MAPS: Modular Agentic Planning and Search

Kayla Lee

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-69
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-69.html

May 15, 2025

Acknowledgement

I am deeply grateful to Professor Daniel Klein and Dr. Eve Fleisig for their guidance throughout my undergraduate and master's studies. To my mom, dad, and brother—thank you for everything, always. To Yash Pansari and Saner Cakir, thank you for the late-night ideation, building together, and your unwavering support throughout this journey.

MAPS: Modular Agentic Planning and Search

By Kayla Chaeeun Lee

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Dan Klein
Research Advisor

5/13/2025

* * * * * * *

Professor Alane Suhr
Second Reader

5/15/2025

5/13/2025

MAPS: Modular Agentic Planning and Search

Kayla Chaeeun Lee[1]

Abstract

We introduce **MAPS**, a modular system for multi-hop retrieval and question answering in open-domain settings. In the open-book, source-not-provided scenario, systems must iteratively retrieve and reason over documents without knowing in advance which sources contain the answer. MAPS addresses this challenge through a structured workflow with three dedicated components: a **Query Planner** that generates and refines search strategies, an **Article Selector** that filters relevant results, and a **Content Extractor** that identifies concise supporting spans. Each module is implemented as a lightweight, task-specific agent, enabling modular separation that improves robustness, scalability, and interpretability. Specifically, MAPS (1) avoids prompt overflow by distributing tasks, (2) enables concurrent document processing to overcome sequential bottlenecks, (3) supports dynamic query refinement through inter-module feedback, and (4) achieves strong performance using smaller models without additional compute cost. We evaluate both the baseline and our approach on a 1,000-question subset of the HotpotQA dataset and achieve a 72% success rate with a GPT-based judge — substantially outperforming the baseline at 42.1%.

```
Question: Who won the World Car of the Year
Award from 2019 to 2023, and what was the
height of each winning car in inches?
```

```
Subquestion: Who won the World Car of the Year Award from 2019-2023?
    Golden Wiki Article: World Car Awards

Subquestion: What is the height of the Jaguar I-Pace in inches?
    Golden Wiki Article: Jaguar I-Pace

Subquestion: What is the height of the Kia Telluride in inches?
    Golden Wiki Article: Kia Telluride

Subquestion: What is the height of the Volkswagen ID.4 in inches?
    Golden Wiki Article: Volkswagen ID.4

Subquestion: What is the height of the Hyundai Ioniq 5 in inches?
    Golden Wiki Article: Hyundai Ioniq 5

Subquestion: What is the height of the Hyundai Ioniq 6 in inches?
    Golden Wiki Article: Hyundai Ioniq 6
```

Figure 1. An example multi-hop question from the FanOutQA dataset, with subquestions and mappings to gold-standard Wikipedia articles.

1. Introduction

Despite the growing popularity of large language models (LLMs) for question answering, these models still struggle to answer questions in an efficient, scalable, and accurate manner. In the standard open-book, source-not-provided setting, recent baselines (e.g., FanOutQA (Zhu et al., 2024)) use a single LLM to issue queries to a search API, retrieve article snippets, and generate final answers. However, this approach faces serious context scaling limitations. Retrieved content must be pre-processed—via chunking, truncation, or BM25 re-ranking—and inserted into the prompt without exceeding input limits. Yet even with these techniques, the system must retain enough detail to support multi-hop reasoning across turns. Relying on a single LLM for both search and answer generation forces the model to repeatedly process raw retrieved content without intermediate filtering. This leads to prompt bloat, as redundant or irrelevant text accumulates. Over time, the model's context fills up, increasing the risk of losing the original query or reasoning path. This makes long-range reasoning fragile and limits recovery from earlier retrieval errors.

MAPS (Modular Architecture for Planning and Search) is a structured system that addresses these challenges:

- **Modular Reasoning Structure:** MAPS splits QA into three modules—planning, retrieval, and extraction—each managed by a task-specific agent. This reduces prompt size and avoids context overflow.
- **Parallel Document Processing:** By decoupling retrieval and extraction, MAPS evaluates articles concurrently, removing ReAct-style sequential bottlenecks.
- **Inter-Module Feedback:** Modules coordinate across steps; e.g., the Article Selector can prompt query refinements, enabling recovery without growing context.
- **Efficient Model Use:** MAPS matches or exceeds larger models by coordinating smaller LLMs effectively, without added compute cost.

We evaluate MAPS on two multi-hop QA benchmarks—**HotpotQA** (Yang et al., 2018) and **FanOutQA** (Zhu et al., 2024)—under the full-wiki, source-not-provided setting. MAPS achieves 72.0% on HotpotQA and 20.0%

on FanOutQA, substantially outperforming a GPT-4o-mini baseline (42.1% and 7.7%). These results demonstrate that modular system design, rather than model scale alone, plays a critical role in complex multi-hop reasoning.

2. Related Work

2.1. Retrieval-Augmented Generation (RAG)

RAG (Lewis et al., 2020) combines dense retrieval with generative models to enhance factuality in open-domain QA. Standard pipelines retrieve top-k documents (e.g., via DPR (Karpukhin et al., 2020)) and pass them to a seq2seq model (e.g., BART, T5). Various extensions address bottlenecks in single-turn retrieval and static queries: FiD (Izacard & Grave, 2020) encodes passages independently, REPLUG (Shi et al., 2023) aligns retrievers with frozen LLMs, and RAG-Fusion (Izacard et al., 2022), HyDE (Cheng et al., 2022), and Multi-hop RAG (Tang & Yang, 2024) expand retrieval coverage and reasoning depth.

2.2. Modular Reasoning Architectures: CoT, ReAct, and MAPS

Recent approaches adopt structured reasoning loops for multi-hop QA. Chain-of-Thought (CoT) prompting (Kojima et al., 2022) encourages models to decompose complex queries into intermediate steps, improving compositional reasoning. ReAct (Yao et al., 2022) extends this idea by interleaving reasoning with external tool use. In ReAct, a single agent maintains all context and alternates between natural language "thoughts" and "actions" such as search API calls.

However, ReAct faces scalability issues in long-horizon tasks: (1) **monolithic prompt expansion** causes context overload, degrading performance as the reasoning chain grows; (2) **sequential bottlenecks** prevent concurrent tool use, slowing down multi-hop retrieval; and (3) **state fragility**, where the agent's entire reasoning history is embedded in one prompt, makes the system brittle—any hallucination or retrieval error can propagate across steps and is hard to recover from.

LLM Compiler (Li et al., 2024) addresses similar challenges via execution graphs of modular subagents. While powerful, it requires formal planning abstractions and is designed for general LLM task orchestration. By contrast, **MAPS** focuses specifically on retrieval-centric QA: it uses lightweight, prompt-based modules for planning, retrieval, and extraction, avoiding the overhead of external execution graphs while preserving the benefits of modularity.

MAPS also draws on recent best practices from Toolformer (Schick et al., 2023) and Auto-CoT (Zhang et al., 2022), which teach models to invoke tools and generate reasoning steps autonomously.

2.3. Extractive QA

Traditional extractive QA benchmarks include **SQuAD** (Rajpurkar et al., 2016), **Natural Questions** (Kwiatkowski et al., 2019), and **TriviaQA** (Joshi et al., 2017), where answers are typically found as contiguous spans within a single passage. Multi-hop variants such as **HotpotQA** (Yang et al., 2018) and **MuSiQue** (Trivedi et al., 2022) introduce additional complexity by requiring systems to integrate information across multiple documents. This shift has led to the development of hybrid pipelines that interleave retrieval and extraction steps.

Agentic systems like **IRCoT** (Trivedi et al., 2023) alternate between reasoning and span selection, refining intermediate facts to guide future steps. **MAPS** builds on this approach by introducing a dedicated *Content Extractor* module that operates independently of planning and retrieval. This separation improves scalability and robustness: extractors can process articles in parallel, while upstream modules remain focused on strategic control. By modularizing extraction, MAPS avoids overwhelming the planner with noisy context and supports efficient multi-document evaluation.

Searching Multiple Articles at Once Single-query pipelines can under-retrieve for multi-hop questions. RAG-Fusion (Izacard et al., 2022) and Multi-hop RAG (Tang & Yang, 2024) address this by issuing multiple queries in parallel, enabling scalable, fault-tolerant reasoning.

Context-Efficient Design Prior work such as RECOMP (Xu et al., 2023) and FiD-KD (Yadav et al., 2022) compress multi-document input to manage context limits. MAPS handles context efficiency through modular filtering and role specialization, as detailed in Section 3.

2.4. Datasets

To evaluate our system, we use two established multi-hop QA benchmarks: **HotpotQA** (Yang et al., 2018) and **FanOutQA** (Zhu et al., 2024). HotpotQA emphasizes precise, two-hop reasoning, while FanOutQA stresses broader retrieval and coordination across many documents.

HotpotQA HotpotQA contains 113,000 Wikipedia-based questions requiring reasoning over two supporting paragraphs. We use the **full-wiki** setting, where models retrieve evidence from the entire English Wikipedia (5M+ articles) without access to gold titles. Supporting fact labels enable fine-grained evaluation of retrieval and answer accuracy.

Although each query involves only two documents, the fact that these documents are drawn from separate Wikipedia articles makes the task well-suited for evaluating retrieval

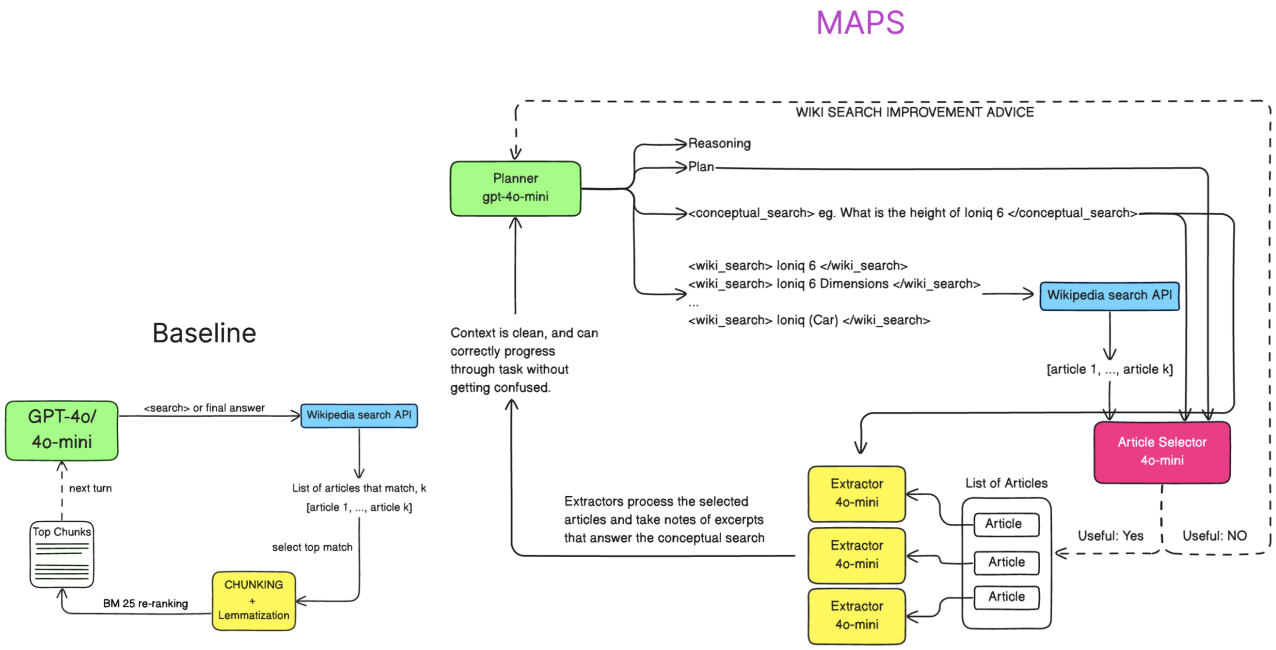

Figure 2. Comparison between baseline and MAPS on a FanOutQA example.

precision and multi-hop reasoning under context constraints.

FanOutQA FanOutQA poses a more demanding variant of multi-hop QA. Each question involves a "fan-out" structure, requiring aggregation of facts scattered across 7+ Wikipedia articles on average. Compared to HotpotQA, it introduces broader evidence chains and greater retrieval noise.

We use the **open-book** setting, where no subquestions or titles are provided. Models must autonomously decompose the query, retrieve documents, and synthesize an answer. Our method replaces the standard BM25 + GPT pipeline with a multi-stage loop better suited for this high-fan-out scenario.

2.5. Baseline

For the baseline, we employ a single GPT-4o-mini instance that interacts with the Wikipedia Search API in a loop until it determines it can produce a final answer. At each step, the model chooses either to issue a new query to the API or to return an answer. When querying, the top-k articles are retrieved, chunked, lemmatized, and re-ranked using BM25. The highest-ranked chunks are then aggregated into a single context, which is fed back into the model for the subsequent turn.

To assess the impact of model size, we replicate this baseline using a GPT-4o instance, allowing us to evaluate whether our multi-agent GPT-4o-mini system can outperform a significantly larger model at a lower computational cost.

2.6. Our Approach (MAPS)

MAPS is designed to address the three key challenges identified in multi-hop QA—prompt overflow, sequential bottlenecks, and state fragility—by modularizing the retrieval and reasoning pipeline. We decompose the task into three core agents: the Query Planner, the Article Selector, and the Content Extractors. Each agent is implemented using GPT-4o-mini and is designed to handle a specific subtask, enabling scalable and robust processing across long multi-hop chains. Below, we describe each module and the specific issue it addresses.

- **Query Planner (mitigates prompt overflow):** This agent receives the initial user query and manages the overall search loop. At each turn, it decides whether to issue a new search query or terminate with a final answer. Crucially, the planner does not process raw documents directly. Instead, it formulates structured search intentions, reducing prompt clutter and keeping context compact and interpretable.

2.6.1. Reasoning

Breaks the original question into atomic factual subgoals, enabling the system to incrementally gather supporting evidence. This is done via prompting.

2.6.2. PLAN

Outlines a high-level multi-hop retrieval strategy, defining the logical steps needed to complete the reasoning chain.

2.6.3. CONCEPTUAL SEARCH

Expresses the next information need in natural language (e.g., "What are the dimensions of the Ioniq 6?")—abstracting away from Wiki query details.

2.6.4. WIKI SEARCH

Translates the conceptual search into one or more concrete keyword strings used to call the Wikipedia Search tool. These are optimized to retrieve relevant titles with three in-context examples. We selected our in-context examples from FanOutQA's dev set.

This structured reasoning protocol ensures that context remains focused on the evolving plan rather than redundant retrieved content. It also allows the planner to recover from mistakes and re-plan when retrieval fails.

- **Article Selector (mitigates state fragility):** After the Planner issues a search, this agent filters the returned article candidates. It receives the search plan, conceptual query, and snippets from the Wikipedia API, and determines which articles are actually useful.

  The Selector returns IDs (or hashes) of articles judged relevant, preventing hallucinated titles and discouraging brittle reliance on surface-level name matching. If none of the articles appear useful, the Selector explicitly prompts the Planner to revise the search query, enabling loop-level recovery and reducing error propagation.

  This coordination improves stability: decisions about retrieval quality are externalized, and state is passed via typed outputs rather than implicit token traces.

- **Extractor Agents (resolve sequential bottlenecks):** For each article accepted by the Selector, a separate Extractor agent is spawned. Each is provided the full article and the associated conceptual search query.

  Each Extractor reads the article and outputs concise snippets directly relevant to the search goal. These are not full summaries, but filtered excerpts designed to feed back into the Planner. This concurrent architecture enables parallel processing of articles, avoiding the step-by-step retrieval-processing cycle of systems like ReAct.

  The concatenated outputs from all extractors are passed to the Planner for the next reasoning step. This allows the system to scale with document fan-out without overwhelming any single prompt.

Together, these components form a modular pipeline with typed, interpretable interfaces. Each module operates with constrained scope and responsibility, which improves debugging, allows for role specialization, and mitigates the core limitations of prior monolithic prompting loops.

System outputs for each stage of MAPS

Query: *What nationality is the actress born in 1940 who appeared in the 2013 film Contest?*

First Turn (identify the actress):

- *Planner Reasoning:* The question asks for an actress born in 1940 who appeared in *Contest (2013)*. Identify her first before retrieving nationality.
- *Plan:* Search for the cast of *Contest*, then verify the birth year.
- *Conceptual Search:* Who are the actresses in the film *Contest*?
- *Wikipedia Search Queries:* `['Contest (film)']`

Search Results: *Contest (disambiguation)*, *Contest (2013 film)*, *No Contest (film)* **Article Selector:** *Contest (2013 film)* **Extractor Summary:** Identifies Mary Beth Peil and Katherine McNamara as cast.

Second Turn (verify actress born in 1940):

- *Planner Reasoning:* McNamara was born in 1996, so likely not the target. Hypothesize Mary Beth Peil as the match; verify her birth year and nationality.
- *Plan:* Search for biographical information on Mary Beth Peil.
- *Conceptual Search:* What is the birthdate and nationality of Mary Beth Peil?
- *Wikipedia Search Queries:* `['Mary Beth Peil']`

Search Results: *Mary Beth Peil*, *Anastasia (musical)* **Article Selector:** *Mary Beth Peil* **Extractor Summary:** Confirms Peil was born in 1940 and is American.

Final Answer: *American*

3. Results

MAPS consistently outperforms the GPT-4o-mini baseline by a large margin: +29.9% on HotpotQA and +12.3% on FanOutQA. On HotpotQA, it even exceeds the performance of GPT-4o despite using a much smaller model.

These gains highlight the effectiveness of modular reasoning and role separation: MAPS avoids prompt overflow, enables parallel evidence extraction, and recovers from poor queries—advantages monolithic prompting frameworks lack.

While FanOutQA provides a challenging benchmark for wide retrieval and planning, it contains some annotation noise due to non-expert labeling. To ensure robustness, we include HotpotQA as a high-quality benchmark for validating multi-hop reasoning with strong supervision.

Table 1. **Success rates of baseline and MAPS on HotpotQA (1000 cases) and FanOutQA (300 cases).**

Model / Method	HotpotQA	FanOutQA
GPT-4o-mini	42.1%	7.7%
GPT-4o	63.7%	22.0%
MAPS (GPT-4o-mini)	**72.0%**	**20.0%**

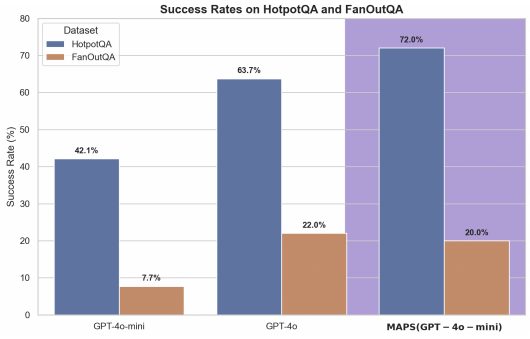

Figure 3. Success rates of baseline and MAPS on HotpotQA (1000 cases) and FanOutQA (300 cases)

3.1. Ablation

3.1.1. Removing the Extractor (Raw Articles to Planner)

To isolate the effect of modularization in MAPS, we evaluated a variant where the Extractor module was removed and raw Wikipedia articles were fed directly into the Planner. This configuration eliminates modular summarization and more closely resembles standard prompting pipelines.

The results were poor: only **9% accuracy** on a **500**-example subset of **HotpotQA**. Over half of the queries resulted in prompt overflow due to excessively long article content. In these cases, the planner's context filled up before it could issue further actions, terminating the loop prematurely.

Even in cases without hard overflows, performance degraded substantially. The planner, when exposed to full articles, failed to reason effectively: it fixated on surface-level spans, lost track of the original conceptual search intent, and generated incoherent or irrelevant plans. This confirms that decomposing retrieval and summarization is essential—not just for memory efficiency, but for maintaining semantic control over the reasoning process.

These findings underscore the importance of modular **Extractors** in MAPS. By filtering and condensing article content, they preserve the Planner's reasoning capacity, enabling it to focus on strategy rather than span-level parsing.

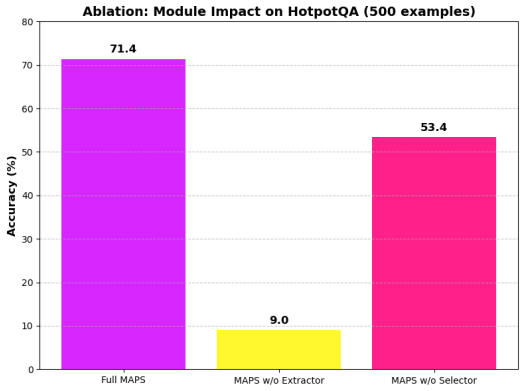

Figure 4. Ablation analysis on HotpotQA showing the impact of removing Extractor and Article Selector modules.

3.1.2. Removing the Article Selector

To assess the value of MAPS's filtering stage, we removed the Article Selector and passed all retrieved articles directly to the Extractor. This setup removes intermediate relevance judgments, forcing the Extractors to process every snippet surfaced by the Wikipedia Search API.

On a 500-example subset of **HotpotQA**, this variant achieved a **53.4%** success rate. While this is lower than the full MAPS score (71.4%), it still outperforms the 4o-mini baseline (42.1%). The gap highlights how intermediate article pruning improves quality, though raw extraction alone still provides some benefit.

Qualitatively, we observed that removing the Article Selector increases the number of search rounds per question. Without filtering, a good article may not be prioritized early, forcing the Planner to issue multiple conceptual searches before converging on useful evidence. This adds latency and increases the risk of early context saturation.

By constraining what gets passed to downstream agents, the

Article Selector limits distractors, accelerates convergence, and strengthens planning precision.

4. Conclusion

We presented **MAPS**, a modular agentic system for multi-hop question answering that cleanly separates planning, retrieval, and extraction. Unlike prior monolithic approaches, MAPS uses lightweight task-specific agents to avoid prompt overflow, support iterative refinement, and process articles in parallel. Our system outperforms stronger baselines—including GPT-4o—despite using smaller models. Ablation studies confirm the importance of modular summarization for maintaining accuracy and stability, especially in long-horizon tasks.

Future work will extend MAPS beyond static prompting toward more adaptive, learning-driven systems. One direction is to develop a trainable controller that dynamically orchestrates module execution based on query complexity, confidence signals, or search trajectory—allowing MAPS to flexibly re-route, skip, or revisit steps. Integrating memory-aware planning would enable long-term context tracking across turns, empowering agents to recall and revise earlier decisions without reprocessing entire histories. Additionally, training the extractor modules with supervised span labels—or via weak supervision from high-confidence answers—could significantly improve precision while enabling fine-tuning on domain-specific tasks. More broadly, MAPS provides a foundation for lifelong QA systems: modular, reconfigurable, and capable of learning new tools or roles over time. As LLMs become more autonomous and teachable, MAPS-style architectures may evolve into persistent reasoning agents that operate across sessions, domains, or even modalities.

References

Cheng, Y. et al. Hyde: Hypothetical document embeddings for qa. arXiv preprint arXiv:2204.04086, 2022.

Izacard, G. and Grave, E. Leveraging passage retrieval with generative models for open domain question answering. arXiv preprint arXiv:2007.01282, 2020.

Izacard, G. et al. Few-shot learning with retrieval-augmented language models. arXiv preprint arXiv:2201.08964, 2022.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and tau Yih, W. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Li, J., Yin, K., Wang, S., Huang, X., Hu, Z., and Neubig, G. Llm compiler: Compile llm agents to computer programs. arXiv preprint arXiv:2310.06825, 2024.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

Schick, T., Dwivedi-Yu, J., Sun, S., Xue, S., Hosseini, S., Zhang, Y., Mueller, J., and Schütze, H. Toolformer: Language models can teach themselves to use tools. arXiv preprint arXiv:2302.04761, 2023.

Shi, W. et al. Replug: Retrieval-augmented language models with frozen lms and optimized retrievers. arXiv preprint arXiv:2301.12652, 2023.

Tang, J. and Yang, D. Multi-hop retrieval-augmented generation for open-domain question answering. arXiv preprint arXiv:2403.07040, 2024.

Trivedi, H., Nori, H., Dua, D., and Gardner, M. Musique: A multi-hop question dataset grounded in wikipedia. arXiv preprint arXiv:2204.07361, 2022.

Trivedi, H., Dua, D., and Gardner, M. Interleaving retrieval with chain-of-thought reasoning for open-domain qa. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

Xu, C., Su, J., Li, J., and Dong, L. Recomp: Retrieval compression for efficient language modeling. arXiv preprint arXiv:2305.03265, 2023.

Yadav, V., Wang, C., Ainslie, J., and Wang, W. Y. Efficient multi-document question answering via knowledge distillation. In *Proceedings of NAACL*, 2022.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Yao, S., Zhao, J., Kassner, N., and Lin, J. React: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629, 2022.

Zhang, X., Sun, S., Deng, Y., and Wu, Y. Automatic chain-of-thought prompting in large language models. arXiv preprint arXiv:2205.11916, 2022.

Zhu, L., Yang, C., Madaan, A., and Bras, R. L. Fanoutqa: A multi-hop, multi-document question answering benchmark for large language models. arXiv preprint arXiv:2402.14116, 2024.