MCP Servers for Pyserini and RankLLM: Enabling Agentic Retrieval-Augmented Generation

Yijun Ge
University of Waterloo
David R. Cheriton School of Computer Science
Waterloo, Ontario, Canada
l2ge@uwaterloo.ca

Zibo Guo
University of Waterloo
David R. Cheriton School of Computer Science
Waterloo, Ontario, Canada
d6guo@uwaterloo.ca

Sahel Sharifymoghaddam
University of Waterloo
David R. Cheriton School of Computer Science
Waterloo, Ontario, Canada
sahel.sharifymoghaddam@uwaterloo.ca

Jimmy Lin
University of Waterloo
David R. Cheriton School of Computer Science
Waterloo, Ontario, Canada
jimmylin@uwaterloo.ca

Abstract

Large language models (LLMs) increasingly rely on retrieval-augmented generation (RAG), where search results are provided as external context to improve grounding and factuality. The Model Context Protocol (MCP) is an emerging industry standard for connecting LLMs to external tools. We present MCP servers for Pyserini and RankLLM, two widely used information retrieval research toolkits, enabling their retrieval and reranking capabilities to be easily accessed by LLM agents. Our system exposes sparse and dense retrieval over prebuilt indexes, multimodal search, LLM-based reranking, and evaluation based on relevance judgments, all as tools callable by LLMs. This integration supports rapid experimentation with agentic RAG workflows and facilitates comparisons across models and clients. We demonstrate end-to-end retrieval, reranking, RAG, and multimodal search using both open-source and proprietary LLMs, bridging IR research infrastructure with LLM tool ecosystems.

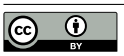
CCS Concepts

• **Information systems → Users and interactive retrieval**.

Keywords

Information Retrieval, Model Context Protocol, Retrieval-Augmented Generation, Reranking, Agents, Pyserini, RankLLM

ACM Reference Format:
Yijun Ge, Zibo Guo, Sahel Sharifymoghaddam, and Jimmy Lin. 2026. MCP Servers for Pyserini and RankLLM: Enabling Agentic Retrieval-Augmented Generation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26), July 20–24, 2026, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3805712.3808376



SIGIR '26, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-2599-9/2026/07
https://doi.org/10.1145/3805712.3808376

1 Introduction

As large language models (LLMs) continue to advance, retrieval-augmented generation (RAG) [16, 20, 34] has emerged as a dominant use case for search technology, incorporating search results as external context to improve grounding. Furthermore, LLMs are shifting to focus on better agentic capabilities [9], where RAG is essential to ensure that LLMs make informed choices. An important feature for LLM agents is the ability to make tool calls, and the Model Context Protocol (MCP) [3] has emerged as the standard for connecting LLMs to tools: it has been adopted by model builders and providers to allow models to easily connect to external tools through MCP servers, and by enterprises to build MCP servers to expose tools specifically for these LLMs [8, 10, 19, 30]. Search, for powering RAG, is an excellent example of a tool that one might find in an MCP server for LLMs.

To better address this new use case of search, agentic RAG, we introduce MCP servers to Pyserini [23] and RankLLM [36], widely-used Python toolkits for information retrieval (IR) research; we have been building toward this release since June 2025,[1] and we now share the full system with the community. Pyserini supports many first-stage retrieval modes and common datasets with prebuilt indexes, topics, and relevance judgments [23]. RankLLM provides a simple interface to a wide selection of LLM rerankers that improve the quality of the final ranked list of results [18, 26, 32, 37]. By integrating MCP servers into these packages, we make it easier for researchers and practitioners to build IR methods for RAG. MCP's portability across LLMs also enables controlled model comparisons on benchmarks such as BrowseComp [41]. To the best of our knowledge, we are the first open-source IR research toolkit supporting MCP-based RAG.

Our contribution is a pair of MCP servers for Pyserini and RankLLM, exposing state-of-the-art retrieval and reranking as tools for LLMs to empower RAG, to bridge the gap between research toolkits and industry standards in agentic retrieval and help researchers and developers build better retrieval-powered agentic systems. We hope our system leads to the advancement of stronger RAG capabilities to power smarter agents.

[1] https://x.com/lintool/status/1929556643961840040

```python
@mcp.tool()
def search(
    query: str | Dict[str, Any],
    index: str = "msmarco-v2.1-doc-segmented",
    hits: int = 10,
    parse: bool = True,
    ef_search: int = 100,
    encoder: str = "",
    query_generator: str = ""
):
    """
    Search the Pyserini index with the appropriate method for
↪ the type of the index provided and return top-k hits.
    [...]

    Args:
        query: Search query [...]
        index: Name of index to search [...]
            Default is msmarco-v2.1-doc-segmented
            which is good for retrieval augmented
            generation for LLMs.
        hits: Number of results to return (default: 10)
        [...]

    Returns:
        List of search results with docid, score, and raw
↪ contents in text or image form
    """
```

Figure 1: Search tool from Pyserini's MCP server.

2 MCP Servers

Our MCP servers are built with the FastMCP framework [17] and follow a modular design. Pyserini's MCP server provides tools for retrieval, index and document interaction, and evaluation. RankLLM's MCP server includes all Pyserini tools and adds reranking capabilities. Pyserini's server therefore covers first-stage retrieval, while RankLLM's server is a superset that extends all Pyserini tools with second-stage reranking. The servers are easy to install and run as part of the Pyserini and RankLLM packages, using their standard installation guides.[2][3]

An example tool definition exposed to the LLM is shown in Figure 1. The Python function docstring serves as the tool description, providing detailed guidance on tool behavior and parameters; a shortened version is shown in the figure. Given a user request, the LLM extracts argument values, calls the MCP tool, and receives the tool output on completion. The remainder of this section describes tools across each stage of the multi-stage Pyserini and RankLLM retrieval pipeline. For detailed documentation of each stage, we refer readers to the corresponding GitHub repositories.

2.1 Pyserini

Pyserini's MCP server exposes tools for first-stage retrieval, index interaction, and evaluation.

2.1.1 ***Retrieval.*** We provide LLMs with access to Pyserini's comprehensive modes of retrieval: lexical BM25 retrieval on Lucene inverted indexes [35], dense retrieval with Lucene and Faiss flat and HNSW vector indexes [14, 27], and learned sparse retrieval [22] with Lucene impact indexes [11].

Search. Since the same tool is used for text and image retrieval, the query can either be a string for a text query or a dictionary with a local path or URL to a query image. The index name is a string referring to a prebuilt Pyserini index, a local index path, or a configuration-defined alias. Prebuilt indexes are resolved against the registry to determine the index type and initialize the appropriate searcher. Local paths default to Lucene inverted indexes, while local indexes of other types are supported via aliases read from a configuration file that specifies the path, index type, and other options like the encoder model for dense retrieval. The most relevant results are returned with their raw contents for easy RAG, supporting both text documents and images.

Fusion. Fusing results from different retrieval methods is an easy way to improve the final ranked list of results [21]. This tool takes two ranked lists of search results and fuses them by averaging the normalized score of every document.

2.1.2 ***Inspection.*** Pyserini offers a large collection of prebuilt indexes, and its MCP server provides tools to interact with them and the raw documents they were derived from.

List Indexes. Pyserini supports various types of indexes, such as Lucene inverted indexes, Lucene flat and HNSW vector indexes [27], Lucene impact indexes, and Faiss indexes [14]. Given the desired type, the tool returns the names of all prebuilt indexes of that type available in Pyserini. This is useful for the LLM to check against the acceptable index names to pass as an argument into other tools or even autonomously select an index to retrieve from.

Get Index. Given the name of an index, the tool returns the metadata for that index and whether it is currently downloaded where the MCP server is running. This can work with the list-indexes tool for easy index interactions and management for a locally-running MCP server.

Get Document. Given the name of an index and the document identifier of a document in that index, the tool returns the raw contents of the document. It allows the user to supply their own retrieval results in the form of document identifiers to use with other tools, or simply for dynamic interactions with documents.

2.1.3 ***Evaluation.*** With search capabilities, it is important to be able to evaluate retrieved results as well. Alongside its prebuilt indexes, Pyserini includes the corresponding relevance judgments, exposed through the following tools.

Get Qrels. This tool returns the query relevance judgments for a given query identifier and an index. With full transparency, users can clearly see which documents are relevant to the query.

Evaluate. Given the index name corresponding to the relevance judgments, the query identifier for which the search was performed, the retrieved results, the evaluation metric, and the cutoff for the number of results to evaluate, this tool runs TREC [40] evaluation and returns the score. This allows users to see the "official" evaluation of retrieval results.

[2] pyserini.io
[3] rankllm.ai

```
Tool Call: search
{
  "query": {
    "qid": "1",
    "query_txt": "sports inclusion cultural influence"
  },
  "hits": 10
}

Result
[
  "Query Results for: sports inclusion cultural influence",
  "DocID: msmarco_v2.1_doc_25_12245309... | Score: 12.3521",
  "Not only is it important for individuals to develop their own
  ↪ cultural awareness, understanding, and ...",
  "DocID: msmarco_v2.1_doc_25_12245309... | Score: 12.3268",
  "Although the participation of girls and women has increased
  ↪ dramatically in recent years, stereotypes ..."
   ... more results omitted ...
]
```

Figure 2: Example LLM call to the search tool and results.

2.2 RankLLM

RankLLM's MCP server offers two new tools: "retrieve-and-rerank" and "rerank".

Retrieve and Rerank. RankLLM's built-in Pyserini retriever is used to retrieve results for a given query and index, which are reranked with the specified LLM and configuration, and the final list of reranked results with raw contents is returned. The available parameters take full advantage of RankLLM's wide range of supported rerankers [36]. When reranking is desired, this tool is preferred over separate retrieval and reranking calls in order to conserve context window space and reduce token costs, and the tool description instructs the LLM accordingly. As RAG and reranking both involve raw documents, the arguments and return values of tools can quickly consume a lot of tokens. Retrieving search results into the context, calling a separate reranking tool, and accepting the return value of the reranking tool takes roughly three times as many tokens as simply calling the tool to retrieve and rerank, which only adds the final reranked results to the context.

Rerank. Nevertheless, we provide a rerank tool for when the user wants to work with already existing retrieval results. This tool takes a list of candidates to rerank, the query text for the reranker to compare relevance to, and the same reranker configuration parameters as the retrieve-and-rerank tool.

3 Demonstration

To demonstrate transferability, we use two model–client settings: one open-source and one proprietary. For the open-source setting, we serve OpenAI's gpt-oss-20b [1] locally with vLLM [39] and connect through a lightweight client script built with Pydantic AI [33]; the script is publicly available in the Pyserini repository, although it does not depend on Pyserini. For the proprietary setting, we use Claude Sonnet 4.5 [6] with Cursor, a popular LLM-powered code editor with strong support for MCP servers [7]. We select these models as representatives of leading providers, given their strong general performance and tool-use capabilities [1, 6]. First,

```
Tool Call: retrieve_and_rerank
{
  "query": "sports societal impact athlete compensation inclusion
  ↪ cultural influence business side equipm...",
  "top_k_candidates": 20
}

Result
[
  {
    "query": {
      "text": "sports societal impact athlete...",
      "qid": "1"
    },
    "candidates": [
      {
        "docid": "msmarco_v2.1_doc_30_642952283...",
        "score": 20.174,
        "doc": {
          "docid": "msmarco_v2.1_doc_30_642952283...",
          "url": "https://tonyevans.org/about/the-urban-alternative/
  ↪ ",
          "title": "About The Urban Alternative...",
          "headings": "About The Urban Alternative...",
          "segment": "Athlete's Impact (AI) exists as an outreach
  ↪ both into and through the sports arena...",
        }
      }
       ... more candidates omitted ...
    ]
  }
]
```

Figure 3: Example LLM call to the retrieve-and-rerank tool and returned results.

we showcase RAG capabilities with the search tool and the retrieve-and-rerank tool, then we conduct multimodal retrieval by searching for similar images to the one provided.

We note that the MCP servers do not introduce new retrieval or reranking capabilities; they expose existing Pyserini and RankLLM functionality as MCP tools, so retrieval effectiveness is inherited directly from the underlying methods and requires no separate evaluation. Beyond effectiveness, a key motivation for this work is alignment: by adopting MCP as a common industry standard, IR research infrastructure becomes directly accessible within the same agentic workflows used in practice, closing the gap between research toolkits and real-world deployments and enabling seamless end-to-end experiences for users.

3.1 RAG

We use gpt-oss-20b with our local MCP client to demonstrate RAG. The query provided to the LLM is shown in Figure 4 and is drawn from TREC RAG 2025 [38], specifically topic ID 14. We append the instruction "Please do deep research" to encourage the model to fully utilize the available retrieval tools.

As shown in Figure 4, the model responds by invoking the search tool four times and the retrieve-and-rerank tool once, retrieving multiple documents to inform its response. Examples of the tool calls and results generated by the LLM are shown in Figures 2 and 3. Notably, the generated retrieval queries include relevant terms inferred from the user's prompt, resembling automatic query

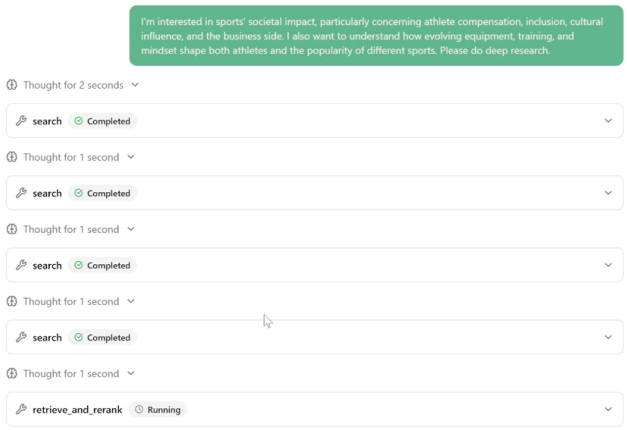


Figure 4: LLM performing deep research for RAG with MCP tools by calling various tools multiple times.

5. The macro-picture: Society, sport, economy

1. **Economic engine** – Sports sectors contribute >US$500 bn to the global economy, driven largely by broadcasting rights, sponsorship, and merch. (Document: *msmarco_v2.1_doc_07_876454531.*)

2. **Cultural mirror** – The demographics of the player base influence the norms, from dress codes to language; inclusive policies now convert marginalised groups into mainstream audiences (docs: *msmarco_v2.1_doc_25_1224530952, msmarco_v2.1_doc_34_1178689875*).

3. **Innovation seed** – Equipment advances, AI-assisted training, and data-powered narratives create new forms of entertainment (docs: *msmarco_v2.1_doc_44_901207583, msmarco_v2.1_doc_53_27098295*).

4. **Human capital** – The rise of athlete-to-coach pipelines (sports education, team management) feeds an emerging sports-management workforce. (Docs: *msmarco_v2.1_doc_08_200552944* – skills required).

Figure 5: LLM incorporating retrieval results into its response and providing inline citations.

expansion by the LLM, a commonly used technique to improve retrieval effectiveness [12].

Figure 2 shows a call to the search tool generated by the LLM, with the query text and the number of hits to return, and an example of the first few results returned by the search tool, with document identifiers, relevance scores, and raw document contents. Similarly, Figure 3 shows an LLM's call to the retrieve-and-rerank tool with the query, the desired number of candidates, and the beginning of the list of results returned by the tool, specifically, the first hit. Finally, Figure 5 shows a snippet of the LLM's consolidated retrieval-augmented response, referencing the retrieved documents with inline citations.

3.2 Multimodal Retrieval

We use Cursor with Claude Sonnet 4.5 to showcase multimodal retrieval. To demonstrate generalizability, we use a random image from the internet, a web image returned for the query "dog". Figure 6 shows the prompt given to the LLM, the target image, the tool call generated by the LLM, one of the result images returned, and a snippet of the final model response. The query here is a dictionary with the URL to the desired image.

Notably, without any textual description, Claude Sonnet 4.5 identified the dogs as golden retrievers or Labradors, demonstrating that the client model actively interprets visual content rather than

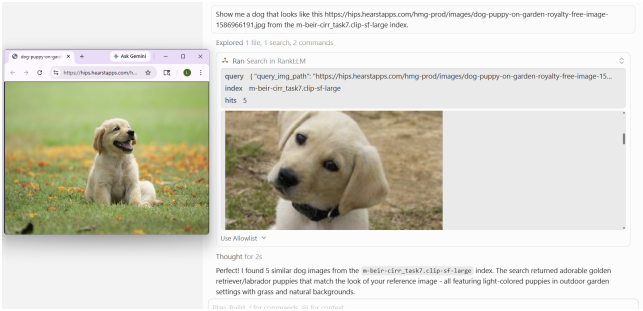


Figure 6: LLM performing multimodal retrieval using the MCP server with a query image.

treating it as an opaque reference. This suggests a straightforward path toward multimodal RAG, where retrieved images could serve as grounding context for vision-capable LLMs responding to image-based queries.

4 Related Work

Retrieval-Augmented Generation. RAG [20] augments LLM generation with externally retrieved evidence, grounding model outputs in up-to-date, verifiable information rather than parametric memory alone. As LLMs have grown more capable and widely deployed, RAG has emerged as a dominant architecture for knowledge-intensive tasks, replacing static fine-tuning with dynamic retrieval that keeps pace with evolving corpora [16]. Agents further extend this paradigm: they autonomously decide when and what to retrieve, iteratively refining their queries in multi-step reasoning loops [9]. Our system targets this agentic RAG setting, exposing research-grade retrieval and reranking infrastructure as MCP tools so that agents can invoke them natively.

Agentic RAG Frameworks. Several open-source frameworks have lowered the barrier to building agentic RAG pipelines by providing reusable abstractions over LLM calls, retrieval backends, and tool orchestration. LangChain [13] offers a composable, chain-based architecture with an extensive integration ecosystem, making it well suited for rapid prototyping of complex, tool-augmented workflows. LlamaIndex [24] takes a retrieval-first approach, providing purpose-built data connectors, indexing primitives, and query engines optimized for RAG over large, heterogeneous document collections. Haystack [31] emphasizes production readiness through explicit, graph-structured pipelines with strong evaluation and observability tooling, and is widely used in enterprise search settings. All three frameworks provide general-purpose scaffolding for connecting LLMs to data; they are complementary to, and can make use of, our MCP servers. By contrast, our contribution is lower in the stack: we expose Pyserini and RankLLM as MCP tools, making state-of-the-art sparse and dense retrieval, learned sparse retrieval, and LLM-based reranking available to *any* MCP-compatible client or framework without additional integration work.

Model Context Protocol and Alternatives. Released by Anthropic in November 2024, MCP [3] defines a vendor-neutral, JSON-RPC-based standard for connecting LLM clients to external tools and

data sources. It has seen rapid industry adoption across agentic coding environments such as Cursor and Claude Code, as well as major model providers including OpenAI, Google, and Microsoft [4]. Within the information retrieval space, dedicated search providers such as Exa [15] already expose semantic web search via MCP, making search a first-class tool for LLM agents. Our work extends this ecosystem to IR research infrastructure, enabling reproducible retrieval experiments—including evaluation against standard relevance judgments—from within any MCP-compatible agent. More recently, Anthropic released Agent Skills [2, 5] as a complementary open standard: structured Markdown playbooks that agents load progressively, incurring minimal token overhead until a skill is actually needed. CLI-based tool interfaces have similarly gained traction as a lighter-weight alternative to MCP's upfront schema declaration [25, 28]. Anthropic itself characterizes the two standards as complementary [29], though the broader ecosystem is still converging on best practices for when to prefer each.

5 Conclusion

We present the MCP servers for Pyserini and RankLLM and the suite of tools they offer. Through these MCP servers, we demonstrate RAG, a new dominant use case of search, and multimodal retrieval. We also show the transferability of our system across different LLMs and clients. Our system enables agentic workflows with advanced retrieval and reranking methods, and we hope it can help advance IR research on RAG.

Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. We are sincerely grateful to Sisi Li for her early contributions and thoughtful discussions, which helped shape the initial direction of this work. We'd also like to thank Neng Li for helpful comments.

References

[1] Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925* (2025).
[2] Agent Skills. 2025. Agent Skills Open Standard. (link).
[3] Anthropic. 2024. Introducing the Model Context Protocol. (link).
[4] Anthropic. 2025. Donating the Model Context Protocol and Establishing the Agentic AI Foundation. (link).
[5] Anthropic. 2025. Equipping Agents for the Real World with Agent Skills. (link).
[6] Anthropic. 2025. Introducing Claude Sonnet 4.5. (link).
[7] Anysphere. 2026. Cursor. (link).
[8] Gemini API. 2025. Function calling with the Gemini API. (link).
[9] Engineering at Anthropic. 2024. Building effective agents. (link).
[10] Michael Bachman and Anna Berenberg. 2025. Announcing Model Context Protocol (MCP) support for Google services. (link).
[11] Andrzej Białecki, Robert Muir, Grant Ingersoll, and Lucid Imagination. 2012. Apache Lucene 4. In *SIGIR 2012 Workshop on Open Source Information Retrieval*. sn, 17.
[12] Claudio Carpineto and Giovanni Romano. 2012. A Survey of Automatic Query Expansion in Information Retrieval. *ACM Computing Surveys (CSUR)* 44, 1 (2012), 1–50.
[13] Harrison Chase. 2022. LangChain. (link).
[14] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss Library. *arXiv preprint arXiv:2401.08281* (2024).
[15] Exa. 2025. Exa MCP Server for AI Web Search. (link).
[16] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6491–6501.
[17] FastMCP. 2026. FastMCP. (link).
[18] Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. 2024. FIRST: Faster Improved Listwise Reranking with Single Token Decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP 2024)*. 8642–8652.
[19] Mohammed Mehedi Hasan, Hao Li, Emad Fallahzadeh, Gopi Krishnan Rajbahadur, Bram Adams, and Ahmed E. Hassan. 2025. Model Context Protocol (MCP) at First Glance: Studying the Security and Maintainability of MCP Servers. *arXiv preprint arXiv:2506.13538* (2025).
[20] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.
[21] Minghan Li, Ming Li, Kun Xiong, and Jimmy Lin. 2021. Multi-Task Dense Retrieval via Model Uncertainty Fusion for Open-Domain Question Answering. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics, Punta Cana, Dominican Republic, 274–287. doi:10.18653/v1/2021.findings-emnlp.26
[22] Jimmy Lin. 2022. A Proposed Conceptual Framework for a Representational Approach to Information Retrieval. *SIGIR Forum* 55, 2, Article 4 (March 2022), 29 pages. doi:10.1145/3527546.3527552
[23] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*. 2356–2362.
[24] Jerry Liu. 2022. LlamaIndex. (link).
[25] LlamaIndex. 2026. Skills vs MCP Tools for Agents: When to Use What. (link).
[26] Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-Shot Listwise Document Reranking with a Large Language Model. *arXiv preprint arXiv:2305.02156* (2023).
[27] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2018), 824–836.
[28] Milvus. 2025. Is MCP Dead? CLI and Skills for AI Agents. (link).
[29] Michael Nuñez. 2025. Anthropic Launches Enterprise 'Agent Skills' and Opens the Standard. (link).
[30] OpenAI. 2025. MCP. (link).
[31] Malte Pietsch, Timo Möller, Bogdan Kostic, Julian Risch, Massimiliano Pippi, Mayank Jobanputra, Sara Zanzottera, Silvano Cerza, Vladimir Blagojevic, Thomas Stadelmann, Tanay Soni, and Sebastian Lee. 2019. Haystack: the end-to-end NLP framework for pragmatic builders. (link).
[32] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! *arXiv preprint arXiv:2312.02724* (2023).
[33] Pydantic. 2025. PydanticAI: AI Agent Framework, the Pydantic way. (link).
[34] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context Retrieval-Augmented Language Models. *Transactions of the Association for Computational Linguistics* 11 (2023), 1316–1331.
[35] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (2009), 333–389.
[36] Sahel Sharifymoghaddam, Ronak Pradeep, Andre Slavescu, Ryan Nguyen, Andrew Xu, Zijian Chen, Yilin Zhang, Yidi Chen, Jasper Xian, and Jimmy Lin. 2025. RankLLM: A Python Package for Reranking with LLMs. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Padua, Italy) *(SIGIR '25)*. Association for Computing Machinery, New York, NY, USA, 3681–3690.
[37] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)*. 14918–14937.
[38] TREC RAG Track Organizers. 2025. TREC RAG Track. (link). Benchmark and task resources.
[39] vLLM Project. 2026. vLLM. (link).
[40] Ellen M. Voorhees and Donna K. Harman. 2005. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press.
[41] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025. BrowseComp: A Simple Yet Challenging Benchmark for Browsing Agents. *arXiv preprint arXiv:2504.12516* (2025).