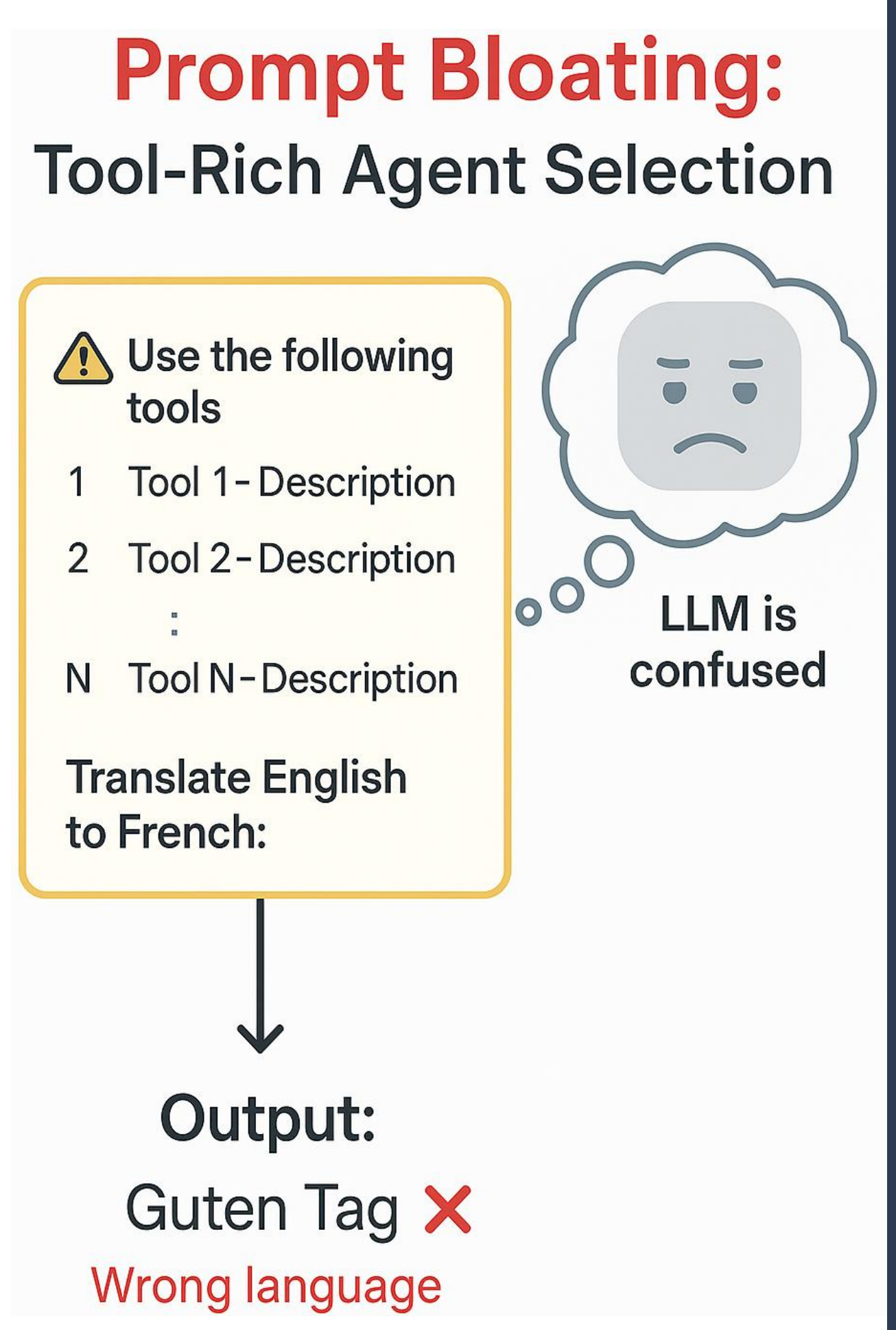
Abstract

Modern LLM applications increasingly rely on external tools, enabled by standards like the Model Context Protocol (MCP), which describes tool capabilities inside the prompt. However, as toolsets grow, prompts bloat—raising token costs and latency while reducing task success due to irrelevant tool inclusion. We present **JSPLIT**, a taxonomy-driven framework for scalable MCP tool use that organizes tools hierarchically and selects only the most relevant tools for a given user query. Beyond prompt optimization, the taxonomy provides diagnostic transparency, allowing developers to trace and debug tool selection behavior. We describe JSPLIT's taxonomy design, selection algorithm, and evaluation dataset. Results show that JSPLIT substantially reduces prompt size without degrading response quality, and at larger tool scales, improves tool selection accuracy—lowering cost while increasing task success in complex multi-tool agent environments.

Introduction

(1) From chatbots to autonomous agents → trust becomes the core issue
AI is rapidly moving from conversational LLMs to **agents** that can autonomously execute actions through external tools (APIs, databases, enterprise systems). This shift raises a key challenge: **trustability**—agents must behave predictably, select appropriate tools, and make correct decisions without constant human control.

(2) MCP enables tool-connected agents, but scaling breaks reliability
The **Model Context Protocol (MCP)** standardizes how agents access tools via structured metadata. However, supporting many tools often requires injecting large tool descriptions into the prompt, creating **prompt bloating**, which increases cost/latency and reduces trust by making tool choice noisier and less reliable.



(3) JSPLIT reduces prompt bloating while improving traceability
To address this, **JSPLIT** organizes MCP tools into a **hierarchical taxonomy** and includes only tool categories relevant to the user query. This not only shrinks prompts, but also improves **transparency and debuggability**, allowing developers to trace and audit tool selection decisions—strengthening agent trustworthiness at scale.

Methodology

(1) JSPLIT: tool-augmented agent execution
JSPLIT is a modular agent framework that solves user queries by combining **LLM reasoning** with **external tool execution** via **MCP servers**. It is designed for scalable, multi-step task resolution in tool-rich environments.

(2) Key component: Taxonomy-MCPResolver for selective tool access
At the core of JSPLIT is the **Taxonomy-MCPResolver**, which reduces tool overload by **classifying the user query into a hierarchical taxonomy** and exposing only the most relevant MCP servers. This keeps the agent context smaller and makes tool routing more controlled and explainable.

(3) Iterative call loop with filtered tools
After server selection, the LLM enters a **call loop**: it either answers directly or invokes tools from the selected servers, appending tool outputs to the context until a final answer is produced or an iteration limit is reached. Outputs include the final answer, used servers, and token statistics. A baseline **Passthrough resolver** is also supported to compare against unfiltered tool access.

(4) Resolver logic: classify → map/rank
Tool selection follows two phases:
Taxonomy classification: the LLM selects the most specific leaf category for the query.
MCP selection: tools mapped to that category are chosen directly or **ranked by the LLM** when multiple candidates exist.

(5) Taxonomies + datasets for evaluation
JSPLIT is built on two taxonomy versions (v1 → v2) and evaluated using:
a ~2,000 MCP server dataset labeled by expert + LLM-assisted classification, and
a ~200 query dataset with ground-truth tool and taxonomy labels for measuring selection accuracy and end-to-end success.

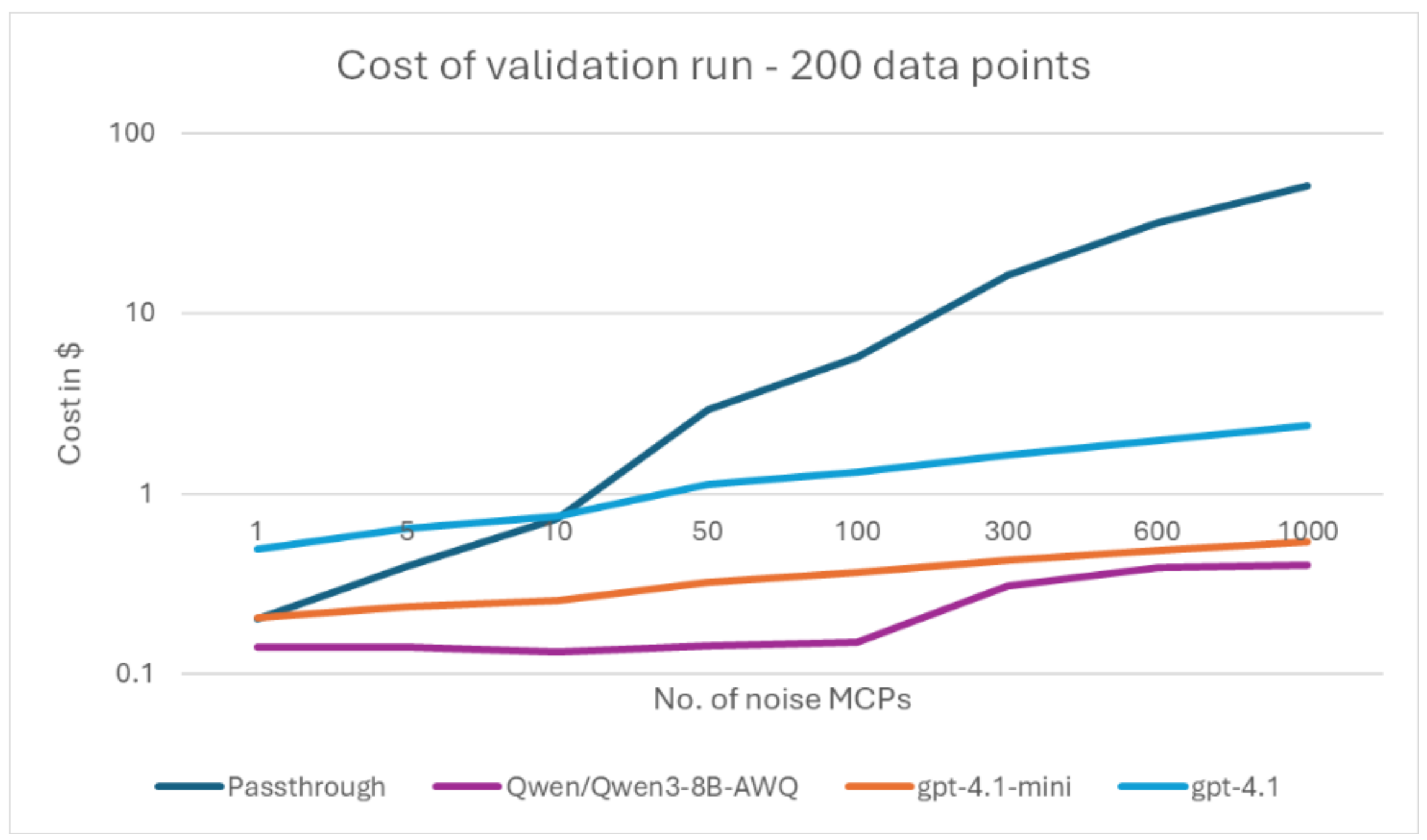
Conclusion

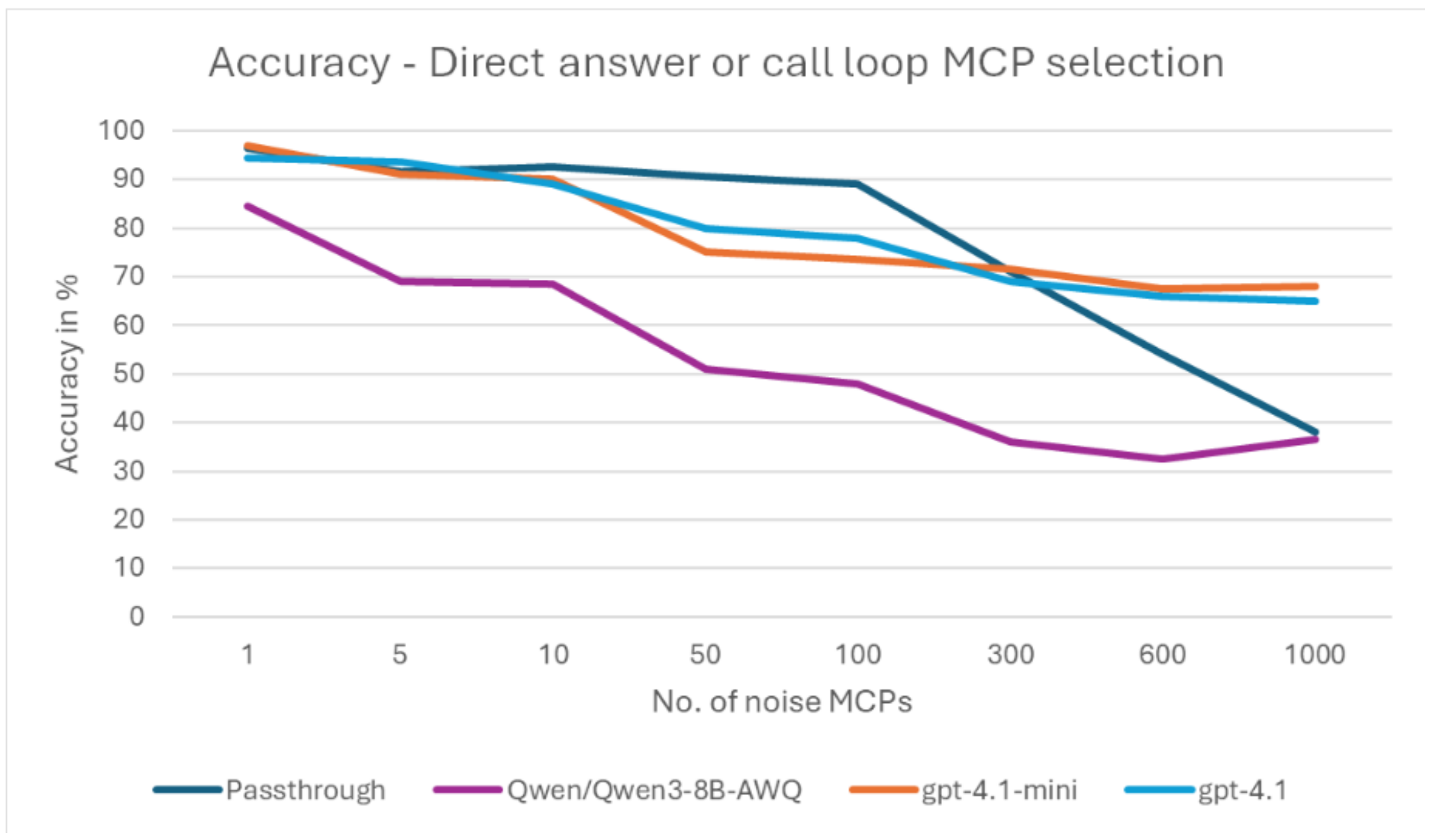
(1) Contribution: JSPLIT reduces agent prompt bloating by using a taxonomy to include only relevant MCP tools in context
.
(2) Results: It cuts token usage significantly while preserving performance and improves tool selection at large tool scales versus full-context baselines.

(3) Trust + next steps: The taxonomy increases traceability for debugging, with future work on better category descriptions, real-time onboarding, and a Taxonomy v3.

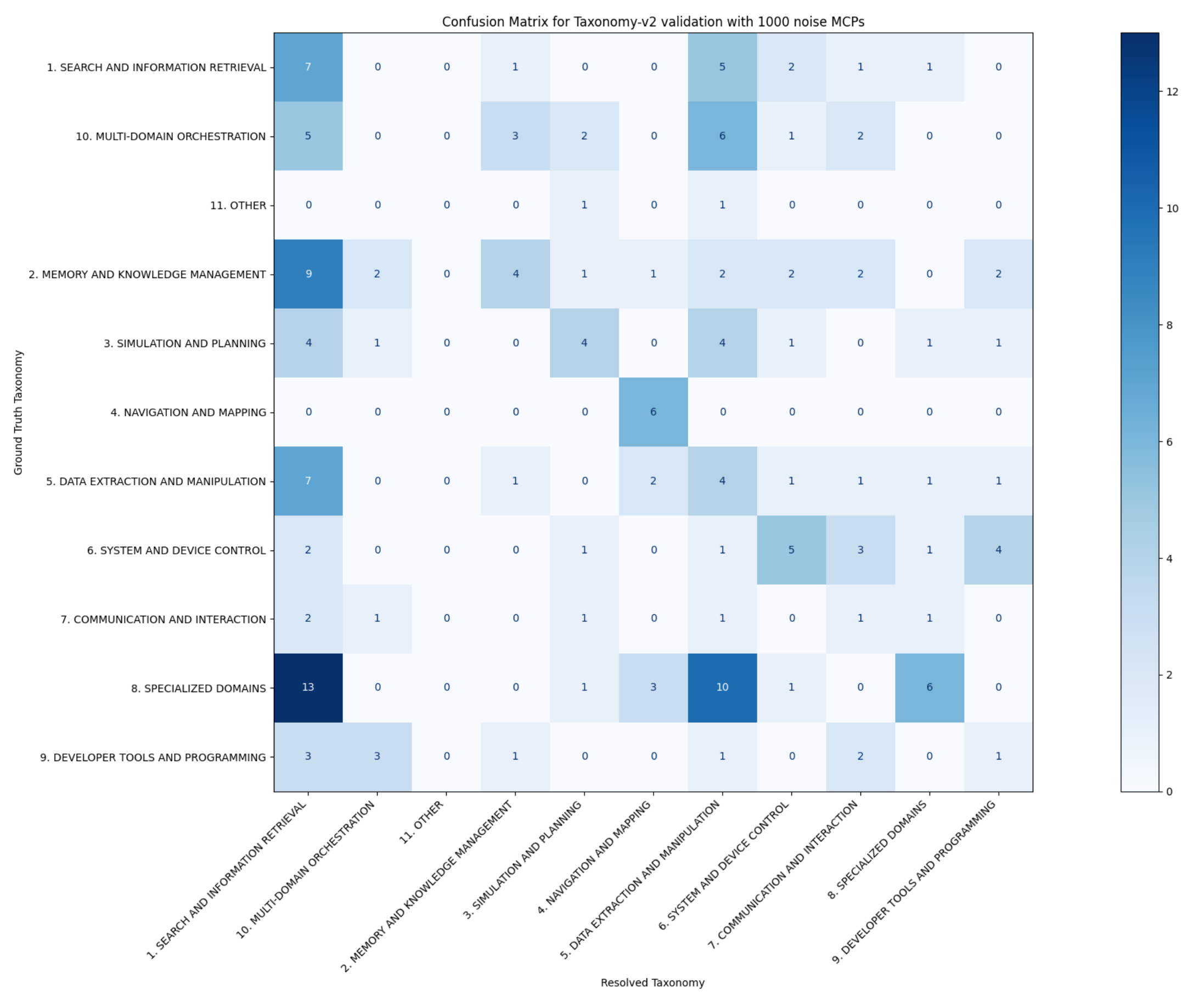
Results

(1) Setup and goal
We evaluate JSPLIT on **tool-selection accuracy** (did it call the correct MCP server?) and **LLM cost** (token usage / estimated cost).
(2) Needle-in-a-haystack stress test
For each query, the correct tool is mixed with **1–1,000 irrelevant MCP servers** sampled from a ~2,000-tool pool, testing performance as tool clutter scales.
(3) Compared systems
Passthrough: all tools injected into the prompt
JSPLIT + Taxonomy v1 / v2: taxonomy-based filtering before the LLM call loop
(4) Key results
As the number of tools grows, **Passthrough accuracy degrades sharply (<40%)**, while **JSPLIT stays stable (~69% with Taxonomy v2)** and reduces input-token cost by **100×**.





(5) Ablation + error analysis
Using a **local model** for tool filtering is cheaper but significantly less accurate; **small API models** give the best cost/accuracy trade-off. Confusion-matrix analysis shows most errors come from **overlapping taxonomy classes**, especially over-predicting "Search."



	1. SEARCH AND INFORMATION RETRIEVAL	10. MULTI-DOMAIN ORCHESTRATION	11. OTHER	2. MEMORY AND KNOWLEDGE MANAGEMENT	3. SIMULATION AND PLANNING	4. NAVIGATION AND MAPPING	5. DATA EXTRACTION AND MANIPULATION	6. SYSTEM AND DEVICE CONTROL	7. COMMUNICATION AND INTERACTION	8. SPECIALIZED DOMAINS	9. DEVELOPER TOOLS AND PROGRAMMING
1. SEARCH AND INFORMATION RETRIEVAL	7	0	0	1	0	0	5	2	1	1	0
10. MULTI-DOMAIN ORCHESTRATION	5	0	0	3	2	0	6	1	2	0	0
11. OTHER	0	0	0	0	1	0	1	0	0	0	0
2. MEMORY AND KNOWLEDGE MANAGEMENT	9	2	0	4	1	1	2	2	2	0	2
3. SIMULATION AND PLANNING	4	1	0	0	4	0	4	1	0	1	1
4. NAVIGATION AND MAPPING	0	0	0	0	0	6	0	0	0	0	0
5. DATA EXTRACTION AND MANIPULATION	7	0	0	1	0	2	4	1	1	1	1
6. SYSTEM AND DEVICE CONTROL	2	0	0	0	1	0	1	5	3	1	4
7. COMMUNICATION AND INTERACTION	2	1	0	0	1	0	1	0	1	1	0
8. SPECIALIZED DOMAINS	13	0	0	0	1	3	10	1	0	6	0
9. DEVELOPER TOOLS AND PROGRAMMING	3	3	0	1	0	0	1	0	2	0	1