GEAR: Graph-enhanced Agent for Retrieval-augmented Generation

Anonymous ACL submission

Abstract

Retrieval-augmented Generation (RAG) relies on effective retrieval capabilities, yet traditional sparse and dense retrievers inherently struggle with multi-hop retrieval scenarios. In this paper, we introduce GEAR, a system that advances RAG performance through two key innovations: (i) an efficient graph expansion mechanism that augments any conventional base retriever, such as BM25, and (ii) an agent framework that incorporates the resulting graph-based retrieval into a multi-step retrieval framework. Our evaluation demonstrates GEAR's superior retrieval capabilities across three multi-hop question answering datasets. Notably, our system achieves state-of-the-art results with improvements exceeding 10% on the challenging MuSiQue dataset, while consuming fewer tokens and requiring fewer iterations than existing multi-step retrieval systems.

1 Introduction

Retrieval-augmented Generation (RAG) has enhanced the performance of Large Language Models (LLMs) (OpenAI, 2023) in Question Answering (QA) tasks (Lewis et al., 2020). While effective for simple queries, multi-hop QA presents a more complex challenge, requiring reasoning across several passages or documents. Consider the example in Table 1, where finding the correct answer requires building a 3-hop reasoning chain starting from the question's main entity (i.e. "Stephen Curry").

What year did the father of Stephen Curry join the team from which he started his college basketball career?		
Stephen Curry	$\xrightarrow{\text{son of}}$	Dell Curry
Dell Curry	$\xrightarrow{\text{college team}}$	Virginia Tech
Dell Curry	$\xrightarrow{\text{college start}}$	1982 (answer)

Table 1: Multi-hop question (top) involving a reasoning chain (bottom) extending across several entities.

To address these complex reasoning requirements, there has been a growing interest in leveraging graph representations of passages to bridge the semantic gap inherent in multi-hop questions (Fang et al., 2024; Li et al., 2024; Edge et al., 2024; Gutierrez et al., 2024; Liang et al., 2024). By extracting entities, atomic facts, or semantic triples (Li et al., 2024; Fang et al., 2024; Gutierrez et al., 2024), these graphs can establish more direct pathways for multi-hop reasoning and associate relevant passages. For instance, HippoRAG extracts triples from passages and employs a variant of PageRank for passage retrieval, conditioned on this extracted knowledge graph (Gutierrez et al., 2024). Furthermore, GraphReader uses an LLM agent, with access to graph-navigating operations for exploring the resulting graph (Li et al., 2024). TRACE relies on an LLM to iteratively select triples to construct reasoning chains, which are then used for grounding the answer generation directly, or for filtering out irrelevant documents from an original set of retrieved results (Fang et al., 2024). However, recursively prompting LLMs to traverse graphs can be computationally expensive, especially as the search space expands.

In this paper, we present GEAR, a **G**raph-**e**nhanced **A**gent for **R**etrieval-augmented generation. During the offline stage, we *align* an index of passages with an index of triples extracted from these passages. With such alignment, passages are intermediately connected through graphs of triples. GEAR contains a graph-based passage retrieval component referred to as SyncGE. Unlike previous works that rely on expensive LLM calls for graph exploration, we leverage an LLM for locating initial nodes (triples) and employ a generic semantic model to expand the sub-graph of triples by exploring diverse beams of triples. Furthermore, GEAR utilises multi-hop contexts retrieved by SyncGE and constructs a memory that summarises information for multi-step retrieval.

Our work refines the neurobiology-inspired paradigm proposed by Gutierrez et al. by modeling the communication between the hippocampus and neocortex during episodic memory formation. In our design, an array of *proximal triples* functions as a memory gist learned through the hippocampus within one or a few shots (iterations), which is then projected back to the neocortex for later recall stages (Hanslmayr et al., 2016; Griffiths et al., 2019). We highlight the complementary potential between our graph retrieval approach and an LLM, which, within our system, emulates the synergy between the hippocampus and neocortex, offering insights from a biomimetic perspective.

We evaluate the retrieval performance of GEAR on three multi-hop QA benchmarks: MuSiQue, HotpotQA, and 2WikiMultihopQA. GEAR pushes the state of the art, achieving significant improvements in both single- and multi-step retrieval settings, with gains exceeding 10% on the most challenging MuSiQue dataset. Furthermore, we demonstrate that our framework can address multi-hop questions in fewer iterationswhile consuming significantly fewer LLM tokens. Even with a single iteration, GEAR offers a more efficient alternative to other iterative retrieval methods, such as HippoRAG w/ IRCoT. Our contributions can be summarised as follows:

- We introduce a novel graph-based retriever, SyncGE, which leverages an LLM for locating initial nodes for graph exploration and subsequently expands them by diversifying beams of triples that link multi-hop passages.
- We incorporate this graph retrieval method within an LLM-based agent framework, materialising GEAR, achieving state-of-the-art retrieval performance across three datasets.
- We conduct comprehensive experiments showcasing the synergetic effects between our proposed graph-based retriever and the LLM within the GEAR framework.

2 Related Work

Our work draws inspiration from two branches of research: (i) retrieval-augmented models for QA and (ii) multi-hop QA using combinations of LLMs with graphical structures.

2.1 Retrieval-augmented Models for QA

Lewis et al. first showcased the benefits of augmenting language models' input context with relevant passages. Recent work by Wang et al. and Shen et al. explores query expansion approaches, generating pseudo-documents from language models to expand the content of original queries. Subsequent frameworks, beginning with IRCoT, have investigated the interleaving of retrieval and prompting steps, allowing each step to iteratively guide and refine the other (Trivedi et al., 2023; Jiang et al., 2023; Su et al., 2024).

2.2 Multi-hop QA with LLMs and Graphs

Several architectures have introduced an offline indexing phase to form hierarchical passage summaries (Chen et al., 2023; Sarthi et al., 2024; Edge et al., 2024). However, summarization must be repeated when adding new data, making knowledge base updates computationally expensive. Recent approaches have leveraged structured knowledge to address multi-hop QA challenges with LLMs (Park et al., 2023; Fang et al., 2024; Li et al., 2024; Gutierrez et al., 2024; Liang et al., 2024; Wang et al., 2024). GraphReader, TRACE and HippoRAG propose offline methods for extracting entities and atomic facts or semantic triples from passages (Li et al., 2024; Fang et al., 2024; Gutierrez et al., 2024). This allows chunks containing the same or neighbouring entities to construct a graph of indexed passages. TRACE relies on an LLM to iteratively select triples for reasoning chains, which ground answer generation or filter retrieved results. However, its search space is limited by pre-filtered candidate lists for each query. Li et al. employ an LLM agent that selects from a set of predefined actions to traverse knowledge graph nodes in real time given an input question. Liang et al. later introduced additional graph standardisation, including instance-to-concept linking and semantic relation completion. However, this approach heavily depends on associating triples with pre-defined concepts for logical form-based retrieval.

HippoRAG leverages an alignment of passages and extracted triples to retrieve passages based on the Personalized PageRank algorithm (Gutierrez et al., 2024). While achieving improvements for single- and multi-step retrieval (when coupled with IRCoT (Trivedi et al., 2023)), it remains agnostic to the semantic relationships of extracted triples. In this paper, we leverage a similar alignment of passages and extracted triples but introduce a new graph-based retrieval framework that uses a small semantic model for exploring multi-hop relationships. Our framework considers the contributions of all triple elements participating in reasoning chains, offering a more robust solution for associating questions with triple reasoning chains.

3 Preliminaries

Let $\mathbf{C} = \{c_1, c_2, \ldots, c_C\}$ be an index of passages and $\mathbf{T} = \{t_1, t_2, \ldots, t_T : t_j = (s_j, p_j, o_j)\}$ be another index representing a set of triples associated with the passages in $\mathbf{C}$ s.t. $\forall t_j \in \mathbf{T} \exists! \, c_i \in \mathbf{C}$, where s_j, p_j and o_j the respective subject, predicate and object of the j-th triple. In this setup, each triple is uniquely linked to exactly one passage, and a passage can potentially be associated with multiple triples.

Given an input query $\mathbf{q}$ and an index of interest $\mathbf{R} = \{r_1, \ldots, r_R\}$, retrieving items from $\mathbf{R}$ relevant to $\mathbf{q}$ can be achieved by using a base retrieval function $h^k_{\text{base}}(\mathbf{q}, \mathbf{R}) \subseteq \mathbf{R}$ that returns a ranked list of k items from $\mathbf{R}$ in descending order, according to a retrieval score. BM25 or a conventional dense retriever can serve as a base retrieval function, without requiring any multi-hop capabilities.

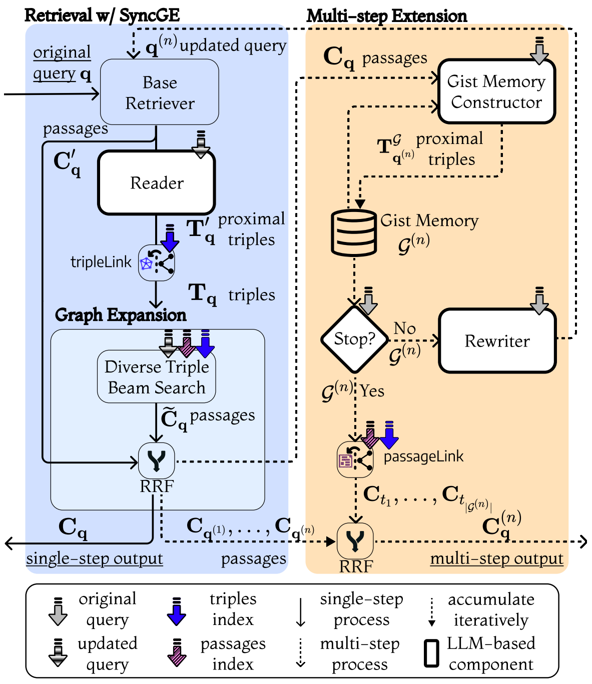
Our goal is to retrieve relevant passages from $\mathbf{C}$ that enable a retrieval-augmented model to answer multi-hop queries (Lewis et al., 2020). To this end, we introduce GEAR, which is a graph-enhanced framework of retrieval agent (see Figure 1).

4 Retrieval with Synchronised Graph Expansion

Given an input query $\mathbf{q}$, let $\mathbf{C}'_{\mathbf{q}} = h^k_{\text{base}}(\mathbf{q}, \mathbf{C})$ be a list of passages returned by the base retriever. Given this list, $\mathbf{C}'_{\mathbf{q}}$, our goal is to derive relevant multi-hop contexts (passages) by retrieving a sub-graph of triples that interconnect their source passages. Two challenges arise in materialising such sub-graph retrieval: (i) locating initial triples (i.e. starting nodes) $\mathbf{T}_{\mathbf{q}}$, and (ii) expanding the graph from these initial triples while reducing the search space. The following sections address these challenges within GEAR.

4.1 Knowledge Synchronisation

We describe a knowledge **Sync**hronisation (**Sync**) process to locate initial nodes for graph expansion. We first employ an LLM to `read` $\mathbf{C}'_{\mathbf{q}}$ (see Appendix J.2) and summarise knowledge triples



Retrieval w/ SyncGE
Multi-step Extension
original query $\mathbf{q}$
$\mathbf{q}^{(n)}$ updated query
Base Retriever
passages
$\mathbf{C}'_{\mathbf{q}}$
Reader
$\mathbf{T}'_{\mathbf{q}}$ proximal triples
tripleLink
$\mathbf{T}_{\mathbf{q}}$ triples
Graph Expansion
Diverse Triple Beam Search
$\widetilde{\mathbf{C}}_{\mathbf{q}}$ passages
RRF
$\mathbf{C}_{\mathbf{q}}$
single-step output
$\mathbf{C}_{\mathbf{q}^{(1)}}, \ldots, \mathbf{C}_{\mathbf{q}^{(n)}}$
passages
$\mathbf{C}_{\mathbf{q}}$ passages
Gist Memory Constructor
$\mathbf{T}^{\mathcal{G}}_{\mathbf{q}^{(n)}}$ proximal triples
Gist Memory $\mathcal{G}^{(n)}$
Stop?
No
$\mathcal{G}^{(n)}$
Rewriter
$\mathcal{G}^{(n)}$ Yes
passageLink
$\mathbf{C}_{t_1}, \ldots, \mathbf{C}_{t_{|\mathcal{G}^{(n)}|}}$
$\mathbf{C}^{(n)}_{\mathbf{q}}$
RRF
multi-step output
original query
triples index
single-step process
accumulate iteratively
updated query
passages index
multi-step process
LLM-based component

Figure 1: System Architecture

that can support answering the current query $\mathbf{q}$:

$$\mathbf{T}'_{\mathbf{q}} = \texttt{read}\left(\mathbf{C}'_{\mathbf{q}}, \mathbf{q}\right). \quad (1)$$

$\mathbf{T}'_{\mathbf{q}}$ is a collection of triples to which we refer as *proximal triples*. Initial nodes $\mathbf{T}_{\mathbf{q}}$ for graph expansion are identified by linking each triple in $\mathbf{T}'_{\mathbf{q}}$ to a triple in $\mathbf{T}$, using the `tripleLink` function:

$$\mathbf{T}_{\mathbf{q}} = \left\{ t_i | t_i = \texttt{tripleLink}(t'_i) \; \forall t'_i \in \mathbf{T}'_{\mathbf{q}} \right\}. \quad (2)$$

The implementation of `tripleLink` can vary. However, in this paper we consider it to be simply retrieving the most similar triple from $\mathbf{T}$.

4.2 Diverse Triple Beam Search

We borrow the idea of constructing reasoning triple chains (Fang et al., 2024) for expanding the graph, and present a retrieval algorithm: *Diverse Triple Beam Search* (see Alg. 1).

We maintain top-b sequences (beams) of triples and the scores at each step are determined by a scoring function. In this paper, we focus on leveraging a dense embedding model to compute the cosine similarity between embeddings of the query and a candidate sequence of triples, leaving other implementations of the scoring function for future work (see Section 9).

Considering all possible triple extensions at each step, in a Viterbi decoding fashion, would be intractable due to the size of $\mathbf{T}$. Consequently, we

Algorithm 1 Diverse Triple Beam Search

```
Input: q: query
       b: beam size
       l: maximum length
       score(·,·): scoring function
       {t_1, t_2, ..., t_n}: initial triples
       γ: hyperparameter for diversity
1:  $B_0 \leftarrow [\,]$
2:  for $t \in \{t_1, t_2, ..., t_n\}$ do
3:      $s \leftarrow \text{score}(\mathbf{q}, [t])$
4:      $B_0.\text{add}(\langle s, [t] \rangle)$
5:  $B_0 \leftarrow \text{top}(B_0, b)$
6:  for $i \in \{1, \ldots, l-1\}$ do
7:      $B \leftarrow [\,]$
8:      for $\langle s, T \rangle \in B_{i-1}$ do
9:          $V \leftarrow [\,]$
10:         for $t \in \text{get\_neighbours}(T.\text{last}())$ do
11:             if $\text{exists}(t, B_{i-1})$ then
12:                 continue
13:             $s' \leftarrow s + \text{score}(\mathbf{q}, T \circ t)$ # concat
14:             $V.\text{add}(\langle s', T \circ t \rangle)$
15:         $\text{sort}(V, \text{descending})$
16:         for $n \in \{0, \ldots, V.\text{length}() - 1\}$ do
17:             $\langle s', T \circ t \rangle \leftarrow V[n]$
18:             $s' \leftarrow s' \times e^{-\frac{\min(n,\gamma)}{\gamma}}$
19:             $B.\text{add}(\langle s', T \circ t \rangle)$
20:     $B_i \leftarrow \text{top}(B, b)$
21: return $B_i$
```

define the neighbourhood of a triple as the set of triples with shared head or tail entities (i.e. get_neighbours in Alg. 1). During each expansion step, we only consider neighbours of the last triple in the sequence, and avoid selecting previously visited triples (i.e. exists in Alg. 1) to further reduce the search space.

While regular beam search can reduce the search space, it is prone to producing high-likelihood sequences that differ only slightly from one another (Ippolito et al., 2019; Vijayakumar et al., 2018). Our algorithm increases the diversity across beams to improve the recall for retrieval. In detail, for each beam, we sort candidate sequences extended from that beam in descending order, and weight their scores based on their relative positions. Candidate sequences that are ranked lower, within a beam, will receive smaller weights. Consequently, the resulting top-b beams at each step are less likely to share the same starting sequence.

The top-b returned sequences are flattened in a breadth-first order. Each triple in the resulting list is then mapped to its source passage. This alignment between triples and passages is described in more detail in Section 3. Let $\widetilde{\mathbf{C}}_{\mathbf{q}}$ be the list of unique passages after alignment. The output of our graph expansion is then given by the Reciprocal Rank Fusion (RRF) (Cormack et al., 2009) of $\widetilde{\mathbf{C}}_{\mathbf{q}}$ and the initial $\mathbf{C}'_{\mathbf{q}}$ list of passages :

$$\mathbf{C}_{\mathbf{q}} = \text{RRF}\left(\widetilde{\mathbf{C}}_{\mathbf{q}}, \mathbf{C}'_{\mathbf{q}}\right). \tag{3}$$

We refer to this method for retrieving passages as <u>**Sync**</u>ronised <u>G</u>raph <u>E</u>xpansion (**SyncGE**).

5 Multi-step Extension

We further present an agentic framework that models a human-like information-seeking process through multi-turn interactions with the graph-enhanced retriever. The resulting agent is referred to as GEAR. We focus on:

- maintaining a gist memory of proximal knowledge obtained throughout the different steps
- incorporating a similar synchronisation process that summarises retrieved passages in proximal triples to be stored in this multi-turn gist memory
- determining if additional steps are needed for answering the original input question

Within this multi-turn setting, the original input question $\mathbf{q}$ is iteratively decomposed into simpler queries: $\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(n)}$, where $\mathbf{q}^{(1)} = \mathbf{q}$ and $n \in \mathbb{N}$ represents the number of the current step. For each query $\mathbf{q}^{(n)}$, we use the graph retrieval method introduced in Section 4 in order to retrieve relevant passages $\mathbf{C}_{\mathbf{q}^{(n)}}$.

5.1 Gist Memory Constructor

To facilitate the multi-step capabilities of our agent, we introduce a *gist memory*, $\mathcal{G}^{(n)}$, which is used for storing knowledge as an array of proximal triples. At the beginning of the first iteration, the gist memory is empty. During the n-th iteration, similar to the knowledge synchronisation module explained in Section 4.1, we employ an LLM to read a collection of retrieved paragraphs $\mathbf{C}_{\mathbf{q}^{(n)}}$ and summarise their content with proximal triples:

$$\mathbf{T}^{\mathcal{G}}_{\mathbf{q}^{(n)}} = \begin{cases} \texttt{read}\left(\mathbf{C}_{\mathbf{q}^{(n)}}, \mathbf{q}\right), & \text{if } n = 1 \\ \texttt{read}\left(\mathbf{C}_{\mathbf{q}^{(n)}}, \mathbf{q}, \mathcal{G}^{(n-1)}\right), & \text{if } n \geq 2 \end{cases} \quad (4)$$

Apart from the first iteration where Eq. 1 and 4 are identical, the inclusion of the memory in the `read` operation differentiates the construction of proximal triples produced at the subsequent steps compared to the ones from Eq. 1. $\mathcal{G}^{(n)}$ maintains the aggregated content of proximal triples s.t.

$$\mathcal{G}^{(n)} = \left[\mathbf{T}^{\mathcal{G}}_{\mathbf{q}^{(1)}} \circ \cdots \circ \mathbf{T}^{\mathcal{G}}_{\mathbf{q}^{(n)}}\right], \quad (5)$$

where $\circ$ defines the concatenation operation. The triple memory serves as a concise representation of all the accumulated evidence, up to the n-th step.

We believe the process introduced by the `read` step along with the information storage paradigm served by the gist memory, aligns well with the communication between the hippocampus and neocortex. The combination of the two establishes the synergetic behaviour between our graph retriever and the LLM that we seek to achieve within GEAR.

5.2 Reasoning for Termination

After updating $\mathcal{G}^{(n)}$, we assess whether it contains sufficient evidence to answer the original question through an LLM reasoning step:

$$\mathbf{a}^{(n)}, \mathbf{r}^{(n)} = \texttt{reason}(\mathcal{G}^{(n)}, \mathbf{q}), \quad (6)$$

where $\mathbf{a}^{(n)}$ denotes the query's answerability given the evidence in $\mathcal{G}^{(n)}$, and $\mathbf{r}^{(n)}$ represents the relevant reasoning. When the query is deemed answerable, the system concludes its iterative process.

5.3 Query Re-writing

The query re-writing process leverages an LLM that incorporates three key inputs: the original query $\mathbf{q}$, the accumulated memory, and crucially, the reasoning output $\mathbf{r}^{(n)}$ from the previous step. This process can be formally expressed as:

$$\mathbf{q}^{(n+1)} = \texttt{rewrite}\left(\mathcal{G}^{(n)}, \mathbf{q}, \mathbf{r}^{(n)}\right), \quad (7)$$

where $\mathbf{q}^{(n+1)}$ represents the updated query, which serves as input for the retriever in the next iteration.

5.4 After Termination

GEAR aims to return a single ranked list of passages. Given the final gist memory $\mathcal{G}^{(n)}$ upon termination, we link each proximal triple in $\mathcal{G}^{(n)}$ to a list of passages as follows:

$$\mathbf{C}_{t_j} = \texttt{passageLink}\left(t_j, k\right), \quad (8)$$

where $j \in \{1, \ldots, |\mathcal{G}^{(n)}|\}$. Similar to `tripleLink`, `passageLink` is implemented by retrieving passages with a triple as the query (see Appendix B.2). The final list of passages returned by GEAR is the RRF of the resulting linked passages and passages retrieved across iterations:

$$\mathbf{C}^{(n)}_{\mathbf{q}} = \mathrm{RRF}\big(\mathbf{C}_{t_1}, \ldots, \mathbf{C}_{t_{|\mathcal{G}^{(n)}|}}, \mathbf{C}_{\mathbf{q}^{(1)}}, \ldots, \mathbf{C}_{\mathbf{q}^{(n)}}\big). \quad (9)$$

All relevant prompts for the `read`, `reason` and `rewrite` steps are provided in Appendix J.2.

6 Experimental Setup

We evaluate our framework on three open-domain multi-hop QA datasets: **MuSiQue** (Trivedi et al., 2022), **HotpotQA** (Yang et al., 2018), and **2Wiki-MultiHopQA** (2Wiki) (Ho et al., 2020). For MuSiQue and 2Wiki, we use the data provided in the IRCoT paper (Trivedi et al., 2023) which includes the full corpus, while for HotpotQA, we follow the same setting as HippoRAG (Gutierrez et al., 2024) to limit experimental costs. More details are provided in Appendix A.

We measure both retrieval and QA performance, with our primary contributions focused on the retrieval component. For retrieval evaluation, we use Recall@k (R@k) for $k \in \{5, 10, 15\}$, showing the percentage of questions where the correct entries are found within the top-k retrieved passages. We include an analysis about the selected recall ranks in Appendix C. Following standard practices, QA performance is evaluated with Exact Match (EM) and F1 scores (Trivedi et al., 2023).

6.1 Baselines

We evaluate GEAR against strong, multi-step baselines, including IRCoT (Trivedi et al., 2023) and HippoRAG w/ IRCoT (Gutierrez et al., 2024), which, similar to our framework, combines graph retrieval and a multi-step agent. To demonstrate our graph retriever's (i.e. SyncGE) benefits, we evaluate it against several stand-alone, single-step retrievers: (i) BM25, (ii) Sentence-BERT (SBERT),

	Retriever	MuSiQue			2Wiki			HotpotQA		
		R@5	R@10	R@15	R@5	R@10	R@15	R@5	R@10	R@15
Single-step Retrieval	ColBERTv2	39.4	44.8	47.7	59.1	64.3	66.2	79.3	87.1	90.1
	HippoRAG	41.0	47.0	51.4	**75.1**	**83.2**	**86.4**	79.8	89.0	92.4
	BM25	33.8	38.5	41.3	59.5	62.7	64.1	74.2	83.6	86.3
	+ NaiveGE	37.5	45.5	48.4	65.0	70.7	71.8	79.1	89.1	91.9
	+ SyncGE	44.7	52.6	57.4	70.5	76.1	79.3	87.4	93.0	94.0
	SBERT	31.1	37.9	41.6	41.2	48.1	51.5	72.1	79.3	84.0
	+ NaiveGE	32.2	41.4	45.4	45.1	54.0	57.3	76.1	84.7	88.8
	+ SyncGE	41.6	51.3	54.2	54.8	64.9	70.7	84.1	89.6	92.8
	Hybrid	39.9	46.3	49.1	60.0	65.8	66.6	77.8	85.8	89.7
	+ NaiveGE	41.8	49.4	53.0	63.0	70.8	72.6	80.6	89.4	92.7
	+ SyncGE	**48.7**	**57.7**	**61.2**	72.6	80.9	82.4	**87.4**	**93.3**	**95.2**
Multi-step Retrieval	IRCoT (BM25)	46.1	54.9	57.9	67.9	75.5	76.1	87.0	92.6	92.9
	IRCoT (ColBERTv2)	47.9	54.3	56.4	60.3	86.6	69.7	86.9	92.5	92.8
	HippoRAG w/ IRCoT	48.8	54.5	58.9	82.9	90.6	93.0	90.1	94.7	95.9
	GEAR	**58.4**	**67.6**	**71.5**	**89.1**	**95.3**	**95.9**	**93.4**	**96.8**	**97.3**

Table 2: Retrieval performance for single- and multi-step retrievers on MuSiQue, 2Wiki, and HotpotQA. Results are reported using Recall@k (R@k) metrics for $k \in \{5, 10, 15\}$.

(iii) a hybrid approach combining BM25 and SBERT through RRF and (iv) HippoRAG. Following Gutierrez et al., we refer to the single-step setup when multiple LLM iterations are not supported.

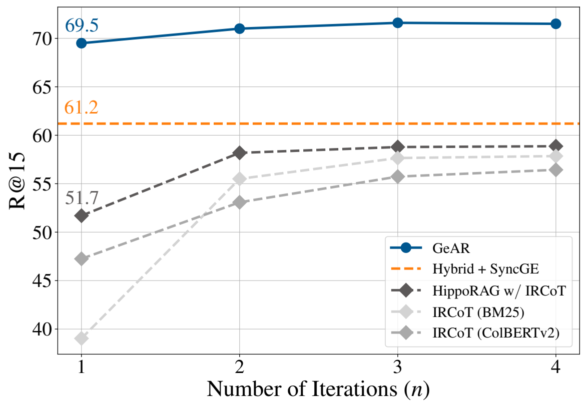


Figure 2: R@15 over 4 iterations on MuSiQue. Recall is computed at each iteration using the cumulative set of retrieved documents, with prior recall values carried forward for questions that terminated in earlier iterations. The horizontal line indicates the single-step performance of Hybrid + SyncGE.

6.2 Implementation Details

We reproduce HippoRAG and IRCoT using the code provided by Gutierrez et al.. To ensure fair comparisons, we employ GPT-4o mini (`gpt-4o-mini-2024-07-18`) for all methods that require an LLM as well as their corresponding triple extraction. The temperature is set to 0. Our triple extraction prompt (in Appendix J.1) is inspired by Gutierrez et al.. We run QA experiments using the prompts provided in Appendix J.3.

In addition to our proposed SyncGE, we consider a more *naive* implementation of GE (i.e. NaiveGE) to evaluate the performance when no LLM is involved and further demonstrate the effectiveness of synchronisation. In NaiveGE, we input all triples associated with $\mathbf{C}'_\mathbf{q}$ (see Section 4) for diverse triple beam search. Comprehensive implementation details are provided in Appendices B–C. Additionally, Appendix E provides more experiments evaluating GEAR with varying configurations.

7 Results

GEAR demonstrates state-of-the-art performance in multi-step retrieval The multi-step results in Table 2 show that our agent-based approach to multi-step retrieval is highly effective, achieving state-of-the-art results across all datasets. While we see significant improvements on saturated datasets like 2Wiki and HotpotQA, GEAR especially excels on MuSiQue, delivering performance gains of over 10% compared to competitors.

SyncGE contributes to state-of-the-art performance in single-step retrieval As shown in the single-step section of Table 2, our proposed Hybrid + SyncGE method achieves state-of-the-art single-step retrieval performance on both MuSiQue and HotpotQA datasets. We observe consistent improvements using NaiveGE and SyncGE, outperforming HippoRAG in many setups regardless of the base retriever (i.e. sparse, dense or hybrid). Most notably, Hybrid + SyncGE surpasses HippoRAG by up to 9.8% at R@15 on MuSiQue.

Higher recall leads to higher QA performance Our analysis shows a positive correlation between recall and QA performance, aligning with the results of prior works (Gutierrez et al., 2024). As shown in Table 3, GEAR achieves the highest EM and F1 scores. A closer examination reveals interesting insights. Taking MuSiQue as an example, GEAR shows a 21% relative improvement in R@15 compared to HippoRAG w/ IRCoT, while achieving a 37% relative improvement in both EM and F1 scores. Similarly to Table 2, SyncGE outperforms HippoRAG on MuSiQue and HotpotQA.

Retriever	**MuSiQue**		**2Wiki**		**HotpotQA**	
	EM	F1	EM	F1	EM	F1
No Passages	2.6	12.5	17.2	27.9	19.5	34.3
Gold Passages	36.6	59.2	54.4	70.3	55.0	75.9
Hybrid + SyncGE	14.0	<u>27.1</u>	38.0	50.2	45.0	63.4
HippoRAG	8.2	18.2	39.8	51.8	40.1	57.6
IRCoT (BM25)	7.6	15.9	28.8	38.5	34.3	50.8
IRCoT (ColBERTv2)	12.2	24.1	32.4	43.6	45.2	63.7
HippoRAG w/ IRCoT	<u>14.2</u>	25.9	<u>45.6</u>	<u>59.0</u>	<u>49.2</u>	<u>67.9</u>
GEAR	**19.0**	**35.6**	**47.4**	**62.3**	**50.4**	**69.4**

Table 3: End-to-end QA performance using the top-5 retrieved passages. The **best** model is in bold and <u>second best</u> is underlined. The top part shows the lower and upper bounds of QA performance, while the middle and bottom sections display scores for single-step and multi-step retrievers, respectively.

8 Discussion

8.1 What makes GEAR work?

NaiveGE vs SyncGE As shown in Table 2, both graph expansion variants enhance every base retriever's performance across all datasets. The superior performance of SyncGE indicates the effectiveness of using LLMs for locating initial nodes. Notably, it surpasses HippoRAG w/ IRCoT's on MuSiQue without multiple iterations.

Diverse Triple Beam Search improves performance As shown in Table 4, our diverse triple beam search consistently outperforms standard beam search across all datasets and recall ranks. By incorporating diversity weights into beam search, we align a language modelling-oriented solution with information retrieval objectives that involve satisfying multiple information needs underlying multi-hop queries (Drosou and Pitoura, 2010).

Metric	**Dataset**	**w/ Diversity**	**w/o Diversity**
R@5	MuSiQue	**48.7**	47.0
	2Wiki	**72.6**	68.2
	HotpotQA	**87.4**	85.0
R@10	MuSiQue	**57.7**	53.9
	2Wiki	**80.9**	76.0
	HotpotQA	**93.3**	92.2
R@15	MuSiQue	**61.2**	58.4
	2Wiki	**82.4**	77.4
	HotpotQA	**95.2**	94.3

Table 4: Effects of beam search diversity on Hybrid + SyncGE across MuSiQue, 2Wiki and HotpotQA.

GEAR *mostly* nails it the first time While GEAR supports multiple iterations, Figure 2 shows that GEAR achieves strong retrieval performance in a single iteration on MuSiQue. This differentiates it from IRCoT-oriented setups that require at least 2 iterations to reach maximum performance. This can be attributed to the fact that GEAR reads (Eq. 4) multi-hop contexts and associates proximal triples in gist memory with passages, establishing synergy between our graph retriever and the LLM. We believe this mirrors the hippocampal process of forming and resolving sparse representations, where gist memories are learned in a one or few-shot manner (Hanslmayr et al., 2016). The 10% performance gap between Hybrid + SyncGE and GEAR at $n = 1$ indicates that the LLM reading and linking processes effectively approximate the hippocampus's role within our framework.

8.2 How robust is GEAR?

GEAR excels at questions of low-to-moderate complexity Figure 3 presents a detailed breakdown of retrieval performance across different hop types, including path-finding and path-following questions categorized by Gutierrez et al.. For 2-hop questions, while GEAR and HippoRAG w/ IRCoT achieve similar interquartile ranges, GEAR shows a higher mean recall, indicating superior performance on low-complexity questions. This advantage becomes more pronounced with 3-hop questions, where GEAR's entire interquartile range exceeds HippoRAG w/ IRCoT's median performance across both hop subdivisions. This demonstrates GEAR's enhanced capability in handling moderately complex questions. In addition to MuSiQue, 2Wiki and HotpotQA, we test GEAR against the *hand-picked* case study data provided by Gutierrez et al.. These include four path-finding questions across four different domains. Our find-

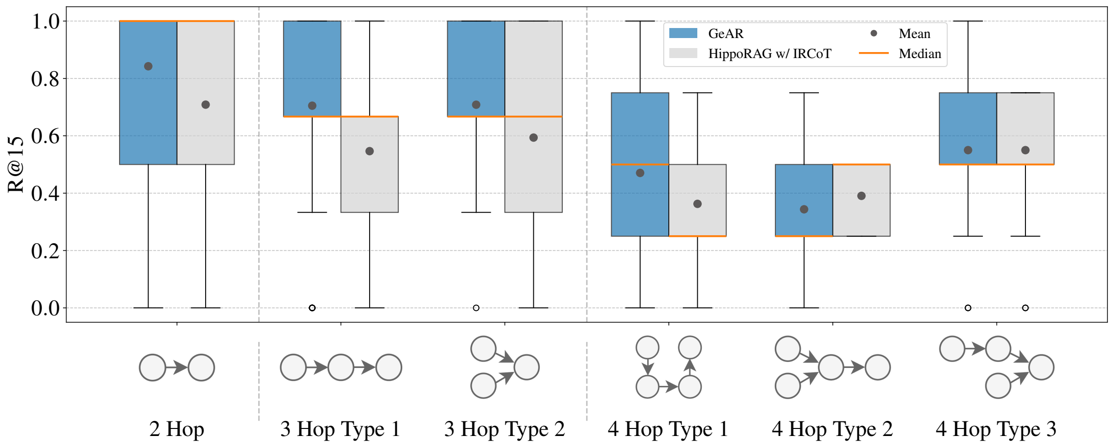


Figure 3: Analysis of R@15 performance divided by hop types on MuSiQue. The hop categorisation follows the MuSiQue documentation. Mean recall values are indicated by grey dots for each hop type.

ings (Appendix H.2) indicate that GEAR's performance is on par with HippoRAG w/ IRCoT, outperforming the competition in three out of the four cases, in terms of recall.

GEAR's performance remains consistent across chunks with varying numbers of triples Using MuSiQue, we group questions based on the average number of triples (i.e. triple density, ρ_t) associated with their golden passages, and evaluate R@15 across four ranges: (i) $\rho_t < 9$, (ii) $9 \leq \rho_t < 11$, (iii) $11 \leq \rho_t < 13$ and (iv) $13 \leq \rho_t$. Across all these ranges, the recall performance of both SyncGE and GeAR exhibits lower variation, with significantly smaller standard deviations of $1.18 \ll 2.04$ and $2.08 \ll 5.59$, respectively, compared to NaiveGE and HippoRAG w/ IRCoT. Further details are provided in Appendix G.

8.3 Is GEAR efficient?

As observed in Figure 2, GEAR requires fewer iterations than the competition to reach its maximum recall performance. Furthermore, Figure 4 shows that GEAR can act as a more efficient alternative with respect to LLM token utilisation. We note that even for a single iteration, GEAR uses fewer tokens than HippoRAG w/ IRCoT. In contrast to ours, this trend exacerbates for the competition as the number of iterations increases. These findings also reiterate the value of SyncGE, which outperforms a significantly more LLM-heavy solution on MuSiQue, using almost 2.9 million fewer tokens. Even in the case that HippoRAG w/ IRCoT runs for a single iteration it would require more than 0.7 million tokens that Hybrid + SyncGE, with a substantially lower R@15 of 51.7.

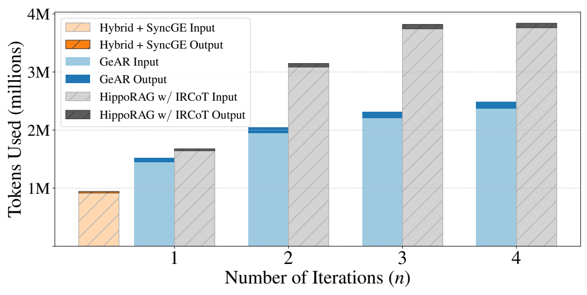


Figure 4: Progressive accumulation of input and output LLM tokens across agent iterations on MuSiQue. The Hybrid + SyncGE method appears only to the left of Iteration 1 as it is a single-step approach.

9 Conclusion

We propose GEAR, a novel framework that incorporates a graph-based retriever within a multi-step retrieval agent to model the information-seeking process for multi-hop question answering.

We showcase the synergy between our proposed graph retriever (i.e. SyncGE) and the LLM within the GEAR framework. SyncGE leverages the LLM to synchronise information from passages with triples and expands the graph by exploring diverse beams of triples that link multi-hop contexts. Our experiments reveal that this strategy improves over more naive implementations, demonstrating the LLM's capability to guide the exploration of initial nodes for graph expansion. Furthermore, GEAR utilises multi-hop contexts returned by SyncGE and constructs a gist memory which is used for effectively summarising information across iterations. GEAR achieves superior performance compared to other multi-step retrieval methods while requiring fewer iterations and LLM tokens.

Limitations

The scope of this paper is limited to retrieval with the aid of a graph of triples that bridge corresponding passages. While we demonstrated the efficacy of our graph expansion approach and GEAR, we acknowledge that the implementation of the underlying graph follows previous study (Gutierrez et al., 2024) and is rather simple. Better graph construction that addresses challenges such as entity disambiguation (Dredze et al., 2010) and knowledge graph completion (Lin et al., 2015) can lead to further improvements. Further discussion is provided in Appendix D.

We focused on employing a dense embedding model for our diverse triple beam search scoring function, though alternative functions could open up promising avenues for future research. For example, one can study the feasibility of formulating the scoring of neighbours as a natural language inference task (Wang et al., 2021), using a model that predicts how confidently a sequence of triples answers the given query.

Additionally, our approach relies on LLMs that can be better prompted to achieve superior performance on the relevant GEAR tasks. Nonetheless, we provide more experiments in Appendix F showcasing that GEAR can achieve equivalent performance with open-weight LLMs.

References

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. 2023. Walking down the memory maze: Beyond context limit through interactive reading. *Preprint*, arXiv:2310.05029.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, page 758–759, New York, NY, USA. Association for Computing Machinery.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. LongRoPE: Extending LLM context window beyond 2 million tokens. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11091–11104. PMLR.

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 277–285, Beijing, China. Coling 2010 Organizing Committee.

Marina Drosou and Evaggelia Pitoura. 2010. Search result diversification. *SIGMOD Rec.*, 39(1):41–47.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization.

Jinyuan Fang, Zaiqiao Meng, and Craig MacDonald. 2024. TRACE the evidence: Constructing knowledge-grounded reasoning chains for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8472–8494, Miami, Florida, USA. Association for Computational Linguistics.

Benjamin J. Griffiths, George Parish, Frederic Roux, Sebastian Michelmann, Mircea van der Plas, Luca D. Kolibius, Ramesh Chelvarajah, David T. Rollings, Vijay Sawlani, Hajo Hamer, Stephanie Gollwitzer, Gernot Kreiselmeyer, Bernhard Staresina, Maria Wimber, and Simon Hanslmayr. 2019. Directional coupling of slow and fast hippocampal gamma with neocortical alpha/beta oscillations in human episodic memory. *Proceedings of the National Academy of Sciences*, 116(43):21834–21842.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Simon Hanslmayr, Bernhard P. Staresina, and Howard Bowman. 2016. Oscillations and episodic memory: Addressing the synchronization/desynchronization conundrum. *Trends in Neurosciences*, 39(1):16–25.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Daphne Ippolito, Reno Kriz, João Sedoc, Maria Kustikova, and Chris Callison-Burch. 2019. Comparison of diverse decoding methods from conditional language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3752–3762, Florence, Italy. Association for Computational Linguistics.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

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. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, Wenbo Su, and Bo Zheng. 2024. GraphReader: Building graph-based agent to enhance long-context abilities of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12758–12786, Miami, Florida, USA. Association for Computational Linguistics.

Lei Liang, Mengshu Sun, Zhengke Gui, Zhongshu Zhu, Zhouyu Jiang, Ling Zhong, Yuan Qu, Peilong Zhao, Zhongpu Bo, Jin Yang, Huaidong Xiong, Lin Yuan, Jun Xu, Zaoyang Wang, Zhiqiang Zhang, Wen Zhang, Huajun Chen, Wenguang Chen, and Jun Zhou. 2024. KAG: Boosting llms in professional domains via knowledge augmented generation. *Preprint*, arXiv:2409.13731.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1).

OpenAI. 2023. GPT-4 technical report. *Preprint*, arXiv:2303.08774.

Jinyoung Park, Ameen Patel, Omar Zia Khan, Hyunwoo J. Kim, and Joo-Kyung Kim. 2023. Graph-guided reasoning for multi-hop question answering in large language models. *CoRR*, abs/2311.09762.

Jinyoung Park, Ameen Patel, Omar Zia Khan, Hyunwoo J. Kim, and Joo-Kyung Kim. 2024. Graph elicitation for guiding multi-step reasoning in large language models. *Preprint*, arXiv:2311.09762.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. 2024. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

Tao Shen, Guodong Long, Xiubo Geng, Chongyang Tao, Yibin Lei, Tianyi Zhou, Michael Blumenstein, and Daxin Jiang. 2024. Retrieval-augmented retrieval: Large language models are strong zero-shot retriever. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15933–15946, Bangkok, Thailand. Association for Computational Linguistics.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. DRAGIN: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12991–13013, Bangkok, Thailand. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search: Decoding diverse solutions from neural sequence models. *Preprint*, arXiv:1610.02424.

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*, pages 9414–9423, Singapore. Association for Computational Linguistics.

Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. 2021. Entailment as few-shot learner. *Preprint*, arXiv:2104.14690.

Yu Wang, Nedim Lipka, Ryan A. Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024. Knowledge graph prompting for multi-document question answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19206–19214.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

A Dataset Choices and Statistics

Table 5 serves as a summary of various facts and statistics related to the employed datasets and the chunking and triple extraction process introduced

	MuSiQue	2Wiki	HotpotQA
Split Source	IRCoT	IRCoT	HippoRAG
# Hops	$2-4$	2	2
# Documents	$139,416$	$430,225$	$9,221$
# Test Queries	500	500	$1,000$
# Chunks ($\mathbf{C}$)	$148,793$	$490,454$	$10,293$
# Triples ($\mathbf{T}$)	$1,521,136$	$4,993,637$	$122,492$
Av. # $\mathbf{T}/\mathbf{C}$	10.2	10.2	11.9

Table 5: Dataset characteristics and preprocessing statistics, where triples are extracted from chunks, and Av. # $\mathbf{T}/\mathbf{C}$ represents the average number of triples per chunk.

in Section 3. Please note for all the evaluated datasets, we use their open-domain setting and answerable subset if applicable.

Reasoning behind dataset split choices For MuSiQue and 2Wiki, we use the data provided by Trivedi et al., including the full corpus and subsampled test cases for each dataset. To limit the experimental cost for HotpotQA, we follow the setting by Gutierrez et al. where both the corpus and test split are smaller than IRCoT's counterpart.

B More Implementation Details

B.1 Baselines Details

We implement all proposed approaches using Elasticsearch[1]. For SBERT, we employ the `all-mpnet-base-v2` model with approximate k-nearest neighbours and cosine similarity for vector comparisons. In IRCoT experiments, we evaluate both ColBERTv2 and BM25 retrievers — ColBERTv2 for alignment with HippoRAG's baselines, and BM25 for consistency with the original IRCoT implementation.

For all multi-step approaches, including ours, we follow Gutierrez et al. with respect to the maximum number of retrieval iterations, which vary based on the hop requirements of each dataset. Thus, we use a maximum of 4 iterations for MuSiQue and 2 iterations for HotpotQA and 2Wiki.

B.2 GEAR Details

GEAR involves several hyperparameters, such as the beam size inside graph expansion. We randomly sampled 500 questions from the MuSiQue development set, which we ensure not to overlap with the relevant test set. We select our hyperparameters based on this sample without performing a grid search across all possible configurations. Our goal is to demonstrate that our method is able to achieve state-of-the-art results without extensive parameter tuning.

The initial retrieval phase utilises the chunks index $\mathbf{C}$ as the information source, while leaving the triple index $\mathbf{T}$ unused. Our graph expansion component implements beam search with length 2, width 10, and 100 neighbours per beam. The hyperparameter γ employed in diverse triple beam search is set to twice the beam search width. For the scoring function, we use the cosine similarity score and the SBERT embedding model. In Appendix E, we test the performance of GEAR across different beam search length values and maximum numbers of agent iterations.

For the single-step configurations (i.e. any base retriever with NaiveGE or SyncGE), we set the base retriever's maximum number of returned chunks to match our evaluation recall threshold. With the multi-step setup, we maintain a consistent maximum of 10 retrieved chunks before knowledge synchronisation for the purpose of matching IRCoT's implementation. While this 10-chunk limitation applies to individual retrieval rounds, please note that the total number of accessible chunks can exceed this threshold through graph expansion and multiple GEAR iterations.

`passageLink` Details We use `passageLink` to link each triple $t_j \in \mathcal{G}^{(n)}$ to its corresponding passages in $\mathbf{C}$ by running a retrieval step as follows:

$$h^k_{\text{base}}(\mathbf{q}, \mathbf{C} \cup \mathbf{T}) = \text{RRF}\Big(h^k_{\text{base}}(t_j, \mathbf{C}), \\ h^k_{\text{base}}(t_j, \mathbf{T})\Big), \qquad (10)$$

where $j \in \left\{1, \ldots, |\mathcal{G}^{(n)}|\right\}$ and $h^k_{\text{base}}(t_j, \mathbf{C} \cup \mathbf{T})$ is the RRF of passages returned by both $\mathbf{T}$ and $\mathbf{C}$ when queried with t_j.

C Ensuring Fair Comparisons

Although related studies often use common datasets, their experimental settings are frequently inconsistent. For instance, Gutierrez et al. subsampled documents from the full corpus, using a different set from the one in the original IRCoT paper. In our paper, we reproduced HippoRAG on MuSiQue and 2Wiki, using the same dataset settings (i.e. full corpus and identical evaluation split) as in IRCoT. On HotpotQA, we follow the same setting as HippoRAG to limit the experimental cost.

[1] https://www.elastic.co

To ensure fairness in our comparisons, we ran all baselines using a consistent experimental setup. Additionally, for areas of potential discrepancy, and where possible, we report the retrieval performance of baselines in their original configuration to confirm that the used experimental setup does not adversely affect performance. Below, we address any potential discrepancies in the following areas: (i) triple extraction methodology, (ii) retrieval metrics, and (iii) LLMs.

Choice of triple extraction methodology HippoRAG employs a sequential approach to triple extraction: it first identifies named entities from a text chunk, and then uses these entities to guide triple extraction in a second step. In contrast, our method extracts both entities and triples simultaneously. Table 6 shows that both approaches achieve comparable retrieval performance across all datasets, with each method excelling in different scenarios. These results validate that joint entity and triple extraction can match the effectiveness of sequential extraction while reducing the number of required processing steps.

Reasoning behind retrieval metrics Our evaluation employs recall at ranks 5, 10, and 15 (R@5, R@10, R@15). While previous works, such as HippoRAG, evaluate R@2, we choose higher rank thresholds since many questions in MuSiQue require information from more than two documents. Additionally, given modern LLMs' expanding context length capabilities (Ding et al., 2024), examining recall beyond R@5 (HippoRAG's highest evaluated rank) provides valuable insights. Following IRCoT's approach, we measure up to R@15 and include R@10 as an intermediate point, offering a comprehensive view of model performance across retrieval depths.

Choice of LLM Gutierrez et al. use `gpt-3.5-turbo-1106` for their experiments, whereas in this paper we reproduce it with GPT-4o mini. GPT-4o mini was selected as a more capable alternative to GPT-3.5 Turbo (please refer to: `https://openai.com/index/gpt-4-mini-advancing-cost-efficient-intelligence`). In order to alleviate any concerns regarding discrepancies with respect to the selected LLM, we also run experiments using `gpt-3.5-turbo-1106`. Table 7 shows the retrieval results of our proposed methods against HippoRAG w/ IRCoT. We observe a similar trend to that in Table 2—GEAR surpasses the performance of HippoRAG w/ IRCoT.

D Why this graph construction method?

We adopt an LLM-based triple extraction methodology, following the approach outlined in HippoRAG (Gutierrez et al., 2024). In their study, they evaluated the performance of various LLMs in OpenIE and compared these results with those of the end-to-end REBEL model (Huguet Cabot and Navigli, 2021). They reported substantial improvements in triple extraction when using LLMs in domains that deviate from conventional ClosedIE or OpenIE settings, which are respectively overly constrained and unconstrained in terms of named entities and pre-defined relations. Similar concerns about the generalisability and scalability of conventional KG construction approaches in open-domain scenarios are recognised by Wang et al., who sought to construct their graphs without relying on pre-existing ontologies, or KGs for named entity disambiguation.

These findings resonate with the growing interest in recent literature towards applying such methodologies for automatic, schema-free knowledge graph construction (Li et al., 2024; Fang et al., 2024; Gutierrez et al., 2024; Park et al., 2024). As our primary focus is retrieval rather than graph construction, we adopt the triple extraction methodology from HippoRAG and refer readers to their paper for a more detailed analysis.

Our work presents a novel framework for advancing the performance of RAG systems in the context of texts associated with schema-free triples. We follow HippoRAG's graph construction approach, as exploring graph construction methods falls outside the scope of this paper. However, this does not imply that our proposed method relies on this specific approach, and we believe that further improvements in graph construction could lead to additional gains.

E GEAR across Different Configurations

Table 8 illustrates the performance of GEAR across varying hyperparameter configurations, including beam search length—applied during graph expansion—and the maximum number of agent iterations. As the maximum number of iterations increases, GEAR achieves better retrieval performance. However, consistently with the trends shown in Figure 2, this improvement levels off when setting the max-

		MuSiQue			2Wiki			HotpotQA		
		R@5	R@10	R@15	R@5	R@10	R@15	R@5	R@10	R@15
HippoRAG	original prompt	**41.9**	46.9	51.1	**75.4**	**83.5**	**86.9**	79.7	88.4	91.4
	our prompt	41.0	**47.0**	**51.4**	75.1	83.2	86.4	**79.8**	**89.0**	**92.4**
HippoRAG w/ IRCoT	original prompt	**49.9**	**56.4**	**59.3**	81.5	90.2	92.3	**90.2**	**94.7**	95.8
	our prompt	48.8	54.5	58.9	**82.9**	**90.6**	**93.0**	90.1	**94.7**	**95.9**

Table 6: Retrieval performance comparison between HippoRAG's sequential triple extraction method and our joint extraction approach across three datasets.

LLM	Retriever	MuSiQue			2Wiki			HotpotQA		
		R@5	R@10	R@15	R@5	R@10	R@15	R@5	R@10	R@15
GPT-3.5 turbo	Hybrid + SyncGE	48.3	55.0	58.1	70.7	78.4	79.7	86.1	92.0	94.3
	HippoRAG w/ IRCoT	45.5	51.0	54.8	80.0	87.9	89.8	87.4	92.6	94.6
	GEAR	**52.9**	**62.1**	**64.4**	**84.2**	**90.0**	**90.3**	**91.0**	**95.6**	**96.1**

Table 7: Retrieval performance for our proposed retrievers and HippoRAG w/ IRCoT across various datasets. We use `gpt-3.5-turbo-1106` with a temperature of 0 as the underlying LLM, to replicate HippoRAG's experimental setup.

imum number of iterations at $n \geq 3$. In contrast, increasing beam search length above 2 slightly reduces performance. Despite this, GEAR maintains highly competitive results and significantly outperforms alternative methods shown in Table 2.

F Compatibility with Open-weight Models

GEAR Results As shown in Table 9, we evaluate GEAR using popular 7-8B parameter open-weight models, comparing them against a closed-source alternative. On HotpotQA, Llama-3.1-7B surpasses the closed-source alternative, achieving higher recall rates at R@10 and R@15. For MuSiQue and 2Wiki, while the closed-source model maintains a slight superior edge in performance, the margin is narrow. Importantly, all tested open-weight models consistently outperform the previous state-of-the-art, HippoRAG w/IRCoT. This decouples GEAR from the need to use closed-source models, suggesting that state-of-the-art multi-step retrieval can be achieved using more accessible models.

Diverse Beam Search Results Expanding upon Table 4, Table 10 demonstrates that diverse beam search consistently improves retrieval performance across both closed-source and open-weight models when using our proposed Hybrid + SyncGE setup. This further confirms the broader applicability of this approach.

G Robustness Studies

We assess the robustness of our framework in retrieving passages when triple extraction produces either limited or excessive triple content. Using the MuSiQue dataset, we group questions based on the average number of triples (i.e. triple density, ρ_t) associated with their golden passages and evaluate R@15 performance across these ranges. Table 11 presents the results for both the single- and multi-step retrieval settings.

The results showcase that SyncGE and GEAR are more robust than the competition at retrieving suitable passages. NaiveGE's performance tends to decline when the average number of triples associated with the gold passages either falls below or exceeds a certain threshold (for MuSiQue, the average number of triples extracted from the gold passages is 11.71). A similar trend is observed for HippoRAG w/ IRCoT in the case of golden passages associated with more than 11 triples. We believe that this trend can be partially attributed to the Personalised PageRank machinery that makes HippoRAG agnostic to the semantic relationships of the extracted triples. In contrast, SyncGE and GEAR are able to maintain consistent performance across both dense and sparse triple extraction outcomes.

		R@k across Different Maximum Numbers of Iterations											
		n = 1			n = 2			n = 3			n = 4		
		R@5	R@10	R@15	R@5	R@10	R@15	R@5	R@10	R@15	R@5	R@10	R@15
Beam Search Length	b = 2	58.1	66.0	69.5	**59.2**	**68.9**	71.3	57.9	68.0	**71.5**	58.4	67.6	**71.5**
	b = 3	55.9	64.6	67.9	57.2	66.6	70.2	58.1	67.8	71.0	56.7	66.1	70.4
	b = 4	54.9	62.9	67.3	56.6	66.3	69.3	58.1	67.9	71.0	56.1	66.1	69.9

Table 8: GEAR's retrieval performance across different hyper-parameters in terms of maximum number of agent iterations (n) and graph expansion's beam search length (b). Results are reported using Recall@k (R@k) metrics for $k \in \{5, 10, 15\}$ for the MuSiQue dataset.

	LLM	MuSiQue			2Wiki			HotpotQA		
		R@5	R@10	R@15	R@5	R@10	R@15	R@5	R@10	R@15
Closed-source	GPT-4o mini	**58.4**	**67.6**	**71.5**	**89.1**	**95.3**	**95.9**	**93.4**	96.8	97.3
Open-weight	Llama-3.1-8B	52.4	62.3	66.7	81.6	91.0	93.7	92.2	**97.4**	**98.1**
	Qwen-2.5-8B	53.7	63.7	66.7	85.9	91.6	93.0	91.7	96.2	96.9

Table 9: Retrieval performance of GEAR across different closed-source and open-weight models on MuSiQue, 2Wiki and HotpotQA. Results are reported using Recall@k (R@k) metrics for $k \in \{5, 10, 15\}$, showing the percentage of questions for which the correct entries are found within the top-k retrieved passages. The included open-weight models are Llama-3.1-8B-Instruct and Qwen-2.5-7B-Instruct, and the closed-source model is GPT-4o mini.

		MuSiQue			2Wiki			HotpotQA		
		R@5	R@10	R@15	R@5	R@10	R@15	R@5	R@10	R@15
GPT-4o mini	w/ diversity	**48.7**	**57.7**	**61.2**	**72.6**	**80.9**	**82.4**	**87.4**	**93.3**	**95.2**
	w/o diversity	47.0	53.9	58.4	68.2	76.0	77.4	85.0	92.2	94.3
Llama-3.1-8B-Instruct	w/ diversity	**46.2**	**54.3**	**57.4**	**69.1**	**78.1**	**81.6**	**87.3**	**92.8**	**95.1**
	w/o diversity	44.9	52.7	55.0	66.9	75.9	78.2	85.0	91.7	94.4

Table 10: Retrieval performance of the Hybrid + SyncGE method with different LLMs for the `read` step (see Eq. 1) w/ and w/o diversity for triple beam search. Results are reported using Recall@k (R@k) metrics for $k \in \{5, 10, 15\}$, showing the percentage of questions for which the correct entries are found within the top-k retrieved passages.

	Retriever	$\rho_t < 9$	$9 \leq \rho_t < 11$	$11 \leq \rho_t < 13$	$13 \leq \rho_t$
Single-step Retrieval	Hybrid + NaiveGE	50.6	54.6	54.1	50.0
	Hybrid + SyncGE	**62.8** ($\uparrow$ 12.2%)	**61.2** ($\uparrow$ 6.6%)	**59.8** ($\uparrow$ 5.7%)	**60.1** ($\uparrow$ 10.1%)
Multi-step Retrieval	HippoRAG w/ IRCoT	64.4	65.5	55.0	52.8
	GEAR	**73.6** ($\uparrow$ 9.2%)	**73.7** ($\uparrow$ 8.2%)	**69.3** ($\uparrow$ 14.3%)	**69.7** ($\uparrow$ 16.9%)

Table 11: Retrieval performance for single- and multi-step retrievers across different triple density measurements in MuSiQue. Results are reported using R@15. Triple densities (ρ_t) are calculated as the average number of triples associated with the gold documents for the questions within the MuSiQue's test set.

Query	GEAR	HippoRAG w/ IRCoT
What team is the player who led the league in sacks in 2017 a member of?	🟩 ("Chandler Jones", "team", "Arizona Cardinals") ("Chandler Jones", "led the league in sacks in", "2017")	🟥 To answer the question about the player who led the league in sacks in 2017, I need to identify the player and their team from external sources, as the provided documents do not contain that specific information
Who is the CEO of FleetBoston Financial's purchaser?	🟩 ("FleetBoston Financial", "purchaser", "Bank of America") ("Bank of America", "CEO", "Brian Moynihan")	🟨 The purchaser of FleetBoston Financial is Bank of America. To find the CEO of Bank of America, I need to look for information about the current leadership of the bank. Next, I will check for the current CEO of Bank of America to answer the question accurately.
What UK label was purchased by the company that, along with the distributor of The Cube, and ABC, is the other major New York-based broadcaster?	🟩 ("CBS", "purchased", "Oriole Records") ("CBS", "is a major broadcaster based in", "New York") ("NBC", "is a major broadcaster based in", "New York") ("The Cube", "was distributed by", "NBC")	🟨 The original network of Undercovers is NBC, which is one of the major broadcasters based in New York. Now, I need to find out which UK label was bought by NBC ...
What was the 2018 population of the Italian city that's underwater?	🟩 ("Venice", "population in 2018", "260,897")	🟥 The Italian city that is underwater is Krag, British Columbia, which is a ghost town...

Table 12: Comparison of MuSiQue queries where GEAR achieves 100% recall at R@15 in a single iteration, while HippoRAG w/ IRCoT shows lower performance despite using all four available iterations. Cell colors indicate recall performance: 🟩 green for 100% recall, 🟥 red for 0% recall, and 🟨 yellow for any intermediate value. Cell values in GEAR represent the proximal triples stored in the Gist Triple Memory. Cell values in HippoRAG w/ IRCoT represent IRCoT's thought process.

H Qualitative Analysis

H.1 Positive Instances in MuSiQue

Table 12 showcases some query instances where GEAR achieves perfect recall in a single iteration, while HippoRAG w/ IRCoT achieves lower recall and consumes all available iterations. The presented examples illustrate how GEAR's Gist Memory $\mathcal{G}^{(n)}$ precisely captures the essential information needed to answer MuSiQue's queries, maintaining the appropriate level of granularity without including superfluous details. In contrast, HippoRAG w/ IRCoT struggles to retrieve crucial information—whether due to limitations in its triple extraction step or retriever functionality—such as the exact population of Venice, which is necessary for accurate responses. Furthermore, the verbose nature of IRCoT's thought process component contrasts with GEAR's streamlined approach. The lack of such verbose component in our approach contributes to the fact that GEAR requires fewer LLM tokens than the competition, as explained in subsection 8.3.

H.2 Beyond MuSiQuE, 2Wiki and HotpotQA

We evaluate our framework on three open-domain multi-hop QA datasets: **MuSiQue** (Trivedi et al., 2022), **HotpotQA** (Yang et al., 2018), and **2Wiki-MultiHopQA** (2Wiki) (Ho et al., 2020). Our dataset choices closely align with the multi-hop QA tasks, and are consistent with related studies in this space (Li et al., 2024; Fang et al., 2024; Gutierrez et al., 2024; Park et al., 2024).

In order to explore the generalisability of GEAR in additional scenarios, we use the *hand-picked* case study data[2] provided by Gutierrez et al.. These include four path-finding questions across four different domains: books, movies, universities and biomedicine. We test GEAR against HippoRAG w/ IRCoT on these cases. Table 13 displays the results. In three out of the four cases, GEAR outperforms the competition in recall, successfully identifying more relevant passages, and misses the relevant passages in only one case.

I Increasing the Number of Agent Iterations

Figure 5 expands upon the analysis shown in Figure 2 by evaluating retrieval performance over 20 iterations, rather than the initial 4 iterations. The results demonstrate a consistent pattern across all methods: retrieval performance stabilises after approximately 4 iterations, with no substantial improvements or degradation in subsequent iterations. While some minor fluctuations occur beyond this point, they are negligible.

This performance plateau can be attributed to two key factors. First, the query re-writing mechanisms in all investigated approaches struggle to generate effective subsequent queries. Second, our analysis has identified several cases of unanswerable queries within MuSiQue's answerable subset. A representative example is provided in Table 14.

[2] `https://github.com/OSU-NLP-Group/HippoRAG/tree/main/data`

Query	GEAR				HippoRAG w/ IRCoT			
	Passage Titles	R@5	R@10	R@15	**Passage Titles**	R@5	R@10	R@15
Which book was published in 2012 by an English author who is a Whitbread Award winner?	$\mathcal{P}_1$ A Stitch in Time $\mathcal{P}_2$ Stevie Parle $\mathcal{P}_3$ **The Red House** $\mathcal{P}_4$ Whitbread Awards ... $\mathcal{P}_{10}$ **Mark Haddon**	50	100	100	$\mathcal{P}_1$ Oranges Are Not ... $\mathcal{P}_2$ **Mark Haddon** $\mathcal{P}_3$ William Trevor ... $\mathcal{P}_4$ The Curious ... ... $\mathcal{P}_{11}$ **The Red House**	50	50	100
Which war film based on a non fiction book was directed by someone famous in the science fiction and crime genres?	$\mathcal{P}_1$ And the Band ... $\mathcal{P}_2$ Band of Brothers $\mathcal{P}_3$ Aircraft in Fiction $\mathcal{P}_4$ Unchained ...	0	0	0	$\mathcal{P}_1$ Shangai Patrol $\mathcal{P}_2$ **Black Hawk Down** $\mathcal{P}_3$ **Ridley Scott** $\mathcal{P}_4$ Outline of science ...	100	100	100
Which Stanford professor works on the neuroscience of Alzheimer's?	$\mathcal{P}_1$ Thomas C. Sudhof $\mathcal{P}_2$ **Thomas C. Sudhof** $\mathcal{P}_3$ Judes Poirier $\mathcal{P}_4$ **Thomas C. Sudhof** ... $\mathcal{P}_{10}$ **Robert Malenka** $\mathcal{P}_{13}$ **Robert Malenka**	50	75	100	$\mathcal{P}_1$ **Thomas C. Sudhof** $\mathcal{P}_2$ **Thomas C. Sudhof** $\mathcal{P}_3$ Manolis Kellis $\mathcal{P}_4$ Giovanna Malluci $\mathcal{P}_5$ Dena Dubal	50	50	50
What drug is used to treat chronic lymphocytic leukemia by interacting with cytosolic p53?	$\mathcal{P}_1$ P53 Regulation $\mathcal{P}_2$ Venetoclax $\mathcal{P}_3$ **Chlorambucil** $\mathcal{P}_4$ Chronic Lymphocytic Leukemia	50	50	50	$\mathcal{P}_1$ P53 $\mathcal{P}_2$ Cirmtuzumab $\mathcal{P}_3$ MDC1 Function $\mathcal{P}_4$ Chronic Lymphocytic Leukemia	0	0	0

Table 13: Retrieval performance comparison between GEAR and HippoRAG w/ IRCoT. Both models are configured with a maximum number of 4 iterations. The example questions are taken from Gutierrez et al. and showcase multi-hop path-finding queries across different domains: books, movies, universities and biomedicine. Cell colours indicate recall performance: green for 100% recall, red for 0% recall, and yellow for any intermediate value. Retrieved passage titles are listed in the 'Passage Titles' columns, with **bold** text indicating gold passages and $\mathcal{P}_n$ indicating their position in the retrieved list.

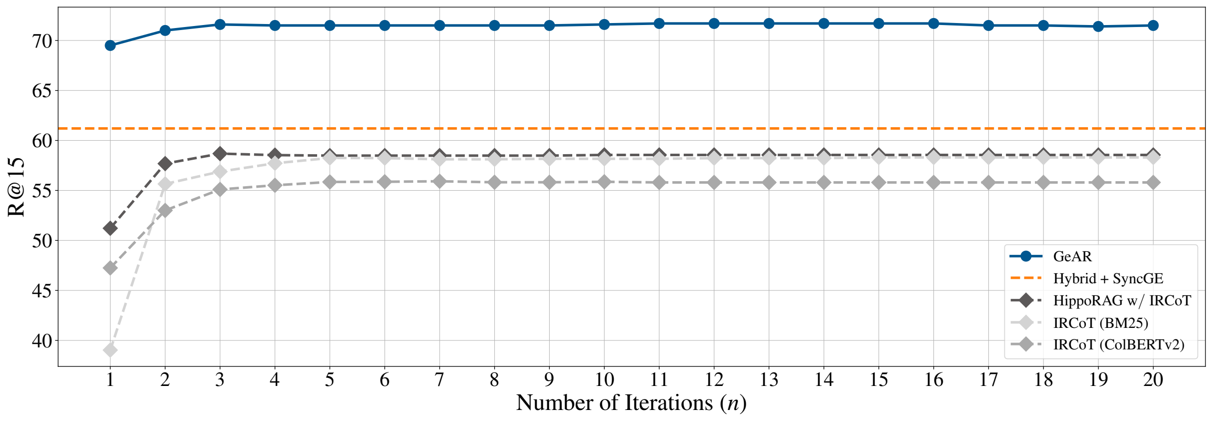


Figure 5: Evolution of R@15 over 20 iterations on MuSiQue. Recall is computed at each iteration using the cumulative set of retrieved documents, with prior recall values carried forward for questions that terminated in earlier iterations. The horizontal line indicates the single-step performance of Hybrid + SyncGE.

Question	Who did the producer of Big Jim McLain play in True Grit?
Gold Passages	1. Big Jim McLain: Big Jim McLain is a 1952 political thriller film starring John Wayne and James Arness as HUAC investigators.
	2. True Grit is a 1969 American western film. It is the first film adaptation of Charles Portis' 1968 novel of the same name. The screenplay was written by Marguerite Roberts. The film was directed by Henry Hathaway and starred Kim Darby as Mattie Ross and John Wayne as U.S. Marshal Rooster Cogburn. Wayne won his only Academy Award for his performance in this film and reprised his role for the 1975 sequel Rooster Cogburn.
Comment	No information about who was the producer of Big Jim McLain is provided in the gold passages

Table 14: Example of a query from MuSiQue that is not answerable solely based on the provided gold passages.

J Prompts

J.1 Offline Prompts

Reader

~~~~
# Instruction

Your task is to construct an RDF (Resource Description Framework) graph from the given passages and named entity lists.
Respond with a JSON list of triples, with each triple representing a relationship in the RDF graph.
Pay attention to the following requirements:
- Each triple should contain at least one, but preferably two, of the named entities in the list for each passage.
- Clearly resolve pronouns to their specific names to maintain clarity.

Convert the paragraph into a JSON dict containing a named entity list and a triple list.

# Demonstration #1

Paragraph:
‘‘‘
Magic Johnson

After winning a national championship with Michigan State in 1979, Johnson was selected first overall in the 1979 NBA draft by the Lakers, leading the team to five NBA championships during their "Showtime" era.
‘‘‘
{{"named_entities": ["Michigan State", "national championship", "1979", "Magic Johnson",
"National Basketball Association", "Los Angeles Lakers", "NBA Championship"]}}
{{
"triples": [
("Magic Johnson", "member of sports team", "Michigan State"),
("Michigan State", "award", "national championship"),
("Michigan State", "award date", "1979"),
("Magic Johnson", "draft pick number", "1"),
("Magic Johnson", "drafted in", "1979"),
("Magic Johnson", "drafted by", "Los Angeles Lakers"),
("Magic Johnson", "member of sports team", "Los Angeles Lakers"),
("Magic Johnson", "league", "National Basketball Association"),
("Los Angeles Lakers", "league", "National Basketball Association"),
("Los Angeles Lakers", "award received", "NBA Championship"),
]
}}
‘‘‘

# Demonstration #2

Paragraph:
‘‘‘
Elden Ring

Elden Ring is a 2022 action role-playing game developed by FromSoftware. It was directed by Hidetaka Miyazaki with worldbuilding provided by American fantasy writer George R. R. Martin.
‘‘‘
{{"named_entities": ["Elden Ring", "2022", "Role-playing video game", "FromSoftware", "Hidetaka Miyazaki", "United States of America", "fantasy", "George R. R. Martin"]}}
{{
"triples": [
("Elden Ring", "publication", "2022"),
("Elden Ring", "genre", "action role-playing game"),
("Elden Ring", "publisher", "FromSoftware"),
("Elden Ring", "director", "Hidetaka Miyazaki"),
("Elden Ring", "screenwriter", "George R. R. Martin"),
("George R. R. Martin", "country of citizenship", "United States of America"),
("George R. R. Martin", "genre", "fantasy"),
]
}}


# Input
~~~~

Convert the paragraph into a JSON dict, it has a named entity list and a triple list.

Paragraph:
"'
{**wiki_title**}

{**passage**}

J.2 Online Retrieval Prompts

The blue-highlighted portions of the Reader prompt below indicate additional text that is only required when the Gist Memory $\mathcal{G}^{(n)}$ is active. When Gist Memory is inactive, these blue sections should be omitted, and the {triples} parameter should be left empty.

Reader with and without Gist Memory

Your task is to find facts that help answer an input question.

You should present these facts as knowlege triples, which are structured as ("subject", "predicate", "object").
Example:
Question: When was Neville A. Stanton's employer founded?
Facts: ("Neville A. Stanton", "employer", "University of Southampton"), ("University of Southampton", "founded in", "1862")

Now you are given some documents:
{**docs**}

Based on these documents and some preliminary facts provided below,
find additional supporting fact(s) that may help answer the following question.

Note: if the information you are given is insufficient, output only the relevant facts you can find.

Question: {**query**}
Facts: {**triples**}

Reasoning for Termination

# Task Description:
You are given an input question and a set of known facts:
Question: {**query**}
Facts: {**triples**}

Your reply must follow the required format:
1. If the provided facts contain the answer to the question, your should reply as follows:
Answerable: Yes
Answer: ...

2. If not, you should explain why and reply as follows:
Answerable: No
Why: ...

# Your reply:

Query Re-writing

# Task Description:
You will be presented with an input question and a set of known facts.
These facts might be insufficient for answering the question for some reason.
Your task is to analyze the question given the provided facts and determine what else information is needed for the next step.

# Example:

Question: What region of the state where Guy Shepherdson was born, contains SMA Negeri 68?
Facts: ("Guy Shepherdson", "born in", "Jakarta")
Reason: The provided facts only indicate that Guy Shepherdson was born in Jakarta, but they do not provide any information about the region of the state that contains SMA Negeri 68.
Next Question: What region of Jakarta contains SMA Negeri 68?

# Your Task:
Question: {**query**}
Facts: {**triples**}
Reason: {**reason**}

Next Question:

J.3 Online Question Answering Prompts

The following prompt with retrieved passages combines the QA generation prompts from Gutierrez et al. and Wang et al.. For the variation without retrieved passages, we omit the preamble and only include the instruction, highlighted in purple .

Retrieved Passages with In-context Example

As an advanced reading comprehension assistant, your task is to analyze text passages and corresponding questions meticulously, with the aim of providing the correct answer.
==================
For example:
==================
Wikipedia Title: Edward L. Cahn
Edward L. Cahn (February 12, 1899 – August 25, 1963) was an American film director.

Wikipedia Title: Laughter in Hell
Laughter in Hell is a 1933 American Pre-Code drama film directed by Edward L. Cahn and starring Pat O'Brien. The film's title was typical of the sensationalistic titles of many Pre-Code films. Adapted from the 1932 novel of the same name buy Jim Tully, the film was inspired in part by "I Am a Fugitive from a Chain Gang" and was part of a series of films depicting men in chain gangs following the success of that film. O'Brien plays a railroad engineer who kills his wife and her lover in a jealous rage and is sent to prison. The movie received a mixed review in "The New York Times" upon its release. Although long considered lost, the film was recently preserved and was screened at the American Cinematheque in Hollywood, CA in October 2012. The dead man's brother ends up being the warden of the prison and subjects O'Brien's character to significant abuse. O'Brien and several other characters revolt, killing the warden and escaping from the prison. The film drew controversy for its lynching scene where several black men were hanged. Contrary to reports, only blacks were hung in this scene, though the actual executions occurred off-camera (we see instead reaction shots of the guards and other prisoners). The "New Age" (an African American weekly newspaper) film critic praised the scene for being courageous enough to depict the atrocities that were occurring in some southern states.

Wikipedia Title: Theodred II (Bishop of Elmham)
Theodred II was a medieval Bishop of Elmham. The date of Theodred's consecration unknown, but the date of his death was sometime between 995 and 997.

Wikipedia Title: Etan Boritzer
Etan Boritzer(born 1950) is an American writer of children 's literature who is best known for his book" What is God?" first published in 1989. His best selling" What is?" illustrated children's book series on character education and difficult subjects for children is a popular teaching guide for parents, teachers and child- life professionals. Boritzer gained national critical acclaim after" What is God?" was published in 1989 although the book has caused controversy from religious fundamentalists for its universalist views. The other current books in the" What is?" series include What is Love?, What is Death?, What is Beautiful?, What is Funny?, What is Right?, What is Peace?, What is Money?, What is Dreaming?, What is a Friend?, What is True?, What is a Family?, What is a Feeling?" The series is now also translated into 15 languages. Boritzer was first published in 1963 at the age of 13 when he wrote an essay in his English class at Wade Junior High School in the Bronx, New York on the assassination of John F. Kennedy. His essay was included in a special anthology by New York City public school children compiled and published by the New York City Department of Education.

Wikipedia Title: Peter Levin
Peter Levin is an American director of film, television and theatre.

Question: When did the director of film Laughter In Hell die?
Answer: August 25, 1963.
==================
Given the following text passages and questions, please present a concise, definitive answer, devoid of additional elaborations, and of maximum length of 6 words.
==================

Wikipedia Title : {**title**} {**text**} `for each retrieved passage` ...
Question: {**question**}

Answer:

No Retrieved Passages

Given the following question, please present a concise, definitive answer, devoid of additional elaborations, and of maximum length of 6 words.

Question: {**question**}

Answer: