Mnemosyne: Accelerating Multi-Hop Question Answering via Cache Hit Order Fitting

Haizhou Du, Jiujiu Li, Dongyang Li*, Luobin Huang, Lisheng Wang

Shanghai University of Electric Power
haizhou.du@shiep.edu.cn

Abstract

Multi-Hop Question Answering (MHQA) requires step-by-step reasoning across multiple pieces of information to answer complex questions. The cache-aided Retrieval-Augmented Generation (RAG) can accelerate the process of external knowledge retrieval at each reasoning step for MHQA. However, existing methods focus on the internal structure and ignore the misalignment between the queries' arrival order and cache hit order. To tackle this, we propose Mnemosyne, a cache hit order fitting method designed to accelerate the RAG progress for MHQA. Specifically, our cache-aware order fitting strategy adjusts the order of queries arrival via graph reordering to better align with the cache hit order, thereby reducing the likelihood of failed or unproductive retrieval attempts. The multi-granularity caching storage mechanism is designed to loosen the strict hit condition to multiple similar semantic matching modes, facilitating that relevant documents can still be retrieved. Experiments conducted on four multi-hop QA datasets demonstrate that Mnemosyne effectively reduces retrieval latency while enhancing task answer F1 score, achieving a superior trade-off between efficiency and effectiveness.

1 Introduction

Large Language Models (LLMs) have demonstrated advancements across various tasks, including machine translation(Zhang et al. 2023; Liang et al. 2025), text summarization(Wang et al. 2025a; Sahu, Vechtomova, and Laradji 2025), and question answering(Lin et al. 2025; D'Souza, Babaei Giglou, and Münch 2025), primarily owing to their strong reasoning abilities. Nevertheless, these models suffer from outdated knowledge and domain-specific limitations, leading to factual errors in their outputs (Chen et al. 2025; Zhang et al. 2025a). To address this, retrieval-augmented generation (Hamza et al. 2025; Su et al. 2024) enables LLMs to leverage external knowledge bases and information retrieval for more reliable and up-to-date information. This mechanism strengthens LLMs' performance, particularly on complex tasks like multi-hop question answering(Yang et al. 2018), which necessitates multi-step reasoning to derive better answers.

*Corresponding authors

Multi-hop question answering approaches employing large language models generally fall into two main categories: (1) Closed-book Reasoning: These methods typically rely on the internal knowledge of large language models to generate answers. Techniques like QDAMR (Deng et al. 2022b) delegate them to an external tool and achieve the decomposition and answering of multi-hop questions through intermediate products. More conveniently, Chain-of-Thought (Wei et al. 2022) prompts LLMs to generate reasoning steps for complex questions progressively. To enhance the zero-shot multi-hop reasoning capabilities of LLMs, (Jiang et al. 2022) trains them on concatenated single-hop questions or logical forms to approximate real multi-hop natural language questions. PCL(Deng et al. 2022a) augments LLMs trained in single-hop tasks by adding extra sub-networks and learning soft prompts for the novel sub-networks to perform type-specific reasoning. (2) Retrieval-augmented Reasoning: This paradigm integrates external knowledge retrieval to enhance LLM performance on MHQA. Some approaches answer complex multi-hop questions by iterative retrieval and decomposition (Wang et al. 2025b; Shi et al. 2024; Ye et al. 2025). Taking this a step further, some advanced methods model the retrieval or decomposition process as a more complex tree structure and perform multi-hop reasoning via a beam search strategy (Zhang et al. 2024a; Chu et al. 2024). To dynamically manage the overhead for simple versus multi-step queries, some adaptive methods determine the question type to dynamically assign a retrieval strategy (Jeong et al. 2024; Zhang et al. 2025b).

To improve retrieval efficiency, two main categories of cache-aided RAG methods have been proposed: (1) Retrieval-aided Caching. These approaches accelerate inference by reusing or prefetching query-related computations. Examples include speculative retrieval and batch validation (Zhang et al. 2024b), preloading resources into extended contexts (Chan et al. 2025), and leveraging query similarity for document reuse (Bergman et al. 2025). (2) Generation-aided Caching. These methods optimize memory efficiency by managing precomputed key-value states. Notable techniques in this category are reusable-chunk caching (Park et al. 2024), virtual memory-inspired attention partitioning (Kwon et al. 2023), and multi-level dynamic caching of intermediate knowledge states (Jin et al. 2024). However,

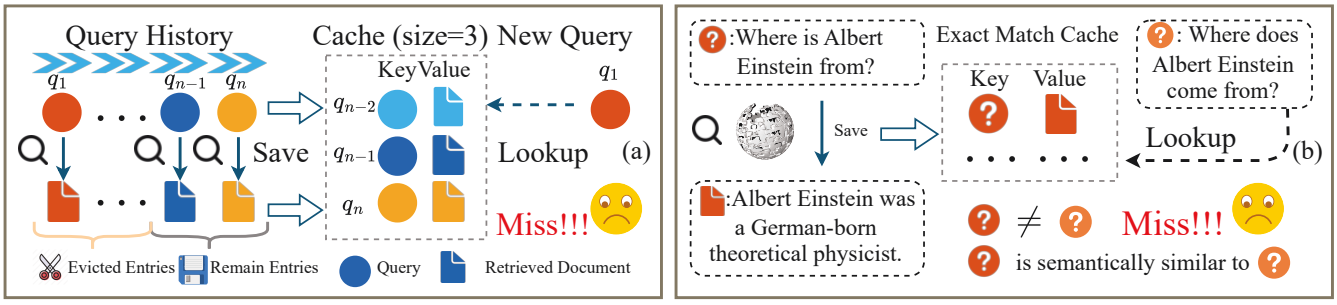

Figure 1: Two reasons for cache miss. Figure 1(a) illustrates that under the condition of a limited cache capacity, semantically similar questions are evicted due to long access intervals. This results in the inability to find these potentially useful entries when needed, thereby leading to cache misses. Figure 1(b) explains that in question-answering tasks, an exact match cache fails to leverage semantically similar but lexically different sentences, thereby causing the inability to utilize knowledge in cache.

the effectiveness of these cache-aided methods is inherently constrained by the stiff hit condition and requests arrival order in MHQA.

The performance of existing cache-aided methods can be attributed to two primary reasons: (1) The order of requests arrival is not a cache-aware order. As illustrated in Figure 1(a), a cache with limited capacity (*e.g.*, storing only two recent queries and their documents) can easily suffer misses when a new query, similar to an earlier one, arrives, as the original entry might have been evicted. (2) A cache with a stiff hit condition is not suitable for a question-answering task with ambiguity. As shown in Figure 1(b), even if an exact match cache stores a semantically similar query and its retrieved document, it may fail to recognize a new, slightly varied query due to strict hit condition. To overcome these critical limitations, our research specifically focuses on the following question: *How can we reduce the retrieval overhead of RAG in MHQA with a limited-capacity cache?*

To address this issue, we propose a cache hit order fitting method designed to accelerate multi-step retrieval by substantially boosting the cache hit rate, which we name Mnemosyne. This method comprises two core components: (1) cache-aware order fitting strategy. This strategy reorders queries by increasing the overlap of named entities between neighbor queries for an optimized sequence with high locality.[1] (2) multi-granularity caching storage mechanism. Besides the traditional exact-match cache, we develop a more granular cache storing entity-document pairs, which provides a more flexible hit condition for a higher hit rate. We evaluate our approach on four MHQA benchmarks. Empirical results demonstrate that our approach achieves a speedup ratio of up to 1.81× while simultaneously improving the answer F1 Score by up to 13 points, thus offering a superior efficiency-effectiveness trade-off.

Our main contributions are summarized as follows:

- We propose Mnemosyne, a cache hit order fitting method for multi-round retrieval that improves the cache hit rate to provide acceleration.
- We introduce a cache-aware order fitting strategy that enhances the utilization rate of cache entries by making similar questions closer in sequence. We further propose the multi-granularity caching storage mechanism to leverage semantic similarity between queries by adding a more granular cache storing entity-documents pairs.
- We evaluate Mnemosyne on four benchmarks. Empirical results consistently demonstrate that our approach provides performance acceleration without compromising answer F1 scores, even with a limited-capacity cache.

[1]Cloud service providers possess finite hardware resources, limiting the number of requests they can process concurrently. Faced with a massive volume of requests, they are compelled to serialize them. By implementing appropriate request reordering, the total request processing time can be reduced.

2 Related Work

2.1 Multi-Hop Question Answering

Multi-hop question answering is more complex than simple QA because it involves not just retrieving information, but also effectively integrating related facts. Existing approaches for multi-hop question answering with large language models often tackle this by breaking down complex queries, like SG-FSM(Wang et al. 2025b), which uses a self-guided finite-state machine to decompose questions and self-correct intermediate answers, dynamically adjusting its reasoning path. Similarly, IRCoT(Trivedi et al. 2023) interleaves retrieval with chain-of-thought (CoT) steps, using CoT to guide retrieval and refine reasoning, while Q-DREAM (Ye et al. 2025) optimizes the semantic space of sub-questions through a three-module pipeline encompassing question decomposition, dependency modeling, and dynamic retrieval alignment. Other methods treat multi-hop retrieval as a joint optimization problem, employing techniques like beam search to maintain multiple partial hypotheses per step to mitigate omission risks; for example, Beam Retrieval(Zhang et al. 2024a) jointly optimizes an encoder and dual classification heads end-to-end, expanding the search space through beam-based partial hypothesis tracking, and BeamAggR(Chu et al. 2024) parses questions into atomic-composite trees, performing bottom-up reasoning with beam-pruned aggregation. Furthermore, some advanced frameworks dynamically select strategies based on query complexity, such as BELLE(Zhang et al. 2025b),

which employs a bilevel multi-agent debate system where operators are combined via deliberative planning and fast/slow debaters monitor reasoning consistency, and Adaptive-RAG(Jeong et al. 2024), which uses a lightweight classifier to predict query complexity and route tasks to appropriate retrieval strategies.

2.2 Cache-Aided RAG

Recent research on cache-aided retrieval-augmented generation primarily focuses on two directions: generation-aided caching and retrieval-aided caching. For KV cache optimization of the generation phase, (Park et al. 2024) proposes Cache-Craft, a system that reuses precomputed key-value (KV) caches for repeated text chunks in RAG inputs while maintaining output quality via partial recomputation and hardware-efficient cache management. (Kwon et al. 2023) introduces PagedAttention, an attention algorithm inspired by virtual memory techniques, which eliminates KV cache fragmentation via paging and enables flexible cache sharing across requests, as implemented in their vLLM serving system. For speculative retrieval acceleration of retrieval phase, (Zhang et al. 2024b) presents RaLMSpec, a framework that batches speculative retrievals with asynchronous verification and prefetching to maximize speedup over iterative RAG-LM pipelines. In approximate caching of the retrieval phase, (Bergman et al. 2025) develops Proximity, which reuses retrieved documents for semantically similar queries to reduce vector database lookups, while (Chan et al. 2025) explores cache-augmented generation, preloading knowledge into extended LLM contexts to bypass retrieval entirely. Additionally, (Jin et al. 2024) proposes RAGCache, a multilevel dynamic cache that organizes retrieved knowledge into GPU/host-memory hierarchies and optimizes replacement policies based on LLM inference patterns.

3 Methodology

The overview of Mnemosyne method is first depicted in Figure 2. Then, we begin with rearranging the questions in cache-aware order fitting strategy, followed by RAG with multi-granularity caching storage mechanism.

3.1 Cache-Aware Order Fitting Strategy

The order of query arrival sequence plays a significant role in cache hit rates. When similar questions appear more closely in the sequence, the cache hit ratio tends to be higher. We utilize w-hop locality to measure the similarity between neighboring questions, defined as follows:

$$\text{locality}(\mathcal{Q}; w) = \sum_{\substack{q_i, q_j \in \mathcal{Q} \\ 0 < j - i \leq w}} s(q_i, q_j), \tag{1}$$

where $\mathcal{Q} = \{q_1, q_2, \cdots, q_n\}$ represents a query sequence and $s(\cdot, \cdot)$ denotes a similarity function, and $w \in \mathbb{Z}^+$ represents the window size.

Optimizing the measure can be defined as constructing a bijective function $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$, which rearranges the elements of the question sequence to obtain $\hat{\mathcal{Q}} = \{\hat{q}_i | \hat{q}_i = q_k, \sigma(k) = i, \forall q_k \in \mathcal{Q}\}$. The goal is to maximize the w-hop sparse locality of $\hat{\mathcal{Q}}$. However, this task is an NP-hard problem, as proven in (Coleman et al. 2022). Specifically, when $w = 2$, the issue is equivalent to the NP-hard problem of maximum weighted Hamiltonian path.

In terms of the semantic relevance between queries and supporting documents, we propose the following new metric of Sparse Locality:

$$\text{S}(\hat{\mathcal{Q}}; w) = \sum_{\substack{\hat{q}_i, \hat{q}_j \in \hat{\mathcal{Q}} \\ 0 < j - i \leq w}} \frac{C(\mathcal{E}_{\hat{q}_i} \bigcap \mathcal{E}_{\hat{q}_j})}{C(\mathcal{E}_{\hat{q}_i}) \cdot C(\mathcal{E}_{\hat{q}_j})}, \tag{2}$$

where $C(\cdot)$ counts the elements in a set, and $\mathcal{E}_{\hat{q}_i}$ denotes the entities identified in $\hat{q}_i$. For entity recognition, we employ Spacy (Honnibal and Montani 2017) and EDC (Zhang and Soh 2024), a LLM-driven information extraction tool, to identify named entities. To optimize this measure, we introduce a cache-aware reordering algorithm applied to a query-entity bipartite graph to maximize w-hop sparse locality. First, we construct a bipartite graph $B = (\mathcal{Q}, \mathcal{E}_\mathcal{Q}, E)$, where $\mathcal{E}_\mathcal{Q} = \bigcup_{i=1}^{n} \mathcal{E}_i = \{e_1, e_2, \cdots, e_{|\mathcal{E}_\mathcal{Q}|}\}$, $E = \{(q_i, e_j) | e_j \in \mathcal{E}_{q_i}\}$.

The cache-aware reordering algorithm receives a bipartite graph, constructs bidirectional adjacency lists, and initializes a max-heap priority queue based on node degrees. During execution, it processes nodes in priority order while maintaining a sliding window of recently processed nodes. This window is used to update priorities for nodes sharing neighbors with the oldest window entry, while simultaneously boosting priorities for nodes sharing neighbors with the current node. The algorithm terminates when all nodes are ordered. The complete pseudocode for this cache-aware reordering algorithm. The time complexity is primarily determined by heap operations and neighbor updates, approximately $O(|\mathcal{Q}| \log |\mathcal{Q}|)$ in sparse graphs, while the space complexity remains $O(|\mathcal{Q}| + |E|)$ due to the storage requirements for adjacency lists and priority tracking structures.

Without loss of generality, we only demonstrate sparse locality in simply connected sub-graphs here, as questions from different connected sub-graphs share no common entities. Thus, the sparse locality computation can be distributed across different simply connected sub-graphs. Singleton sub-graphs indicate no caching benefit; therefore, we disable the cache component in the RAG subsequently to avoid cache thrashing. The output of this section is a locality-aware question sequence $\hat{\mathcal{Q}}$.

3.2 Multi-Granularity Caching Storage Mechanism

We enhance multi-round RAG with Multi-Granularity Caching Storage Mechanism (MGCSM) to balance effectiveness and efficiency in the retrieval phase, consisting of an exact match query-document cache (L1-cache) and an approximate entity-document bipartite cache (L2-cache). The former cache operates with a strict hit condition and has a lower hit ratio, while the latter adopts a more flexible approach with higher ambiguity but achieves higher hit rates. The bipartite cache structure comprises two lists, $l_\mathcal{E}$ and $l_\mathcal{D}$, storing entities and documents, respectively. A dictionary

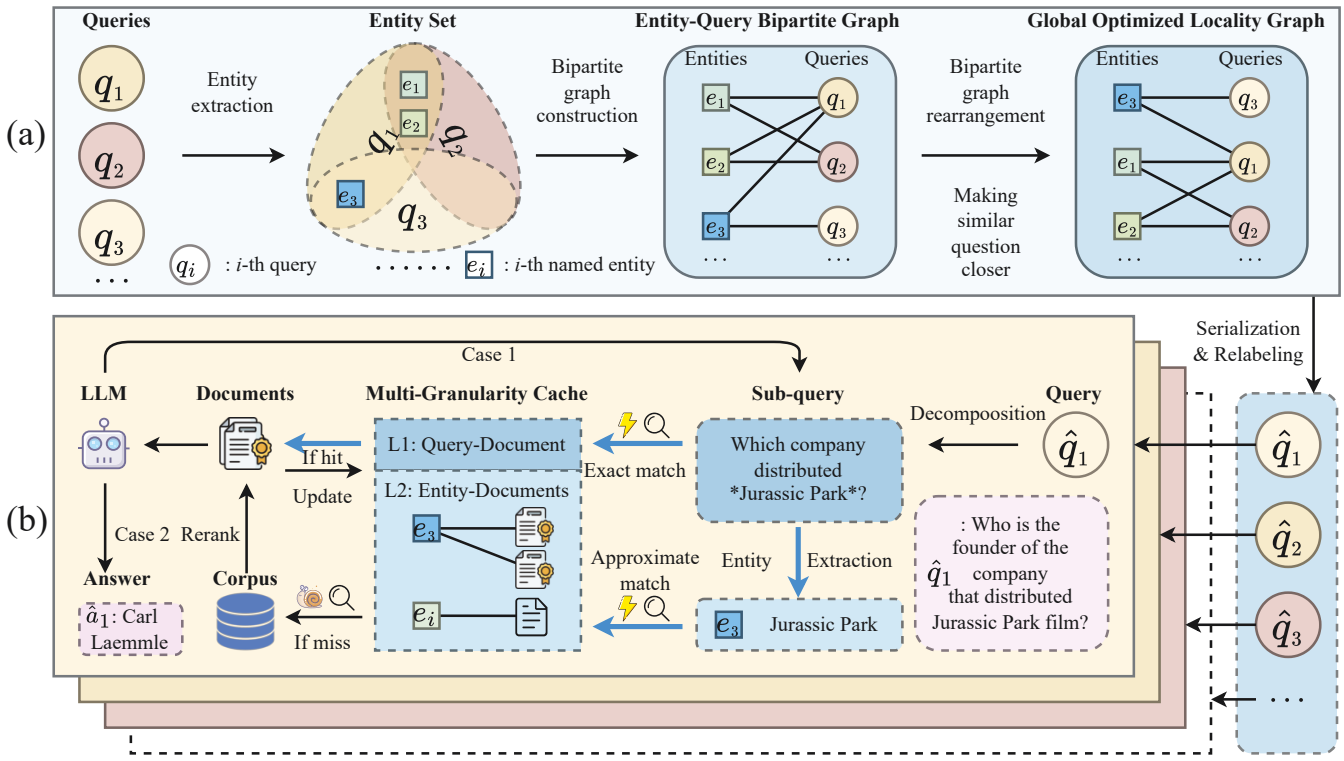

Figure 2: Mnemosyne architecture. Figure 2(a) shows cache-aware order fitting strategy, which rearranges questions to place semantically similar questions nearer, by enhancing the 2-hop sparse locality for a higher cache hit rate. Figure 2(b) shows the workflow of the existing RAG with multi-granularity caching storage mechanism, utilizing a query-document cache (L1) and an entity-document cache (L2).

$\mathcal{M}$ maintains mappings between entities and documents. In this mapping, each entity can correspond to multiple documents, and each document may be associated with multiple entities. Both L1-cache and L2-cache have a fixed capacity C and are initialized as empty caches.

Following previous work (Shi et al. 2024), the RAG for Mnemosyne consists of two phases: retrieval and deduction. In the deduction stage, given an initial question $\hat{q}_i$ (denoted as q^0), we decompose q^0 into a simpler sub-question q^1 and a core named entity e^1 by utilizing a custom prompt ϕ to guide the LLM to generate "*" surrounding e^1. During the retrieval phase, we simultaneously look up q^1 in the L1-cache and retrieve e^1 from the L2-cache. For the L2-cache, we compute the normalized Levenshtein distance (Yujian and Bo 2007) between e^1 and all entities stored in $l_{\mathcal{E}}$. If the distance is less than the cache tolerance λ, the retrieval on L2-cache is classified as a cache hit. When both caches hit simultaneously, we prioritize collected documents $\mathcal{D}^1_{L1-cache}$ from the L1-cache, as the L1-cache provides more precise information. If only one cache hits, we collect the corresponding documents. After retrieval, we use the gathered documents (denoted as $\mathcal{D}^1$) to reason the next sub-question q^2. This process iterates until the LLM outputs q^t containing the tag "FINISH[<answer>]" or the iteration count reaches N_i, where "<answer>" stands for the final answer. The general reasoning process follows this workflow:

$$q^t \leftarrow LLM(\phi \oplus q^0 \oplus \mathcal{D}^1 \cdots \oplus q^{t-1} \oplus \mathcal{D}^{t-1}), \quad (3)$$

where $\oplus$ denotes the text concatenation operation.

If both of the two caches miss in the t-th iteration, we retrieve the top-N documents $\mathcal{D}^t_{corpus}$ from the corpus, rerank them, and collect the top-K documents $\mathcal{D}^t_{rerank}$ as the final document set $\mathcal{D}^t$. The generic retrieval process can be summarized as follows:

$$\mathcal{D}^t = \begin{cases} \mathcal{D}^t_{L1-cache} & \mathcal{D}^t_{L1-cache} \neq \emptyset \\ \mathcal{D}^t_{L2-cache} & \mathcal{D}^t_{L1-cache} = \emptyset \wedge \mathcal{D}^t_{L2-cache} \neq \emptyset \\ \mathcal{D}^t_{rerank} & \mathcal{D}^t_{L1-cache} = \emptyset \wedge \mathcal{D}^t_{L2-cache} = \emptyset \end{cases}. \quad (4)$$

In cache update process, $l_{\mathcal{E}}$ appends the current entity e^t while $l_{\mathcal{D}}$ extends the list with the elements from the corresponding document set $\mathcal{D}^t$. Upon reaching the L2-cache's capacity C, our implementation triggers the Least-Recently-Used replacement strategy to remove the old entries, mirroring this behavior in the L1-cache.

We observe that when the cache returns unhelpful documents, the LLM tends to generate repetitive sub-questions and named entities similar to the previous hop, leading to an ineffective reason-hit loop. To mitigate this cycle, we propose an idempotency rule: if the L2-cache receives the same

named entity twice consecutively during processing a single question, it returns an empty set to break the loop.

The efficiency of the multi-granularity caching storage mechanism originates from three key factors: (1) A limited-capacity cache provides a significantly smaller search space than the entire corpus; (2) Multi-granularity caching has a looser hit condition; (3) The multi-granularity cache stores high-quality documents after reranking, effectively reusing the noise filtering process.

4 Experiments

4.1 Experimental Setups

We describe our experimental setups in this section, including downstream datasets, retrievers, and the implementation details of the baselines.

Datasets. Following established practices in the field, we evaluate our approach on four widely-used MHQA benchmarks: HotpotQA (HQA)(Yang et al. 2018), MuSiQue(Trivedi et al. 2022), 2WikiMultiHopQA (WQA)(Ho et al. 2020), and Multihop-RAG (MRQA)(Tang and Yang 2024). Among these, MRQA represents a specialized benchmarking dataset designed to assess RAG on multi-hop queries, containing 2,556 carefully curated questions categorized into four distinct types (inference, comparison, temporal, and null), each supported by 2-4 ground-truth evidence snippets. To comprehensively evaluate our approach's performance, we strategically sampled 250 items from questions containing shared entities and another 250 items from the remaining question pool.

Retrievers. Following previous work (Zhang et al. 2024b), we evaluate our approach against both dense retrievers (vector-based) and sparse retrievers (bag-of-words-based) to demonstrate the robustness of our approach. For dense retrieval, we compare exact and approximate methods, where the latter offers faster performance at the cost of reduced precision. Specifically, we employ bge-small-en-v1.5 (Xiao et al. 2024) as our exact dense retriever and jina-colbert-v2 (Jha et al. 2024) as the approximate dense retriever. For sparse retrieval, we utilize the BM25 algorithm (Robertson and Zaragoza 2009), implemented through Pyserini[2], while adopting the Stanford ColBERT[3] implementation for dense retrieval.

Baselines. For naive retrieval-augmented generation, we follow the implementation from (Trivedi et al. 2023), which interleaves retrieval with individual reasoning steps in a CoT process. We denote the baseline naive RAG with exact-match query caching as *CAG-R*, implemented according to (Chan et al. 2025). Additionally, we employ the approximate caching as *Proximity* baseline from (Zhang et al. 2024b; Bergman et al. 2025).

Implementation Details. For overall performance evaluation in Table 1, we set the cache capacity to 20 across all methods and configured the cache tolerance at 0.2 for both Mnemosyne and the approximate approach. Furthermore, we incorporate bge-reranker-v2-m3 (Chen et al. 2024) as our reranker and deploy it as an online service using FastAPI. In all our experiments, we retrieve top-10 passages and rerank top-2 passages for each question. Additional experiments investigating different cache capacities and tolerance thresholds are presented for further analysis. In Mnemosyne, we optimized locality by setting the sliding window size to 6 and N_i to 6. We mainly employ Qwen2-7B-Instruct as the LLM, with decoding temperature fixed at 0 for deterministic generation. The implementation uses Python and was tested on a virtual container equipped with one A800 GPU and 14 Xeon(R) Gold 6348 CPU cores.

4.2 Comparison across Retrievers and Datasets

Our empirical evaluation across multiple retrievers (BM25, bge-en-small-1.5, Jina-ColBERT-v2) and datasets demonstrates Mnemosyne's consistent superiority over CAG-R and proximity caching baselines. As shown in Table 1, our approach achieves the best speedup (up to 1.81×), hit rate (up to +47.4%), and F1 improvement (up to +13.6%) on HotpotQA, 2WikiMultiHopQA, and Multihop-RAG. Notably, with exact dense retrieval, Mnemosyne attains a 1.27× speedup and +3.1 F1 gain on the challenging MuSiQue dataset, while maintaining hit rate advantages of 13.2% (sparse) and 19.7% (approximate dense), confirming its robustness across diverse retrieval paradigms. The results highlight Mnemosyne's ability to balance efficiency and accuracy through its multi-granular caching architecture and ranking module.

Our analysis reveals several critical insights into the behavior of different caching strategies across retrieval methods and datasets. The CAG-R approach suffers from increased retrieval latency (0.87-0.98 × speedup) due to cache retrieval and update overhead, while its low hit rates (0.6-5.1%) stem from the inherent complexity of multi-hop questions, where decomposed sub-questions rarely repeat verbatim. In contrast, proximity caching improves speed (1.12-1.45×) and hit rates (8.4-27.6%) but consistently degrades answer quality (Δ F1: -2.3 to -16.1), as its reliance on coarse-grained question similarity introduces noise, particularly detrimental in multi-hop reasoning where precise intermediate retrieval is crucial. Our approach uniquely optimizes all three metrics: it achieves superior speedups (up to 1.81×), higher hit rates (up to 47.4%), and F1 gains (up to +13.6%) by leveraging multi-granular caching to balance precision-efficiency trade-offs and a ranking module to optimize retrieval order. Notably, dense retrievers (exact/approximate) exhibit greater benefits under Mnemosyne, higher hit rates (18.7-41.2%) and more pronounced speedups (1.17-1.81×), because their slower baseline retrieval times amplify the impact of caching, while their superior document quality (vs. sparse retrieval) enhances downstream answer fidelity by reducing LLM reasoning errors. Dataset-wise, Mnemosyne excels on HotpotQA, 2WikiMultiHopQA, and Multihop-RAG, where its design aligns with typical multi-hop patterns, but faces limitations on MuSiQue's extreme compositional questions: here, proximity caching's aggressive acceleration (1.14-1.19×) comes at severe F1 costs

[2]https://github.com/castorini/pyserini

[3]https://github.com/stanford-futuredata/ColBERT

Method	HotpotQA			MuSiQue			2WikiMultiHopQA			Multihop-RAG		
	Speedup	Hit	Δ F1	Speedup	Hit	Δ F1	Speedup	Hit	Δ F1	Speedup	Hit	Δ F1
Sparse Retriever												
CAG-R	0.89 ×	2.3	0.3	0.92 ×	0.6	**0.1**	0.87 ×	2.1	1.1	0.92 ×	4.2	0.6
Proximity	1.21 ×	27.4	-8.1	**1.14** ×	8.4	-5.4	1.13 ×	15.4	-12	1.21 ×	21.3	-7.2
Mnemosyne	**1.57** ×	**35.3**	**3.2**	1.08 ×	**13.2**	-0.6	**1.24** ×	**24.7**	**13.6**	**1.56** ×	**47.4**	**9.3**
Exact Dense Retriever												
CAG-R	0.98 ×	3.1	0.4	0.97 ×	0.7	0.0	0.96 ×	1.7	0.6	0.98 ×	5.1	0.2
Proximity	1.34 ×	22.9	-13.6	1.24 ×	11.2	-2.3	1.34 ×	13.2	-5.2	1.41 ×	27.6	-5.4
Mnemosyne	**1.74** ×	**39.4**	**1.3**	**1.27** ×	**18.7**	**3.1**	**1.60** ×	**28.1**	**7.3**	**1.81** ×	**41.2**	**2.1**
Approximate Dense Retriever												
CAG-R	0.95 ×	3.2	0.2	0.94 ×	0.7	0.0	0.95 ×	1.9	0.2	0.96 ×	4.7	1.3
Proximity	1.45 ×	24.5	-16.1	**1.19** ×	13.1	-4.2	1.12 ×	16.9	-9.1	1.36 ×	19.7	-8.2
Mnemosyne	**1.68** ×	**37.6**	**8.3**	1.17 ×	**19.7**	**1.2**	**1.31** ×	**32.1**	**3.3**	**1.71** ×	**34.3**	**5.1**

Table 1: Evaluation results on four MHQA datasets with different types of retrievers, including sparse retriever (BM25), exact dense retriever(bge-en-small-1.5), approximate dense retriever(Jina-ColBERT-v2). The speedup denotes the acceleration ratio in the retrieval phase, hit refers to the cache hit rate, and F1 represents the answer F1 score.

Ablation Study	HQA			WQA
	Latency	Hit	F1	Latency
Mnemosyne	**0.18**	**35.3**	**54.6**	**0.13**
w/o Reorder	0.28	22.9	53.5	0.16
w/o MGCSM-L1	0.21	35.2	54.2	0.15
w/o MGCSM-L2	0.28	4.1	52.3	0.18
w/o Idempotency	0.20	39.9	54.0	0.15

Table 2: Evaluation results of ablation study on HopotQA and 2WikiMultihopQA with sparse retriever. MGCSM denotes the multi-granularity caching storage mechanism.

(-4.2% to -5.4%), while CAG-R's rigidity preserves F1 (Δ+0.1) but fails to accelerate (0.92-0.97×), highlighting Mnemosyne's current sensitivity to ultra-fine-grained reasoning chains. These findings underscore the need for dynamic caching strategies that adapt not only to retrieval types but also to dataset-specific reasoning depths.

4.3 Ablation Study

We conducted ablation studies on the HotpotQA and 2WikimultihopQA datasets under Sparse Retriever conditions.

w/o Reorder Strategy. When removing the cache-aware order fitting strategy from Mnemosyne to examine the impact of question arrival order, Table 2 shows that this ablation leads to a significant drop in hit rate (from 35.3% to 22.9%) and increased latency (from 0.18s to 0.28s), though the acceleration effect is not entirely eliminated. This indicates that proper arrival ordering positively affects cache hits, while the multi-granularity caching mechanism still contributes partially.

w/o MGCSM. To evaluate the contribution of single-granularity caches, we separately ablated L1-cache and L2-cache. Removing L1-cache results in marginal hit rate degradation (35.3% to 35.2%) but worsens answer F1 (54.6% to 54.2%) and latency (0.18s to 0.21s), suggesting L1-cache provides higher precision despite lower coverage. In contrast, removing L2-cache drastically reduces hit rate (35.3% to 4.1%) and F1 (54.6% to 52.3%) while increasing latency (0.18 to 0.28), confirming L2-cache's critical role in boosting coverage despite introducing some noise. The results demonstrate that L1-cache alone is insufficient for higher recall, necessitating L2-cache's complementary role.

w/o Idempotency Rule. By ablating the idempotency rule, we observe a paradoxical increase in hit rate (35.3% to 39.9%) alongside declines in F1 (54.6% to 54.0%) and latency (0.18s to 0.20s). This reveals that without an idempotency rule, cached irrelevant documents may mislead the LLM into generating repetitive sub-questions, artificially inflating hit rates while degrading answer quality and efficiency due to reasoning loops.

4.4 Impact of Cache Tolerance

To investigate the impact of cache tolerance on our approach, we conducted experiments with cache tolerance values ranging from 0.1 to 0.4. As shown in Table 3, the results demonstrate that as cache tolerance reduces, the hit rate generally decreases while the F1 score initially improves before declining at higher tolerance levels, and latency exhibits a similar trend of first decreasing then slightly increasing. Specifically, we observe that an overly high cache tolerance (*e.g.*, 0.4) leads to an increased hit rate (41.7% for HQA) but fails to reduce latency (0.22s for HQA) while simultaneously degrading answer quality (42.4% F1 for HQA), indicating that such settings introduce noisy text segments that may boost cache hits but hinder the LLM's reasoning capability over documents. Conversely, an excessively low cache tolerance (*e.g.*, 0.1) causes reduced hit rates (28.1% for HQA), lower F1 scores (48.9% for HQA), and marginally

Cache	HQA			WQA	
Tolerance	Latency	Hit	F1	Latency	F1
0.4	0.22	**41.7**	42.4	0.16	51.7
0.3	0.22	37.7	44.6	0.15	55.0
0.2	**0.18**	35.3	**54.6**	**0.13**	**66.2**
0.1	0.20	28.1	48.9	0.14	57.6

Table 3: Comparisons of different cache tolerance on HopotQA and 2WikiMultihopQA with sparse retriever.

Cache	HQA			WQA	
Capacity	Latency	Hit	F1	Latency	F1
10	0.26	24.7	52.2	0.16	57.6
20	0.18	35.3	54.6	0.13	66.2
40	0.17	38.5	55.7	0.12	68.6
80	**0.17**	**39.6**	**55.9**	**0.12**	**68.9**

Table 4: Comparisons of different cache capacity on HopotQA and 2WikiMultihopQA with sparse retriever.

higher latency (0.2s for HQA), suggesting that overly strict matching conditions underutilize potentially relevant cached entries. The optimal balance is achieved at a tolerance of 0.2, which delivers the best performance across metrics: lowest latency (0.18s for HQA, 0.13s for WQA), competitive hit rates (35.3% for HQA), and peak F1 scores (54.6% for HQA, 66.2% for WQA). This non-monotonic behavior underscores the importance of carefully calibrating cache tolerance to balance retrieval efficiency with answer quality.

4.5 Effect of Cache Capacity

To evaluate the impact of cache capacity on our approach, we conducted experiments with varying capacities (10, 20, 40, and 80), as shown in Table 4. The results demonstrate that increasing the cache capacity generally improves performance across all metrics, though with diminishing returns. Specifically, we observe that lower capacities (*e.g.*, 10) lead to higher latency (0.26s for HQA, 0.16s for WQA), lower hit rates (24.7% for HQA), and reduced F1 scores (52.2% for HQA, 57.6% for WQA), which can be attributed to insufficient storage for useful entries, resulting in frequent evictions and cache thrashing. As the capacity increases to 20 and 40, latency decreases (0.18s to 0.17s for HQA; 0.13s to 0.12s for WQA), hit rates improve (35.3% to 38.5% for HQA), and F1 scores rise (54.6% to 55.7% for HQA; 66.2% to 68.6% for WQA). However, the gains diminish beyond 40, with marginal improvements at 80 (*e.g.*, HQA F1 increases only from 55.7% to 55.9%). This suggests that while larger capacities enhance hit rates and performance, the benefits plateau due to data distribution constraints, indicating an optimal range (*e.g.*, 40-80) where further increases yield limited returns.

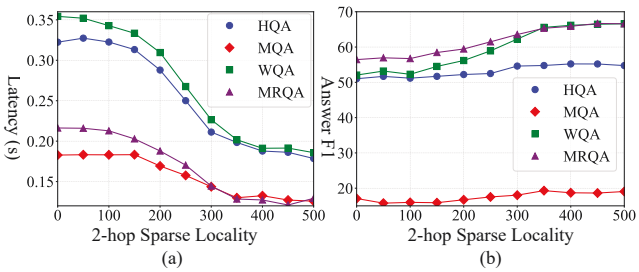

Figure 3: Impact of query locality on retrieval latency and answer F1 on four datasets with approximate dense retriever.

4.6 Query Locality Analysis

To validate the effectiveness of high query locality in caching, we conducted comparative experiments on four datasets using Approximate Dense Retrieval under varying 2-hop sparse locality, as shown in Figure 3. We systematically adjusted the locality parameter (from 0 to 500 in increments of 50) to organize the test data. As illustrated in Figure 3(a), latency exhibits a decreasing trend as 2-hop sparse locality increases, with the rate of decline being initially rapid before slowing down. This demonstrates that higher locality indeed improves cache hit rates and accelerates cache performance. In the early stages of locality improvement, the gains are modest, indicating suboptimal cache utilization. However, once locality reaches a certain threshold, cache hit rates rise sharply, leading to a significant reduction in access latency. In the saturation phase, further increases in locality cause the working set to exceed cache capacity, resulting in diminishing returns for hit rates. Figure 3(b) reveals that answer F1 scores follow a sigmoid curve as 2-hop sparse locality increases, initially rising slowly, then accelerating, before eventually plateauing. This behavior can be attributed to three phases: (1) Initially, performance is less affected by locality improvements; (2) As hit rates increase substantially, F1 scores show marked improvement; and (3) Eventually, marginal gains diminish due to saturation effects.

5 Conclusion

In this work, we introduced Mnemosyne, a cache hit order fitting method designed to address the critical limitations of existing cache-aided RAG methods in MHQA, particularly issues with their performance due to low cache hit rates. Our approach tackles this by improving cache utilization and broadening hit condition. Specifically, the cache-aware order fitting strategy optimizes cache utilization by strategically making semantically similar queries closer. Complementing this, the multi-granularity caching storage mechanism provide a more flexible hit condition by caching sub-question-document pairs and entity-documents pairs. Through empirical evaluations across four MHQA benchmarks and three types of retrievers, Mnemosyne consistently demonstrated significant performance gains and its ability to provide a superior efficiency-effectiveness trade-off for accelerating multi-step retrieval in RAG with limited-capacity caches. (Cai et al. 2024)

Acknowledgments

This work was supported in part by the Shanghai Municipal Education Commission Artificial Intelligence Plan under Grant Z2024-119.

References

Bergman, S. A.; Ji, Z.; Kermarrec, A.-M.; Petrescu, D.; Pires, R.; Randl, M.; and de Vos, M. 2025. Leveraging Approximate Caching for Faster Retrieval-Augmented Generation. In *Proceedings of the 5th Workshop on Machine Learning and Systems*, 66–73.

Cai, C.; Wang, Z.; Gao, J.; Liu, W.; Lu, Y.; Zhang, R.; and Yap, K.-H. 2024. Empowering Large Language Model for Continual Video Question Answering with Collaborative Prompting. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 3921–3932. Miami, Florida, USA: Association for Computational Linguistics.

Chan, B. J.; Chen, C.-T.; Cheng, J.-H.; and Huang, H.-H. 2025. Don't Do RAG: When Cache-Augmented Generation is All You Need for Knowledge Tasks. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, 893–897.

Chen, J.; Xiao, S.; Zhang, P.; Luo, K.; Lian, D.; and Liu, Z. 2024. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *ACL*, 2318–2335.

Chen, K.; Chen, Q.; Zhou, J.; Tao, X.; Ding, B.; Xie, J.; Xie, M.; Li, P.; and Feng, Z. 2025. Enhancing Uncertainty Modeling with Semantic Graph for Hallucination Detection. *AAAI*, 39(22): 23586–23594.

Chu, Z.; Chen, J.; Chen, Q.; Wang, H.; Zhu, K.; Du, X.; Yu, W.; Liu, M.; and Qin, B. 2024. BeamAggR: Beam Aggregation Reasoning over Multi-source Knowledge for Multi-hop Question Answering. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *ACL*, 1229–1248.

Coleman, B.; Segarra, S.; Smola, A. J.; and Shrivastava, A. 2022. Graph Reordering for Cache-Efficient Near Neighbor Search. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *NeurIPS*, volume 35, 38488–38500.

Deng, Z.; Zhu, Y.; Chen, Y.; Qi, Q.; Witbrock, M.; and Riddle, P. 2022a. Prompt-based Conservation Learning for Multi-hop Question Answering. In Calzolari, N.; Huang, C.-R.; Kim, H.; Pustejovsky, J.; Wanner, L.; Choi, K.-S.; Ryu, P.-M.; Chen, H.-H.; Donatelli, L.; Ji, H.; Kurohashi, S.; Paggio, P.; Xue, N.; Kim, S.; Hahm, Y.; He, Z.; Lee, T. K.; Santus, E.; Bond, F.; and Na, S.-H., eds., *Proceedings of the 29th International Conference on Computational Linguistics*, 1791–1800. Gyeongju, Republic of Korea: International Committee on Computational Linguistics.

Deng, Z.; Zhu, Y.; Chen, Y.; Witbrock, M.; and Riddle, P. 2022b. Interpretable AMR-Based Question Decomposition for Multi-hop Question Answering. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 4093–4099. International Joint Conferences on Artificial Intelligence Organization. Main Track.

D'Souza, J.; Babaei Giglou, H.; and Münch, Q. 2025. YESciEval: Robust LLM-as-a-Judge for Scientific Question Answering. In Che, W.; Nabende, J.; Shutova, E.; and Pilehvar, M. T., eds., *ACL*, 13749–13783.

Hamza, A.; , A.; Ahn, Y. H.; Lee, S.; and Kim, S. T. 2025. LLaVA Needs More Knowledge: Retrieval Augmented Natural Language Generation with Knowledge Graph for Explaining Thoracic Pathologies. *AAAI*, 39(3): 3311–3319.

Ho, X.; Duong Nguyen, A.-K.; Sugawara, S.; and Aizawa, A. 2020. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In Scott, D.; Bel, N.; and Zong, C., eds., *COLING*, 6609–6625.

Honnibal, M.; and Montani, I. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Jeong, S.; Baek, J.; Cho, S.; Hwang, S. J.; and Park, J. 2024. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. In Duh, K.; Gomez, H.; and Bethard, S., eds., *NAACL*, 7036–7050.

Jha, R.; Wang, B.; Günther, M.; Mastrapas, G.; Sturua, S.; Mohr, I.; Koukounas, A.; Wang, M. K.; Wang, N.; and Xiao, H. 2024. Jina-ColBERT-v2: A General-Purpose Multilingual Late Interaction Retriever. In Sälevä, J.; and Owodunni, A., eds., *Proceedings of the Fourth Workshop on Multilingual Representation Learning (MRL 2024)*, 159–166.

Jiang, Z.; Araki, J.; Ding, H.; and Neubig, G. 2022. Understanding and Improving Zero-shot Multi-hop Reasoning in Generative Question Answering. In Calzolari, N.; Huang, C.-R.; Kim, H.; Pustejovsky, J.; Wanner, L.; Choi, K.-S.; Ryu, P.-M.; Chen, H.-H.; Donatelli, L.; Ji, H.; Kurohashi, S.; Paggio, P.; Xue, N.; Kim, S.; Hahm, Y.; He, Z.; Lee, T. K.; Santus, E.; Bond, F.; and Na, S.-H., eds., *Proceedings of the 29th International Conference on Computational Linguistics*, 1765–1775. Gyeongju, Republic of Korea: International Committee on Computational Linguistics.

Jin, C.; Zhang, Z.; Jiang, X.; Liu, F.; Liu, X.; Liu, X.; and Jin, X. 2024. RAGCache: Efficient Knowledge Caching for Retrieval-Augmented Generation. arXiv:2404.12457.

Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J.; Zhang, H.; and Stoica, I. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, 611–626.

Liang, Y.; Zhang, Y.; Zhang, Z.; Zhao, Y.; Xiang, L.; Zong, C.; and Zhou, Y. 2025. Single-to-mix Modality Alignment with Multimodal Large Language Model for Document Image Machine Translation. In Che, W.; Nabende, J.; Shutova, E.; and Pilehvar, M. T., eds., *ACL*, 12391–12408.

Lin, X.; Huang, Z.; Zhang, Z.; Zhou, J.; and Chen, E. 2025. Explore What LLM Does Not Know in Complex Question Answering. *AAAI*, 39(23): 24585–24594.

Park, S.; Namkoong, H.; Choi, B.; Sullivan, M. B.; and Kim, J. 2024. CacheCraft: Enhancing GPU Performance under Memory Protection through Reconstructed Caching. In *Proceedings of the 2024 57th IEEE/ACM International Symposium on Microarchitecture*, MICRO '24, 324–337.

Robertson, S.; and Zaragoza, H. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.*, 3(4): 333–389.

Sahu, G.; Vechtomova, O.; and Laradji, I. H. 2025. A Guide To Effectively Leveraging LLMs for Low-Resource Text Summarization: Data Augmentation and Semi-supervised Approaches. In Chiruzzo, L.; Ritter, A.; and Wang, L., eds., *NAACL*, 1584–1603.

Shi, Z.; Zhang, S.; Sun, W.; Gao, S.; Ren, P.; Chen, Z.; and Ren, Z. 2024. Generate-then-Ground in Retrieval-Augmented Generation for Multi-hop Question Answering. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *ACL*, 7339–7353.

Su, W.; Tang, Y.; Ai, Q.; Wu, Z.; and Liu, Y. 2024. DRAGIN: Dynamic Retrieval Augmented Generation based on the Real-time Information Needs of Large Language Models. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *ACL*, 12991–13013.

Tang, Y.; and Yang, Y. 2024. MultiHop-RAG: Benchmarking Retrieval-Augmented Generation for Multi-Hop Queries. arXiv:2401.15391.

Trivedi, H.; Balasubramanian, N.; Khot, T.; and Sabharwal, A. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10: 539–554.

Trivedi, H.; Balasubramanian, N.; Khot, T.; and Sabharwal, A. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *ACL*, 10014–10037.

Wang, L.; Wu, L.; Song, S.; Wang, Y.; Gao, C.; and Wang, K. 2025a. Distilling Structured Rationale from Large Language Models to Small Language Models for Abstractive Summarization. *AAAI*, 39(24): 25389–25397.

Wang, X.; He, J.; Chen, L.; Haffari, G.; Wang, Y.; Yang, Z.; Meng, X.; Pan, K.; and Sui, Z. 2025b. SG-FSM: A Self-Guiding Zero-Shot Prompting Paradigm for Multi-Hop Question Answering Based on Finite State Machine. In Chiruzzo, L.; Ritter, A.; and Wang, L., eds., *NAACL*, 6025–6037.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; ichter, b.; Xia, F.; Chi, E.; Le, Q. V.; and Zhou, D. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *NeurIPS*, volume 35, 24824–24837.

Xiao, S.; Liu, Z.; Zhang, P.; Muennighoff, N.; Lian, D.; and Nie, J.-Y. 2024. C-Pack: Packed Resources For General Chinese Embeddings. In *SIGIR*, SIGIR '24, 641–649.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In Riloff, E.; Chiang, D.; Hockenmaier, J.; and Tsujii, J., eds., *EMNLP*, 2369–2380.

Ye, L.; Yu, L.; Lei, Z.; Chen, Q.; Zhou, J.; and He, L. 2025. Optimizing Question Semantic Space for Dynamic Retrieval-Augmented Multi-hop Question Answering. In Che, W.; Nabende, J.; Shutova, E.; and Pilehvar, M. T., eds., *ACL*, 17814–17824.

Yujian, L.; and Bo, L. 2007. A Normalized Levenshtein Distance Metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6): 1091–1095.

Zhang, B.; and Soh, H. 2024. Extract, Define, Canonicalize: An LLM-based Framework for Knowledge Graph Construction. In *EMNLP*, 9820–9836.

Zhang, C.; Feng, Z.; Zhang, Z.; Qiang, J.; Xu, G.; and Li, Y. 2025a. Is LLMs Hallucination Usable? LLM-based Negative Reasoning for Fake News Detection. *AAAI*, 39(1): 1031–1039.

Zhang, J.; Zhang, H.; Zhang, D.; Yong, L.; and Huang, S. 2024a. End-to-End Beam Retrieval for Multi-Hop Question Answering. In Duh, K.; Gomez, H.; and Bethard, S., eds., *NAACL*, 1718–1731.

Zhang, T.; Li, D.; Chen, Q.; Wang, C.; and He, X. 2025b. BELLE: A Bi-Level Multi-Agent Reasoning Framework for Multi-Hop Question Answering. In Che, W.; Nabende, J.; Shutova, E.; and Pilehvar, M. T., eds., *ACL*, 4184–4202.

Zhang, X.; Rajabi, N.; Duh, K.; and Koehn, P. 2023. Machine Translation with Large Language Models: Prompting, Few-shot Learning, and Fine-tuning with QLoRA. In Koehn, P.; Haddow, B.; Kocmi, T.; and Monz, C., eds., *Proceedings of the Eighth Conference on Machine Translation*, 468–481.

Zhang, Z.; Zhu, A.; Yang, L.; Xu, Y.; Li, L.; Phothilimthana, P. M.; and Jia, Z. 2024b. Accelerating iterative retrieval-augmented language model serving with speculation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24.