Beyond Semantic Similarity: Performance and Costs of Agentic Retrieval for Complex Tasks

Reza Esfandiarpoor[*1], Radek Osmulski[*1], Yauhen Babakhin[1], Gabriel de Souza P. Moreira[1], Oliver Holworthy[1], Jie He[‡2], Ronay Ak[1], Jiarui Cai[1], Ryan Chesler[1], Bo Liu[1], Even Oldridge[1]
[1]NVIDIA [2]University of Edinburgh

Abstract

Modern information systems, including many agentic workflows, use dense retrieval to explore large amounts of unstructured data. However, dense retrieval relies on surface-level semantic similarity, which is insufficient for increasingly complex search applications. Here, we investigate agentic retrieval that combines the reasoning capabilities of Large Language Models (LLMs) with the efficient corpus exploration of retrievers in a ReAct agentic loop to solve complex retrieval tasks. In our experiments, we show that agentic retrieval is more effective than standard retrieval, improving nDCG@10 by 8.7 points using the same embedding model. Moreover, while specialized retrieval methods struggle on out-of-domain tasks, agentic retrieval is highly generalizable: the same pipeline achieves competitive results on both the ViDoRe v3 and BRIGHT leaderboards. However, this improvement comes at a cost. On average, agentic retrieval takes 107.4 seconds, compared to 0.67 seconds for standard retrieval, and consumes 764.1K input and 5.8K output tokens per query. In short, our study demonstrates the effectiveness of agentic retrieval in modern data systems and motivates future work on more cost-efficient retrieval agents for large-scale deployment. Code: https://github.com/NVIDIA/NeMo-Retriever/tree/main/retrieval-bench

CCS Concepts

• **Information systems** → **Information retrieval query processing**; **Query reformulation**; **Query intent**; **Retrieval models and ranking**; • **Computing methodologies** → *Natural language processing*.

Keywords

agentic retrieval, ReACT, dense retrieval, large language models, data systems, query semantics

1 Introduction

Many AI workflows, such as retrieval-augmented generation [11, 15] and DeepResearch [2], use information retrieval to process massive amounts of unstructured data. With the growing popularity of these applications, retrieval tasks are shifting from targeted, well-defined search queries to high-level and abstract task descriptions [2, 7]. For example, an agent given a math problem should be able to retrieve potentially useful theorems based on the problem description [16]. Here, we study agentic retrieval to address the needs of complex and reasoning-intensive retrieval tasks[1].

The needs of these emerging retrieval tasks go beyond the capabilities of standard dense retrieval, which relies solely on semantic similarity represented by the cosine similarity between vector representations of the query and documents [10]. In more complex retrieval tasks, however, there may be limited surface-level semantic similarity between a given query, such as a math problem, and the relevant documents, such as useful mathematical theorems. Instead, successfully completing these tasks requires higher-level skills, including reasoning, such as searching over mathematical theorems; knowledge of the dynamics of real-world systems, such as retrieval of tool specifications; and iterative exploration, such as in DeepResearch [2, 7, 16].

In this work, we investigate an agentic approach to information retrieval that combines the capabilities of LLMs and dense retrievers through a ReAct loop [19] to address the requirements of complex retrieval tasks. Our agent inherits the reasoning skills and world knowledge of LLMs while also being able to sift through a massive corpus using lightweight dense retrievers. We evaluate our agent on two dataset collections: ViDoRe v3 [12] for enterprise document retrieval and BRIGHT [16] for reasoning-intensive retrieval tasks. Our main findings are:

- **Effectiveness of Agentic Retrieval**. We show that agentic retrieval is a promising approach for solving complex retrieval tasks. On average, our best agent achieves 8.7 points better nDCG@10 than standard retrieval with the same embedding model.
- **Generalizability**. Agentic retrieval generalizes effectively across domains without any changes. While specialized methods experience a significant drop in performance on out-of-domain tasks, our agentic retrieval pipeline achieves competitive results on both the ViDoRe v3 and BRIGHT leaderboards without modification.
- **Costs**. In addition to performance, we study the costs associated with agentic retrieval, in terms of additional compute. We find that agentic retrieval incurs significantly higher costs than standard dense retrieval, and further work is needed before retrieval agents can be deployed at scale.

Our work provides empirical evidence that standard dense retrieval is insufficient for emerging retrieval applications and establishes agentic retrieval as a promising direction. We also discuss the limitations of widespread adoption of agentic retrieval, primarily in terms of cost and efficiency. We hope our results encourage future work on agentic retrieval that addresses these limitations.

2 Retrieval Agent

The skills needed to solve information-seeking tasks effectively are spread across two types of models. On one hand, LLMs possess

[*]Equal contribution.
[‡]Work done during internship at NVIDIA.
[1]An earlier version of this work has been previously released as the NVIDIA NeMo Retriever Agentic Retrieval pipeline [13].
SAO Workshop at CAIS '26, May 26, 2026, San Jose, California, USA.

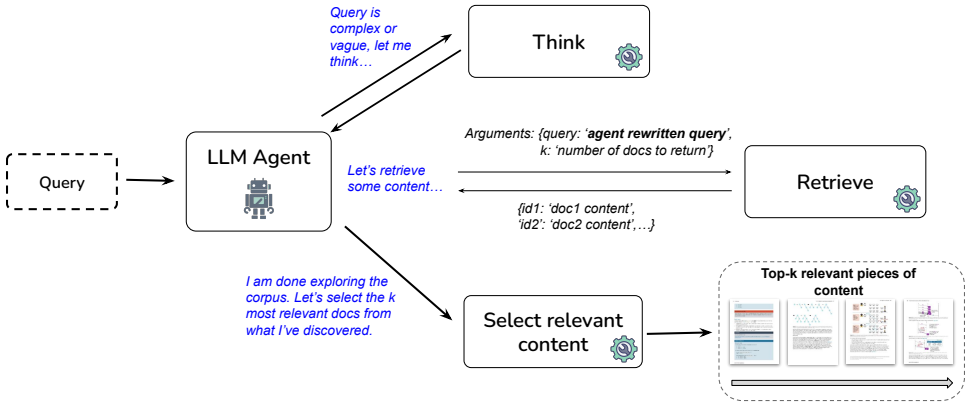

Figure 1: Overview of our retrieval agent. Given a query, the agent iteratively invokes the `think` tool to reason and plan for complex queries, the `retrieve` tool to explore the corpus, and finally `final_results` to return the top-k discovered documents that are most relevant to the query.

significant reasoning capabilities and world knowledge, but they struggle to handle massive corpora of millions of documents. On the other hand, retrievers can easily sift through millions of documents, but they are often small and lack extensive reasoning capabilities. Prior attempts to combine these two model types often rely on static, manually designed workflows. For example, in query rewriting, an LLM is used to rewrite the original query into one or more revised queries, which are then used as inputs to the retriever [8, 17]. This static design limits the autonomy and capabilities of the final solution. For instance, in such designs, the LLM cannot refine queries based on newly discovered information from the retriever.

To address this issue, we create NeMo Retriever Agent in which the query is given to the LLM, and the LLM can then invoke the retriever in a loop to explore the corpus as needed until it discovers the necessary information (Figure 1). This design allows the LLM to adapt its search strategy to each query and corpus, and to dynamically decide when, how often, and with what queries the retriever should be called. We implement this agent using a ReAct loop and standard tool calling through JSON schemas [19]. Compared with custom prompting templates, the ReAct loop makes the agent easier to extend in the future by adding new tools, such as tools for using different retrievers simultaneously, and also enables feedback to be provided to the agent through tool outputs.

Specifically, our agent starts with a system prompt that explains the goal of the task, namely discovering all relevant documents, and describes each tool. The main query is provided in the first user message. We also include the initial documents retrieved for the original query by the dense retriever as part of the user message, and ask the agent to find any missing documents. These initial retrieval results avoid unnecessary search attempts and, more importantly, allow the agent to understand the type of documents in the corpus, such as math theorems or function descriptions, and adjust its search strategy and subsequent queries accordingly. Finally, the agent is provided with the following tools that it can use to solve the task and report the results.

- **Retrieve**. The Retrieve tool takes a query and an integer k as arguments. It is powered by a dense retriever and uses the cosine similarity between query and document embeddings to select the top-k documents most similar to the query and returns them to the agent, along with their unique IDs.
- **Think**. The Think tool takes a string thought argument, which allows the agent to generate additional tokens for reasoning about complex queries or planning its search strategy. The Think tool does not return any output or change the environment.
- **Final_Results**. This tool allows the agent to report the results and terminate the interaction. It expects as input a list of k document IDs that are most similar to the query, sorted by relevance. If the agent selects the wrong number of documents, the tool returns an error and asks the agent to try again. This tool also expects a second argument containing the agent's rationale for selecting these documents.

Reliability. Unlike dense retrievers, retrieval agents can fail for a variety of reasons before selecting the final list of documents. Such failures include exceeding the context window or raising content violation errors. To improve reliability, in these cases, our pipeline falls back to Reciprocal Rank Fusion (RRF). Specifically, when an error occurs, we collect the ranked list of documents from each search attempt before the error. We then use RRF to merge these rankings and calculate a final score for each of these documents.

Infrastructure. Model Context Protocol (MCP) has emerged as a standard method for exposing tools to LLMs [1]. However, in our experiments, we find that for retrieval agents, where a limited number of tools is called many times, MCP leads to compounding overheads: each run requires spinning up a separate server, loading corpus embeddings into GPU memory, and managing the lifecycle of both client and server processes. Moreover, network round-trips add latency to every retrieval call. To address these issues, we replace the MCP server with a *thread-safe singleton retriever* that lives in the same process as the agent. In this approach, the model and corpus embeddings are loaded once, all access is protected by a reentrant lock, and the same `retrieve()` interface is exposed to several concurrent agents. This implementation eliminates an

Table 1: Performance of agentic retrieval and standard dense retrieval with different models.

Method	LLM	Embedding	NDCG@10
		ViDoRe v3	
NeMo Retriever Agent	Opus 4.5	colembed-vl-8b-v2	**69.22**
	gpt-oss	colembed-vl-8b-v2	66.38
	gpt-oss	llama-nemotron-1b	62.42
Standard Retrieval	-	colembed-vl-8b-v2	64.36
	-	llama-nemotron-1b	55.83
		BRIGHT	
NeMo Retriever Agent	Opus 4.5	llama-nv-reasoning-3b	**50.79**
	gpt-oss	llama-nv-reasoning-3b	41.27
	gpt-oss	llama-nemotron-1b	33.85
Standard Retrieval	-	llama-nv-reasoning-3b	38.28
	-	llama-nemotron-1b	19.56

entire class of deployment errors and substantially improves GPU utilization and experiment throughput.

3 Experiments

3.1 Setup

Datasets. We evaluate our pipeline on two benchmarks for complex retrieval. First, we use ViDoRe v3 [12], a benchmark for enterprise document retrieval that spans six languages and 10 domains, including finance, pharmaceutical, and government energy reports. It also contains seven types of queries, such as extractive queries, multi-hop queries, and queries requiring numerical reasoning. We also evaluate on the BRIGHT benchmark for reasoning-intensive text retrieval [16]. Across different domains, BRIGHT measures complex reasoning skills, including logical deduction, code understanding, and mathematical reasoning.

Models. We evaluate our agent using Opus 4.5 as a capable proprietary model. We also experiment with gpt-oss-120b[2] as a representative open-source model. For ViDoRe v3, we use the colembed-vl-8b-v2 retriever[3] [4], a capable model for visual document retrieval. For BRIGHT, we use llama-nv-reasoning-3b[4], which is trained specifically for reasoning tasks. We also report results with the smaller llama-nemotron-1b[5] model on both datasets.

3.2 Results

Table 1 reports the performance of agentic and standard retrieval with different models. Across all cases, agentic retrieval outperforms the same embedding model used in a standard retrieval pipeline. This shows that our retrieval agent better understands the complexity of the given queries and uses the embedding models more effectively than standard dense retrieval pipelines. Our results also emphasize the importance of reasoning capabilities in modern retrieval problems: using the best embedding model for each dataset, our agent with the stronger Opus 4.5 model achieves, on average, a 6.2 point higher nDCG@10 than the same pipeline with gpt-oss-120b. This performance difference is more pronounced on the BRIGHT benchmark, which requires more extensive reasoning skills, where the gap is 9.5 nDCG@10 points. In our investigation on ViDoRe v3, we find that Opus 4.5 makes more search calls per query, 9.2 on average, than gpt-oss-120b, which makes 2.4 calls on average. This suggests that the limited exploration capabilities of gpt-oss-120b may be one cause of the performance difference.

Table 2: Performance of our agent and INF-X-Retriever [20] on ViDoRe v3 and BRIGHT. We also report the performance of standard dense retrieval as well as that of INF-X-Retriever using the same embedding model as our agent. #1 and #2 are leaderboard ranks on March 13, 2026.

Pipeline	ViDoRe v3	BRIGHT
NeMo Retriever Agent (ours)	**69.22** (#1)	50.90 (#2)
INF-X-Retriever	51.01	**63.40** (#1)
INF-X + nemotron-colembed-vl-8b-v2	62.31	–
Dense only (nemotron-colembed-vl-8b-v2)	64.36	–

A particularly interesting finding is that the agent can partially compensate for the limitations of the embedding model. Specifically, the performance gap between weaker and stronger embedding models is much smaller in the agentic retrieval setup than in standard retrieval with the same models: agentic retrieval reduces the performance gap between embedding models with different capabilities by almost half, 57%, on average. This is particularly important because embedding models are often kept very small to handle large numbers of documents, which limits their capabilities.

3.3 Analysis

Generalizability. Agentic retrieval, with its dynamic search strategy for each task, generalizes better to a wide range of tasks than specialized retrieval systems. As of March 13, 2026, our retrieval agent holds the #1 spot on the ViDoRe v3 leaderboard[6] and the #2 spot on the BRIGHT leaderboard[7]. To put the generalizability of agentic retrieval in context, we evaluate the #1 method from the BRIGHT leaderboard, INF-X-Retriever [20], on the ViDoRe v3 benchmark and find that its performance is significantly lower than that of our retrieval agent (Table 2). Moreover, we also evaluate their pipeline using the same retriever that our agent uses. Although the new embedding model improves its performance, the specialized method for the BRIGHT benchmark performs even worse than the standard retrieval baseline in this case. In contrast, the same retrieval agent, without any changes, achieves competitive results on both benchmarks, demonstrating that the agent's dynamic adaptation provides genuine cross-domain generalizability.

Successful Search Patterns. The combination of the LLM's reasoning skills and the iterative agentic loop leads to several recurring search patterns that contribute to the agent's success. *Query Refinement*: the agent dynamically refines and adjusts its subsequent

[2]https://huggingface.co/openai/gpt-oss-120b
[3]huggingface.co/nvidia/nemotron-colembed-vl-8b-v2
[4]huggingface.co/nvidia/llama-nv-embed-reasoning-3b
[5]huggingface.co/nvidia/llama-nemotron-embed-vl-1b-v2
[6]huggingface.co/spaces/vidore/vidore-leaderboard?tab=vidore-v3-pipeline
[7]github.com/NVIDIA/NeMo-Retriever/blob/main/retrieval-bench/submissions/bright_agentic.md

Table 3: Average delay and costs of agentic retrieval on ViDoRe v3 for each query.

	Opus 4.5	gpt-oss-120b
Delay (sec.)	136.3	78.6
Input tokens	759.4k	768.9k
Output tokens	6.2k	5.4k

search queries based on newly discovered information. *Persistent Rephrasing*: the agent persistently rephrases queries until it finds useful information. *Complexity Decomposition*: the agent breaks complex, multi-part queries into multiple simpler queries with clear goals, which are easier for the embedding model to understand.

Costs. The performance gains of agentic retrieval come at a cost. Table 3 reports the average costs of agentic retrieval for each query in ViDoRe v3. While standard retrieval takes 0.67 seconds, agentic retrieval takes 107.45 seconds per query on average. Moreover, completing each query consumes 764.1k input and 5.8k output tokens, which is an additional cost compared to standard retrieval. The additional overhead of LLM inference is the main limitation of agentic retrieval at scale. However, given the increasing complexity of retrieval tasks and the ability of retrieval agents to address this complexity, we hope our results encourage future work on more efficient models and approaches for agentic retrieval.

4 Discussion

Our work offers several lessons for future work on data systems that handle massive amounts of unstructured data.

Evolving applications of retrieval demand new approaches to retrieval. While standard retrieval based on semantic similarity works well for narrow and targeted search queries, it is insufficient for emerging retrieval applications in systems, such as RAG or DeepResearch. These applications demand high-level skills, such as iterative exploration and intermediate reasoning steps, that go beyond what standard retrieval methods offer. Our work shows that agentic retrieval is a promising approach for these more complex retrieval tasks, and our results encourage future work in this direction.

The costs of retrieval go beyond embedding and index lookup. Agentic retrieval introduces new dimensions to retrieval cost, namely the number of reasoning steps and tokens consumed, which are functions of query complexity and dynamic agent behavior. This differs from standard retrieval, where costs primarily correlate with the size of the search index and, therefore, the size of the corpus. Thus, just as there are different methods for controlling costs associated with the search index, such as variants of approximate nearest neighbor search [6], future work on agentic retrieval should investigate similar capabilities for agentic retrieval, allowing users to adapt token and reasoning costs to their specific applications.

More efficient and effective open-source models for agentic retrieval. As shown in Table 1, the performance of open-source models noticeably lags behind that of proprietary counterparts. Moreover, unlike general-purpose agents, the task for retrieval agents is clear, well defined, and requires very specific skills, such as iterative exploration. Taken together, we believe a promising direction for future work is to develop novel models for agentic retrieval that are more effective than current open-source models and more efficient than proprietary models like Opus 4.5.

Infrastructure and system design should adapt to the needs of the target task. As evidenced by our migration from an MCP-based retriever tool to an in-process implementation, standard approaches for agent development are not the best choice for all applications. Another example is tool-calling patterns. Agents often make sequential tool calls one at a time, which aligns with the nature of most real-world tasks. For retrieval, however, parallel tool calls, that is, multiple search attempts in a single turn, are a better option and can reduce the number of steps while potentially improving results by providing more context for the next search attempt. We believe such task-specific optimizations are key to creating effective and efficient retrieval agents which are reusable across many systems.

4.1 Related Work

Retrieval methods have long been based on semantic similarity between queries and documents as captured by their vector embeddings [10, 14, 18, 21]. Many works have also used LLMs to improve this paradigm, for example by rewriting and improving the original query [5, 8, 17]. However, this line of work still relies on a static pipeline and does not fully take advantage of emerging LLM capabilities, such as iterative reasoning and exploration. Agentic retrieval, in contrast, goes beyond semantic similarity alone and incorporates the agentic capabilities of recent LLMs in a dynamic pipeline that adapts to the complexity of each query.

In another line of work, retrievers are frequently incorporated as key components of modern agentic systems for solving specific tasks, such as DeepResearch [2, 3, 9]. In these systems, the retrieval method remains the same as before, while the goal is to have an agent that can effectively use standard retrieval to solve a specific task. Conversely, our goal is to make the retrieval task itself agentic and to create a generalizable retrieval agent that can be reused as the search component of many downstream applications.

5 Conclusion

In this work, we investigate agentic retrieval as a promising solution for addressing the increasing complexity of retrieval tasks. Since standard dense retrieval is based solely on semantic similarity between query and document embeddings, we develop a ReAct agent that uses standard retrievers to explore the corpus while also using LLMs to provide the higher-level capabilities needed for complex queries, such as reasoning and iterative exploration. Our experiments on two complex benchmarks, ViDoRe v3 and BRIGHT, show that agentic retrieval is better equipped to address this level of complexity and significantly improves performance compared to standard dense retrieval. Moreover, we show that agentic retrieval is highly generalizable: the same pipeline performs well on diverse tasks, whereas the performance of specialized solutions drops significantly when they are evaluated on out-of-domain tasks. Finally, we compare the costs of agentic and standard retrieval and discuss the additional compute requirements of retrieval agents. Overall, our work demonstrates the promise of agentic retrieval, discusses its limitations, and encourages future work to further improve the efficiency and effectiveness of agentic retrieval.

References

[1] Anthropic. 2024. Introducing the Model Context Protocol. https://www.anthropic.com/news/model-context-protocol. Accessed: 2025-06-30.

[2] Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, et al. 2025. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600* (2025).

[3] Mingyue Cheng, Jie Ouyang, Shuo Yu, Ruiran Yan, Yucong Luo, Zirui Liu, Daoyu Wang, Qi Liu, and Enhong Chen. 2025. Agent-r1: Training powerful llm agents with end-to-end reinforcement learning. *arXiv preprint arXiv:2511.14460* (2025).

[4] Gabriel de Souza P. Moreira, Ronay Ak, Mengyao Xu, Oliver Holworthy, Benedikt Schifferer, Zhiding Yu, Yauhen Babakhin, Radek Osmulski, Jiarui Cai, Ryan Chesler, Bo Liu, and Even Oldridge. 2026. Nemotron ColEmbed V2: Top-Performing Late Interaction Embedding Models for Visual Document Retrieval. arXiv:2602.03992 [cs.IR] https://arxiv.org/abs/2602.03992

[5] Kaustubh D Dhole and Eugene Agichtein. 2024. Genqrensemble: Zero-shot llm ensemble prompting for generative query reformulation. In *European Conference on Information Retrieval*. Springer, 326–335.

[6] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). arXiv:2401.08281 [cs.LG]

[7] Reza Esfandiarpoor, Vishwas Suryanarayanan, Stephen H Bach, Vishal Chowdhary, and Anthony Aue. 2025. TheMCPCompany: Creating General-purpose Agents with Task-specific Tools. *arXiv preprint arXiv:2510.19286* (2025).

[8] Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653* (2023).

[9] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516* (2025).

[10] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*. 6769–6781.

[11] 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. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401 [cs.CL] https://arxiv.org/abs/2005.11401

[12] António Loison, Quentin Macé, Antoine Edy, Victor Xing, Tom Balough, Gabriel Moreira, Bo Liu, Manuel Faysse, Céline Hudelot, and Gautier Viaud. 2026. ViDoRe V3: A Comprehensive Evaluation of Retrieval Augmented Generation in Complex Real-World Scenarios. arXiv:2601.08620 [cs.AI] https://arxiv.org/abs/2601.08620

[13] Radek Osmulski, Reza Esfandiarpoor, Yauhen Babakhin, Gabriel de Souza P. Moreira, Oliver Holworthy, and Bo Liu. 2026. Beyond Semantic Similarity: Introducing NVIDIA NeMo Retriever's Generalizable Agentic Retrieval Pipeline. NVIDIA Hugging Face Blog. https://huggingface.co/blog/nvidia/nemo-retriever-agentic-retrieval

[14] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*. 5835–5847.

[15] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652* (2023).

[16] Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O. Arik, Danqi Chen, and Tao Yu. 2025. BRIGHT: A Realistic and Challenging Benchmark for Reasoning-Intensive Retrieval. arXiv:2407.12883 [cs.CL] https://arxiv.org/abs/2407.12883

[17] Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 9414–9423.

[18] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808* (2020).

[19] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629 [cs.CL] https://arxiv.org/abs/2210.03629

[20] Yichen Yao, Jiahe Wan, Yuxin Hong, Mengna Zhang, Junhan Yang, Zhouyu Jiang, Qing Xu, Kuan Lu, Yinghui Xu, Wei Chu, Emma Wang, and Yuan Qi. 2025. INF-X-Retriever: A Pragmatic Framework for Reasoning-Intensive Dense Retrieval. https://github.com/yaoyichen/INF-X-Retriever GitHub repository.

[21] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 1503–1512.