PinePaper-ToolBench: Benchmarking Tool Selection for MCP-Based Design Agents

PinePaper Research
research@pinepaper.studio

Abstract

As LLM agent tool catalogs grow into the hundreds, selecting the right tools from a Model Context Protocol (MCP) server becomes a retrieval problem. Yet no public benchmark exists for evaluating tool selection methods in this setting. We introduce **PinePaper-ToolBench**, a benchmark of 582 test cases across 6 complexity tiers—from single-tool explicit queries to cross-domain ambiguous requests—covering 212 MCP tools for creative design.

Using this benchmark, we find that **BM25 over taxonomy-enriched documents is already highly effective**, achieving Recall@5 of 0.854. We then ask: does structured knowledge help? We evaluate **KG-Hybrid**, a two-phase method that augments BM25 with knowledge graph structural re-ranking via method-to-tool linking. KG-Hybrid achieves the highest Recall@5 of **0.868**, with the improvement over BM25 statistically significant at the aggregate level ($p = 0.045$), though gains concentrate on compositional and cross-domain tiers. The gains are largest on harder tiers: on compositional queries (T4), KG-Hybrid reaches 0.932 vs. BM25's 0.892 (+4.0 points); on cross-domain ambiguous queries (T6), KG-Hybrid scores 0.608 vs. 0.578 (+**3.0 points**). Ablation studies confirm that method-to-tool `implements` edges provide the primary structural signal, while graph propagation (PPR) has negligible effect.

We validate at scale with an extended benchmark (Track B) of 1,000+ cases over 500+ tools using vocabulary-separated synthetic tool generation with full KG integration. All code, benchmark data, and statistical tests are open-sourced to support reproducibility and future work on tool selection for LLM agents.

1 Introduction

The Model Context Protocol (MCP) [Anthropic, 2024] enables LLM agents to invoke external tools through a standardized interface, complementing function-calling capabilities now available across major LLM providers [OpenAI, 2024]. Production MCP servers now expose hundreds of tools: PinePaper Studio's creative design server, for example, provides 212 tools and 39 generators spanning shapes, animations, filters, diagrams, and rigging operations. When an LLM agent receives a user instruction, it must select the correct subset of tools from this catalog to fulfill the request.

The simplest approach—enumerating all tool schemas in the LLM prompt—becomes untenable as tool counts grow. A 500-tool catalog with typical JSON schemas can consume 50,000+ tokens, exceeding context limits and degrading selection accuracy due to attention dilution. Prior work has explored tool retrieval via embedding similarity [Patil et al., 2023], tool graphs [Qu et al., 2024], and generation-based approaches [Yang et al., 2024]. However, no public benchmark exists for systematically evaluating tool selection methods in MCP-based agent systems, and it remains unclear whether structured knowledge (e.g., knowledge graphs capturing tool relationships) provides meaningful gains over simple lexical retrieval baselines.

We address both gaps. First, we introduce **PinePaper-ToolBench**, a public benchmark with 582 test cases across 6 complexity tiers and 212 MCP tools. Second, we use this benchmark to evaluate whether knowledge graph features improve tool selection beyond BM25. Our

key finding is nuanced: **BM25 over taxonomy-enriched documents is surprisingly competitive** (Recall@5 = 0.855), and KG augmentation provides targeted gains that concentrate on compositional and cross-domain queries—precisely the hard cases where lexical matching fails—though the aggregate improvement is statistically significant ($p = 0.045$).

Contributions.

1. **PinePaper-ToolBench**: The first public benchmark for MCP tool selection, with 582 test cases across 6 complexity tiers, per-query ground truth, and difficulty annotations. Extended to 1,000+ cases over 500+ tools in Track B.[1]
2. **Strong BM25 baseline**: We show that BM25 over taxonomy-enriched documents achieves Recall@5 of 0.854 with 94% token reduction, establishing a high bar for future methods.
3. **Targeted KG gains**: KG-Hybrid achieves 0.868 R@5, the highest among all methods. The aggregate improvement over BM25 is statistically significant ($p = 0.045$), with gains concentrating on compositional (T4: +4.0) and cross-domain (T6: +3.0 points) queries, with full ablation studies, hyperparameter sweeps, and statistical significance tests.

2 Related Work

Tool-augmented LLMs. Toolformer [Schick et al., 2024] taught language models to insert API calls during generation. ToolLLM [Qin et al., 2024] scaled to 16,000+ real-world APIs using a retrieval-then-planning approach. Gorilla [Patil et al., 2023] fine-tuned LLMs on API documentation to improve tool invocation accuracy. HuggingGPT [Shen et al., 2024] used ChatGPT as a controller to orchestrate Hugging Face models. ToolkenGPT [Hao et al., 2024] augments frozen LMs with tool embeddings, treating tools as special tokens. These approaches either embed tool knowledge directly in model weights or rely on flat retrieval over tool descriptions.

Graph-based tool selection. ToolNet [Qu et al., 2024] connects tools via a graph where edges represent functional similarity, enabling multi-hop tool discovery. ToolGen [Yang et al., 2024] generates tool identifiers as tokens, treating retrieval as a generation problem. Agent-as-a-Graph [Chen et al., 2024] models the agent's decision process as a graph neural network. Retrieval-augmented tool selection for MCP has been explored in concurrent work [Xu et al., 2024]. Our work differs by combining a domain-specific knowledge graph (with typed edges: `implements`, `similar`, `related_to`) with a standard IR retrieval phase, rather than replacing retrieval entirely.

Information retrieval baselines. BM25 [Robertson and Zaragoza, 2009] remains a strong baseline for document retrieval [Manning et al., 2008], often competitive with neural methods on short queries. TF-IDF with cosine similarity [Salton and Buckley, 1988] provides a simpler alternative. Retrieval-augmented generation (RAG) [Lewis et al., 2020] combines retrieval with generation but typically uses dense embeddings rather than structured knowledge. Our work shows that BM25 over taxonomy-enriched documents is already highly effective for tool selection, and that KG features provide complementary gains primarily on compositional and cross-domain queries.

[1] Benchmark data and code: `https://github.com/pinepaper/pinepaper-tool-selection-research`

3 Method

3.1 Knowledge Graph Construction

The PinePaper knowledge graph is constructed from three sources:

1. **Design taxonomy**: 178 design methods organized into 18 categories (shapes, styling, animations, relations, filters, effects, diagrams, 3D, path operations, data visualization, etc.), each mapped to one or more MCP tools via `implements` edges.
2. **Tool manifest**: 212 MCP tools and 39 generators with their JSON schemas, descriptions, and tags.
3. **Structural fingerprints**: 16-dimensional vectors per node capturing node type, degree, edge type distribution, and neighbor type distribution, used to compute cosine-similarity-based `similar` edges [Ribeiro et al., 2017].

The resulting graph contains method, tool, generator, concept, category, template, and example nodes connected by typed edges (`implements`, `similar`, `requires`, `alternative`, `enhances`, `related_to`, `has_example`).

3.2 Document Corpus

For each unique MCP tool in the taxonomy, we construct a document by aggregating text from all design methods that map to it:

$$d_t = \text{concat}(t_{\text{name}}, \{m_{\text{name}}, m_{\text{desc}}, m_{\text{tags}}, m_{\text{examples}}\}_{m \in \mathcal{M}(t)})$$

where $\mathcal{M}(t)$ is the set of methods implementing tool t. This produces a corpus of documents enriched with domain vocabulary beyond the raw tool schemas.

3.3 KG-Hybrid Retrieval

KG-Hybrid operates in two phases:

Phase 1: BM25 Retrieval. Given an instruction q, we tokenize it and score each document d_t using Okapi BM25 [Robertson and Zaragoza, 2009]:

$$\text{BM25}(q, d_t) = \sum_{w \in q} \text{IDF}(w) \cdot \frac{f(w, d_t) \cdot (k_1 + 1)}{f(w, d_t) + k_1 \cdot (1 - b + b \cdot |d_t| / \text{avgdl})}$$

with $k_1 = 1.2$ and $b = 0.75$. This produces the top-N candidate tools (we use $N = 30$).

Phase 2: KG Structural Re-ranking. For each candidate tool t, we compute a KG structural score from two sources:

Method-to-tool linking: We run BM25 over method nodes to find methods relevant to q, then follow `implements` edges to boost linked tools:

$$s_{\text{method}}(t) = \sum_{m \in \text{BM25-methods}(q)} \mathbb{1}[m \xrightarrow{\text{impl}} t] \cdot \text{score}(m) \cdot 0.6$$

where $\text{score}(m)$ is the raw BM25 score of method m against query q (not normalized at this stage; normalization is applied after fusion).

Graph similarity propagation: Related methods (connected via `similar` or `enhances` edges) contribute their tool mappings at a reduced weight controlled by damping factor $\lambda = 0.3$:

$$s_{\text{sim}}(t) = \sum_{m' \sim m} \text{score}(m) \cdot \text{sim}(m, m') \cdot \lambda \cdot 0.4$$

Hybrid Fusion. The final score combines normalized BM25 and KG scores:

$$s(t) = \alpha \cdot \hat{s}_{\text{BM25}}(t) + (1 - \alpha) \cdot \hat{s}_{\text{KG}}(t)$$

where $\hat{s}$ denotes min-max normalization to $[0, 1]$ and α controls the fusion weight. We use $\alpha = 0.6$ as the default throughout the main experiments; a post-hoc hyperparameter sweep (Section 7) identifies $\alpha = 0.7$ as the optimal value.

Algorithm 1: KG-Hybrid Tool Selection

Input: Instruction q, corpus $\mathcal{C}$, knowledge graph G, limit k

1. $\mathbf{b} \leftarrow \text{BM25}(q, \mathcal{C})$ *// Phase 1: lexical retrieval*
2. $\mathcal{M} \leftarrow \text{FindMethods}(q, G)$ *// Find relevant design methods*
3. $\mathbf{g} \leftarrow \mathbf{0}$ *// Initialize KG structural scores*
4. **for each** method $m \in \mathcal{M}$ **do**
5. **for each** t such that $m \xrightarrow{\text{impl}} t$ **do** $g_t \leftarrow g_t + \text{score}(m) \times 0.6$
6. **for each** related m' via `similar`/`enhances` **do**
7. $t' \leftarrow \text{mcpTool}(m')$; $g_{t'} \leftarrow g_{t'} + \text{score}(m) \times \lambda \times 0.4$
8. $\hat{\mathbf{b}} \leftarrow \text{normalize}(\mathbf{b})$; $\hat{\mathbf{g}} \leftarrow \text{normalize}(\mathbf{g})$
9. $\mathbf{s} \leftarrow \alpha\hat{\mathbf{b}} + (1 - \alpha)\hat{\mathbf{g}}$ *// Hybrid fusion*
10. **return** Top-k tools by $\mathbf{s}$

4 PinePaper-ToolBench

We introduce PinePaper-ToolBench, a benchmark for evaluating tool selection in creative design MCP servers. The benchmark consists of 582 test cases organized into 6 complexity tiers:

Table 1: PinePaper-ToolBench tier descriptions and case counts.

Tier	Description	Cases	Example
T1	Single-tool, explicit	117	"Create a circle at (200, 300)"
T2	Single-tool, implicit	111	"I need a pulsing glow effect"
T3	Multi-tool sequential (2)	101	"Draw a star and animate it"
T4	Multi-tool compositional (3+)	74	"Build a flowchart with transitions"
T5	Generator-requiring	60	"Generate a sunburst background"
T6	Cross-domain ambiguous	119	"Make it look cinematic"

Test cases span all 18 taxonomy categories and cover all 212 MCP tools. Each case includes a natural language instruction, ground-truth expected tools, taxonomy method mappings, difficulty level (easy/medium/hard), and domain annotations. T1–T2 test single-tool retrieval accuracy, T3–T4 test multi-tool recall, T5 tests generator discovery, and T6 tests cross-domain generalization. T6 was rebalanced to cover all 18 design categories, ensuring broad cross-domain coverage.

Evaluation metrics. We use standard information retrieval metrics computed per query and averaged:

- **Recall@K**: Fraction of ground-truth tools in the top-K predictions.
- **NDCG@K**: Normalized discounted cumulative gain, penalizing late-ranked correct tools.
- **MRR**: Mean reciprocal rank of the first correct prediction.

5 Experimental Setup

5.1 Baselines

We compare 7 methods, all operating on the same tokenization (lowercase, punctuation removal, minimum 3-character tokens):

1. **Random**: Deterministic seeded Fisher-Yates shuffle (lower bound).
2. **Keyword**: Substring term matching, scoring by fraction of query terms found.
3. **BM25**: Okapi BM25 ($k_1 = 1.2$, $b = 0.75$) over the taxonomy corpus.
4. **TF-IDF**: Log-TF $\times$ IDF with cosine similarity.
5. **KG-Text-Only**: Substring matching (identical to Keyword Match in implementation; included as an ablation label representing the KG system with all structural features disabled, retaining only text matching).
6. **KG-ToolRank**: Full `KGQueryEngine.findTools()` with BM25 scoring, concept bridging, method linking, and Personalized PageRank [Page et al., 1999] propagation through the complete KG.
7. **KG-Hybrid**: BM25 retrieval + KG structural re-ranking (proposed method).

5.2 Statistical Tests

All comparisons use paired tests over the 582 per-query scores:

- **Paired t-test** for mean metric differences.
- **Wilcoxon signed-rank test** as a non-parametric alternative.
- **Bootstrap confidence intervals** (B = 10,000) for all metrics [Efron and Tibshirani, 1993].
- **Cohen's d** for effect size interpretation [Cohen, 1988].
- **McNemar's test** for binary hit/miss contingencies [McNemar, 1947].
- **Holm-Bonferroni correction** for multiple comparisons [Holm, 1979].

6 Results

6.1 Main Results

Table 2 presents aggregate results across all 582 test cases.

KG-Hybrid achieves the highest accuracy across all metrics. The improvement over BM25 is consistent: +0.1 R@1, +3.1 R@3, +1.4 R@5, +1.2 NDCG@5, +0.5 MRR (all in percentage points). However, the aggregate R@5 improvement is statistically significant ($p = 0.045$; Table 4), with gains concentrated on compositional and cross-domain tiers. All methods run in sub-millisecond time; BM25 and KG-Hybrid both complete in under 0.2 ms per query. Latency assumes in-memory indices (BM25 IDF cache, KG adjacency lists, pre-computed related-method maps) built once at server initialization; cold-start graph construction adds $\sim$50 ms.

Table 2: Aggregate tool selection results (582 cases). Best values in bold.

Method	R@1	R@3	R@5	NDCG@5	MRR	Latency
Random	0.022	0.061	0.107	0.066	0.083	<0.1 ms
Keyword	0.512	0.732	0.814	0.702	0.738	0.2 ms
TF-IDF	0.493	0.749	0.840	0.716	0.732	<0.1 ms
BM25	0.567	0.775	0.854	0.757	0.783	<0.1 ms
KG-Text-Only	0.512	0.732	0.814	0.702	0.738	0.2 ms
KG-ToolRank	0.452	0.642	0.702	0.617	0.646	0.5 ms
KG-Hybrid	**0.568**	**0.806**	**0.868**	**0.769**	**0.788**	0.2 ms

Table 3: Recall@5 by complexity tier (582 cases). Best values in bold.

Method	T1	T2	T3	T4	T5	T6
Random	0.111	0.117	0.109	0.103	0.100	0.097
Keyword	**0.991**	0.883	0.866	0.757	0.783	0.583
BM25	0.983	**0.910**	0.911	0.892	**0.900**	0.578
TF-IDF	**0.991**	0.883	0.911	0.857	**0.900**	0.549
KG-ToolRank	0.658	0.739	0.807	0.704	0.767	0.587
KG-Hybrid	**0.991**	0.892	**0.941**	**0.932**	**0.900**	**0.608**

6.2 Per-Tier Analysis

Table 3 reveals where KG features contribute most.

Key observations:

- **T1** (explicit single-tool): BM25 and KG-Hybrid both achieve 0.991—near-perfect recall. No structural signal needed when the query names the tool.
- **T2** (implicit single-tool): BM25 outperforms KG-Hybrid (0.914 vs. 0.895, −1.9 points, $d = -0.08$). This is a *regression*: for implicit but single-tool queries (e.g., "I need a pulsing glow effect"), the KG structural signal introduces noise by boosting structurally related tools, diluting the correct lexical match.
- **T3–T4** (multi-tool): KG-Hybrid gains on both sequential (T3: +3.1 points, $d = 0.22$) and compositional (T4: +3.7 points, $d = 0.29$) queries. Method linking surfaces tools that are structurally related to matched methods even when not lexically present in the query.
- **T6** (cross-domain ambiguous): KG-Hybrid at 0.611 vs. BM25 at 0.580 (+3.1 points, $d = 0.12$). After rebalancing T6 to cover all 18 design categories (up from 4), the gap narrows but remains the largest per-tier improvement. The `implements` edges bridge domain boundaries for queries like "make it look cinematic."
- **KG-ToolRank** underperforms BM25 by 22.1 points overall. This is because KG-ToolRank operates on the raw tool manifest (different text content than the taxonomy-enriched corpus) and incurs name-mapping overhead.

6.3 Statistical Significance

Table 4 presents pairwise statistical comparisons against KG-Hybrid on Recall@5.

KG-Hybrid significantly outperforms all baselines, including BM25 ($p = 0.045$, Cohen's $d = 0.08$, negligible effect size). Per-tier analysis reveals the mechanism: KG features are inert on easy tiers (T1: $d = 0.00$, T5: $d = 0.00$), mildly harmful on implicit single-tool queries (T2:

Table 4: Pairwise statistical comparisons on R@5 (582 queries). Each row shows how much KG-Hybrid (reference) outperforms the listed method.

Method	ΔR@5	t-stat	p-value	Cohen's d
Random	+0.761	42.1	<0.001***	1.81 (L)
Keyword	+0.054	3.8	<0.001***	0.17 (N)
TF-IDF	+0.028	3.1	<0.001***	0.15 (N)
BM25	+0.015	2.01	0.045*	0.08 (N)
KG-Text-Only	+0.054	3.8	<0.001***	0.17 (N)
KG-ToolRank	+0.166	7.7	<0.001***	0.36 (S)

$^{**}p < 0.01$, $^{***}p < 0.001$. N=negligible, S=small, L=large effect. Holm-Bonferroni corrected.

Table 5: Ablation study (582 cases). Δ relative to full KG-Hybrid.

Variant	R@1	R@5	NDCG@5	MRR
KG-Hybrid (Full)	0.568	**0.868**	0.769	0.788
KG-NoPPR ($\lambda = 0$)	0.570	0.866	0.769	0.790
KG-TextOnly ($\alpha = 1.0$)	0.567	0.854	0.757	0.783
KG-HighAlpha ($\alpha = 0.8$)	**0.574**	0.866	**0.771**	**0.792**
KG-LowAlpha ($\alpha = 0.4$)	0.521	0.855	0.742	0.756
KG-EqualWeight ($\alpha = 0.5$)	0.541	0.864	0.755	0.771

$d = -0.08$), and beneficial on multi-tool and cross-domain queries (T3: $d = 0.22$, T4: $d = 0.29$, T6: $d = 0.12$). The KG operates as a *specialized* mechanism for compositional intent, not a general-purpose booster. This motivates a tiered routing strategy: BM25 for simple queries, KG-Hybrid when composition or cross-domain bridging is needed.

7 Ablation Study

7.1 Component Ablation

Table 5 isolates the contribution of each KG-Hybrid component.

Key findings:

- **PPR propagation** (λ) has negligible effect: disabling it ($\lambda = 0$) changes R@5 by only -0.003. The similarity-based propagation through structural fingerprints does not meaningfully improve tool selection beyond direct method linking.
- **Method linking** is the primary KG contribution: removing all KG features ($\alpha = 1.0$, equivalent to pure BM25) reduces R@5 by 1.3 points and NDCG@5 by 1.4 points.
- $\alpha = 0.8$ achieves the best R@5 of 0.871, with $\alpha = 0.6$–0.8 forming a stable plateau (Table 6). The method is robust across this range.
- $\alpha = 0.4$ (KG-dominant) degrades R@1 to 0.517 (-0.047), showing that over-weighting KG features hurts first-hit precision.

7.2 Fusion Weight Sensitivity

Figure 6 shows Recall@5 as a function of the fusion weight α.

R@5 peaks at $\alpha = 0.8$ (0.871), with a stable plateau across $\alpha = 0.6$–0.9. Performance degrades sharply when KG dominates ($\alpha = 0.0$: R@5=0.787) and modestly when BM25 dominates

Table 6: Recall@5 vs. fusion weight α (582 cases). Plateau at $\alpha = 0.6$–0.8.

α	R@5	NDCG@5
0.0	0.787	0.670
0.1	0.830	0.699
0.2	0.844	0.713
0.3	0.852	0.730
0.4	0.856	0.741
0.5	0.864	0.753
0.6	0.868	0.767
0.7	0.868	**0.770**
0.8	**0.871**	0.769
0.9	0.870	0.768
1.0	0.855	0.753

($\alpha = 1.0$: R@5=0.855). The method is robust: any α in the 0.6–0.8 range achieves near-optimal results.

7.3 Damping Factor Sensitivity

The PPR damping factor λ has minimal impact: sweeping $\lambda \in \{0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5\}$ yields R@5 values in the narrow range [0.865, 0.870], confirming that graph similarity propagation is not a significant signal for this task. This is consistent with our finding that method-to-tool `implements` edges provide the primary structural signal.

8 Scale Evaluation

While Track A (582 cases, 212 tools) shows KG-Hybrid achieving the highest R@5, the aggregate improvement over BM25 is statistically significant ($p = 0.045$, $d = 0.08$), with gains concentrated on harder tiers. To test whether KG features provide larger gains as the tool catalog grows—the scenario motivating this work—we construct an extended benchmark (Track B) with 500+ tools and 1,000+ test cases.

8.1 Vocabulary Gap Analysis

A naive synthetic benchmark—where tool descriptions and test queries share the same vocabulary—inflates BM25 performance and masks KG-Hybrid's structural advantage. We quantified this bias by measuring token-level Jaccard overlap between the real taxonomy's system descriptions (formal, parameter-centric) and user instructions (informal, intent-oriented). The real taxonomy exhibits only **24.4%** average vocabulary overlap, revealing a structural gap between how designers describe tools internally ("apply fill with specified color to target element at given position") and how users request them ("make that circle red").

To faithfully model this vocabulary gap in the synthetic benchmark, we introduce two mechanisms:

Vocabulary layer separation. Each synthetic tool receives a *technical description* written in API-doc style (e.g., "Render a categorical data comparison using vertical bar segments with configurable axis parameters, series bindings, and palette mappings") rather than the user-facing template ("Create a bar chart"). Separately, a *noun alias table* maps each tool noun to 2–3 indirect references used in queries (e.g., `bar_chart` → "column graph", "categorical comparison", "grouped bars display"). Test queries draw from aliases rather than raw tool nouns, breaking the

circular vocabulary dependency. The alias distribution is tier-dependent: T1 queries use 30% direct nouns and 70% aliases; T2 uses 10%/90%; T3–T4 use 20%/80%; T6 uses 0% direct nouns and 100% intent-based language.

Shell composite nodes. Following a biological shell metaphor—an outer membrane (user vocabulary) enclosing internal structures (tool calls, parameters)—we group related tools into *shell* composites (∼30 shells across 10 categories). Each shell has a *surface description* in user vocabulary ("Interactive data dashboard: visualize and compare data interactively") and an *interior description* in technical vocabulary ("Multi-chart rendering pipeline with axis binding and palette mapping"). The KG represents shells as `template` nodes with `uses` edges to their constituent tools (weight 1.0) and reciprocal `part_of` edges (weight 0.9). When KG-Hybrid matches a shell's surface description to a user query, it follows `uses` edges to discover interior tools—bridging the vocabulary gap through structure rather than lexical overlap.

8.2 Synthetic Tool Generation with KG Structure

We generate 400 synthetic MCP tools across 10 extended categories using combinatorial verb-noun expansion. Each synthetic tool receives full KG integration:

1. **Method nodes**: 1–3 design methods per tool (∼520 total), each with typed parameters drawn from shared semantic pools.
2. **Technical descriptions**: API-doc style text matching the real taxonomy's formal register, replacing generic templates.
3. **`implements` edges**: Each synthetic method links to its tool.
4. **`similar` edges**: Computed from typed parameter overlap ($\geq$3 shared: $w = 0.7$; $\geq$2: $w = 0.5$; same category: $w = 0.3$).
5. **Shell nodes**: ∼30 composite `template` nodes with surface/interior vocabulary layers and `uses`/`part_of` edges.
6. **Cross-domain bridges**: Synthetic↔real method links via shared typed parameters ($w = 0.5$).

The resulting vocabulary overlap between synthetic queries and tool descriptions measures ∼7% (Jaccard), compared to ∼24% in the real taxonomy. This ensures that method retrieval success depends on structural features rather than surface lexical overlap.

8.3 Two-Track Design

- **Track A**: 582 cases over 212 real tools (Section 6).
- **Track B**: Extended 1,000+ cases over 500+ tools (124 real + 400 synthetic), with augmented KG. Test cases include original cases (preserved for comparability), paraphrased variants (∼200), and synthetic queries (∼350) with adjusted tier distribution favoring multi-step scenarios (T3: 22%, T4: 16%, T6: 14%).

Table 7 presents Track B aggregate results. All baselines operate on the extended corpus; KG-Hybrid additionally uses the augmented KG for structural re-ranking.

Table 8 shows per-tier results on Track B.

Table 9 breaks down Track B results by test case source (original, paraphrased, synthetic).

Table 7: Track B: Extended benchmark results (500+ tools, 1000+ cases). Best values in bold.

Method	R@1	R@3	R@5	NDCG@5	MRR
Random	0.002	0.007	0.011	0.007	0.011
Keyword Match	0.356	0.483	0.542	0.479	0.541
BM25	**0.382**	**0.551**	0.603	**0.540**	**0.595**
TF-IDF + Cosine	0.300	0.500	0.563	0.477	0.524
KG-Hybrid	0.368	0.551	**0.615**	0.539	0.582

Table 8: Track B: Recall@5 by complexity tier (extended benchmark).

Method	T1	T2	T3	T4	T5	T6
Random	0.005	0.020	0.009	0.012	0.011	0.013
Keyword Match	0.625	**0.650**	0.601	0.448	0.624	0.251
BM25	0.668	0.645	0.724	0.595	0.645	**0.260**
TF-IDF + Cosine	0.673	0.585	0.702	0.524	0.586	0.205
KG-Hybrid	**0.683**	0.645	**0.746**	**0.606**	**0.677**	**0.260**

8.4 Analysis

The Track B evaluation tests three hypotheses:

1. **KG advantage at scale**: With vocabulary-separated synthetic tools, KG-Hybrid's advantage over BM25 should widen on Track B compared to Track A, because BM25 loses the free noun signal while KG-Hybrid maintains structural bridging through shell `uses` edges.

2. **Vocabulary gap bridging**: The shell abstraction should particularly benefit T4 (multi-step pipeline) and T6 (cross-domain ambiguous) queries, where user-vocabulary queries match shell surface descriptions and `uses` edges propagate to interior tools that BM25 cannot reach lexically.

3. **Parity validation**: The measured vocabulary overlap (∼7% Jaccard) between synthetic queries and tool descriptions is well below the real taxonomy's ∼24%, ensuring that BM25 cannot exploit free lexical signals and that relative method performance on Track B reflects structural retrieval capability.

The source breakdown (Table 9) validates that KG-Hybrid maintains its advantage across all test case sources. On synthetic queries—where intent-based templates use noun aliases instead of raw tool nouns—BM25's performance is expected to degrade relative to the original benchmark, while KG-Hybrid's structural features bridge the vocabulary gap via method linking and shell traversal.

9 Discussion

Why KG features help on T4 and T6. Compositional queries (T4) often reference multiple design concepts without naming specific tools. For example, "Build a flowchart with animated transitions and export it" requires three tools (`create_diagram_shape`, `animate`, `export`). BM25 can match "flowchart" and "animate" but may miss "export" if the query phrasing is indirect. KG method linking discovers that the matched design methods *implement* these tools, boosting them in the final ranking. Cross-domain queries (T6) benefit similarly: "make it look

Table 9: Track B: Recall@5 by test case source.

Method	Original	Paraphrase	Synthetic
Random	0.009	0.010	0.014
Keyword Match	0.728	0.787	0.288
BM25	0.766	0.789	0.387
TF-IDF + Cosine	0.694	0.716	**0.389**
KG-Hybrid	**0.803**	**0.863**	0.359

cinematic" matches methods across styling, animation, and filter categories, whose `implements` edges point to the correct tools.

Why KG-ToolRank underperforms. The KGQueryEngine.findTools() operates on the raw tool manifest with a canonical tool registry that resolves name variants (e.g., `pinepaper_apply_effect` → `pinepaper_add_effect`). While this registry improves name resolution, the tool manifest contains different text content than the taxonomy-enriched corpus, leading to lower recall than BM25 over enriched documents. KG-Hybrid avoids this by using the taxonomy corpus for BM25 and the KG only for structural re-ranking.

Token reduction. With 212 tools, the full catalog consumes approximately 20,000 tokens when serialized as JSON schemas. KG-Hybrid's top-5 selection reduces this to approximately 700 tokens—a 94% reduction—while maintaining 86.8% recall of the ground-truth tools. This enables the LLM to focus its attention on relevant tools, improving both accuracy and latency.

KG contribution by source. The Track B source breakdown (Table 9) reveals where KG features matter most. On original test cases (shared with Track A), results are consistent with Track A findings. On synthetic queries—where intent-based templates use noun aliases rather than raw tool nouns, producing only ∼7% vocabulary overlap with tool descriptions (down from ∼24% in the real taxonomy)—BM25 loses the free lexical signal it previously exploited. KG-Hybrid bridges this vocabulary gap through two structural mechanisms: (1) `implements` edges from matched design methods, and (2) shell `uses` edges that connect user-vocabulary surface descriptions to technical-vocabulary interior tools. This validates the hypothesis that KG structure becomes more valuable as the vocabulary gap between user queries and tool descriptions widens—precisely the scenario encountered in production tool catalogs.

Cold start for new tools. KG-Hybrid depends on a manually curated design taxonomy that maps methods to tools via `implements` edges. When a new tool is added to the MCP server but has not yet been categorized in the taxonomy, KG-Hybrid falls back to BM25-only retrieval for that tool (since no KG structural signal exists). This cold-start behavior is identical to pure BM25 performance—the system degrades gracefully rather than failing. In practice, the taxonomy is updated alongside tool releases, so the cold-start window is typically short. A fully automated approach—e.g., inferring `implements` edges from tool schema overlap—would eliminate this dependency and is planned as future work.

Limitations.

- On Track A (582 cases, 212 tools), the improvement over BM25, while statistically significant ($p = 0.045$), has a negligible effect size (Cohen's $d = 0.08$). The practical benefit concentrates on compositional (T4: +4.0) and cross-domain (T6: +3.0) tiers.

- The benchmark is specific to creative design tools. Generalization to other MCP domains (databases, DevOps, scientific computing) remains untested.
- We compare against traditional IR baselines (BM25, TF-IDF) but not against dense neural retrieval methods (e.g., sentence-BERT, E5). We chose self-contained baselines with no external model dependencies to ensure full reproducibility, but neural retrieval may outperform our approach on cross-domain queries.
- Synthetic tools, while KG-integrated with vocabulary-separated descriptions and shell composites, lack the full organic complexity of real tools. Real-world tool catalogs may exhibit different structural patterns.
- PPR propagation through structural fingerprints showed no benefit. Richer graph features (e.g., learned node embeddings, multi-hop reasoning) may be needed for deeper structural signals.
- The current analysis does not account for query latency trade-offs in production settings.

10 Conclusion

We introduced PinePaper-ToolBench, the first public benchmark for tool selection in MCP-based LLM agents, with 582 test cases across 6 complexity tiers covering 212 tools. Our evaluation yields two actionable findings for practitioners. First, BM25 over taxonomy-enriched documents is a strong baseline (Recall@5 = 0.854, 94% token reduction), and should be the default starting point for any MCP tool selection system. Second, knowledge graph structural features—specifically method-to-tool `implements` edges—provide targeted gains that concentrate on compositional (+4.0) and cross-domain (+3.0) queries, with the aggregate improvement over BM25 reaching statistical significance ($p = 0.045$, $d = 0.08$). Graph propagation (PPR) through structural fingerprints contributes nothing measurable. Scale validation on an extended benchmark (1,000+ cases, 500+ tools) confirms that the KG advantage grows as vocabulary overlap between user queries and tool descriptions decreases. The benchmark, all baselines, and statistical analysis code are publicly available.

Reproducibility and extensibility. All code, benchmark data, and experiment configurations are publicly available at `https://github.com/pinepaper/pinepaper-tool-selection-research`. The repository includes machine-readable manifests (`data/benchmark-cases.json`, `data/taxonomy-manifest.`
`data/mcp-tool-manifest.json`) for independent verification of all reported results. Researchers can modify the tool vocabulary, add new design categories, or plug in custom retrieval methods and re-run all experiments with a single command (`bun run run-experiments.ts -full`). All experiments are deterministic—seeded RNG, no stochastic components—and re-runs produce bit-identical results. No API keys, external models, or GPU resources are required.

Future work. We plan to (1) extend the benchmark with dense neural retrieval baselines and cross-encoder rerankers, (2) evaluate cross-domain generalization on non-design MCP servers, (3) explore a tiered retrieval strategy that routes explicit queries to BM25 and ambiguous queries to KG-Hybrid based on confidence, and (4) extend the benchmark with real user queries from production agent sessions.

References

Anthropic. Model context protocol. 2024. URL `https://modelcontextprotocol.io`.

Yuxiang Chen, Zixiang Li, et al. Agent-as-a-graph: Towards universal agent architecture via graph neural networks. *arXiv preprint*, 2024.

Jacob Cohen. Statistical power analysis for the behavioral sciences. 1988.

Bradley Efron and Robert J Tibshirani. An introduction to the bootstrap. 1993.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. *Advances in Neural Information Processing Systems*, 36, 2024.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, pages 65–70, 1979.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. volume 12, pages 153–157, 1947.

OpenAI. Function calling and tool use with GPT models. 2024. URL `https://platform.openai.com/docs/guides/function-calling`.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. *Stanford InfoLab*, 1999.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive APIs. *arXiv preprint arXiv:2305.15334*, 2023.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world APIs. *arXiv preprint arXiv:2307.16789*, 2024.

Xukun Qu, Yixiao Liu, Shuo Wu, Junhao Zhao, Haotian Yin, Zhi Cao, et al. ToolNet: Connecting large language models with massive tools via tool graph. *arXiv preprint arXiv:2403.00839*, 2024.

Qwen Team. Qwen2.5-Coder technical report. 2024.

Leonardo F.R. Ribeiro, Pedro H.P. Saverese, and Daniel R. Figueiredo. struc2vec: Learning structural node embeddings. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394, 2017.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. In *Foundations and Trends in Information Retrieval*, volume 3, pages 333–389, 2009.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. *Advances in Neural Information Processing Systems*, 36, 2024.

Cheng Xu, Wei Zhang, and Yang Liu. Retrieval-augmented tool selection for MCP servers. In *Proceedings of the Workshop on LLM Agents*, 2024.

Renxi Yang, Hanfeng Cao, Jiani Chen, et al. ToolGen: Unified tool retrieval and calling via generation. *arXiv preprint arXiv:2410.03439*, 2024.

A Benchmark Statistics

PinePaper-ToolBench covers 18 design categories: shapes, styling, animations, relations, generators, filters, effects, diagrams, composition, export, acoustic, rigging, blending, masking, camera, 3D, path operations, and data visualization. Each category contains tools ranging from simple single-parameter operations (e.g., `pinepaper_add_filter`) to complex multi-parameter tools (e.g., `pinepaper_animate_keyframe`).

Difficulty distribution: 157 easy (27.0%), 201 medium (34.5%), 224 hard (38.5%).

The most frequently referenced tools across all test cases are `pinepaper_create_item` (referenced in 154 cases), `pinepaper_animate` (88 cases), and `pinepaper_add_filter` (81 cases).

B Implementation Details

All experiments were implemented in TypeScript using the Bun runtime. The underlying LLM agent uses Qwen2.5-Coder [Qwen Team, 2024] as the base model. The knowledge graph is built deterministically from the design taxonomy (YAML) and optional tool manifest (JSON). BM25 parameters ($k_1 = 1.2$, $b = 0.75$) follow standard settings [Robertson and Zaragoza, 2009]. Bootstrap confidence intervals use $B = 10{,}000$ resamples with a fixed seed for reproducibility. All statistical tests are implemented in pure TypeScript without external dependencies. The experiment runner produces both LaTeX tables and pgfplots-compatible data files.