Joint Enhancement of Relational Reasoning for Long-Context LLMs

Zhirui Chen[♣], **Wei Shen**[♠], **Jiashui Huang**[♠], **Ling Shao**[♣*]
[♣]UCAS-Terminus AI Lab, University of Chinese Academy of Sciences, China
[♠]AI Lab, Terminus International, Terminus Group, China
chenzhirui23@mails.ucas.ac.cn, ling.shao@ieee.org

Abstract

Despite significant progress, large language models (LLMs) still struggle with long contexts due to memory limitations and their inability to tackle complex and long-context tasks. Additionally, LLMs often suffer from a lack of transparency and are prone to producing hallucinations. To address these challenges, we propose **JERR**, a novel framework designed to enhance long-context comprehension via graph-based reasoning in LLMs. JERR integrates three key components: synopsis extraction, graph construction, and relational reasoning. First, synopsis is extracted by chunking text strategically, allowing the model to summarize and understand information more efficiently. Second, we build a directed acyclic graph (DAG) to resolve redundancy, ensuring logical consistency and clarity. Finally, we incorporate Monte Carlo Tree Search (MCTS) to help the model navigate complex reasoning paths, ensuring more accurate and interpretable outputs. This framework provides a novel solution that enables LLMs to handle extended contexts and complex reasoning tasks with improved reliability and transparency. Experimental results show that JERR consistently outperforms all baselines on the ROUGE and F_1 metrics, achieving the highest scores on the LLM-Rater evaluation.

1 Introduction

In recent years, large language models (LLMs) have achieved significant achievements in natural language processing (Zhao et al., 2023). However, transformer-based LLMs still face limitations in high-performance reasoning with long-context inputs due to constraints on memory usage and context window size (Liu et al., 2024b; Shi et al., 2023). Another challenge is that, although these models are pre-trained on extensive text corpora to respond coherently and appropriately to user inputs, they still struggle with complex knowledge-reasoning tasks (Hu et al., 2023; Wang et al., 2024; Li and Wu, 2025).

We consider that the capability to handle long-context and graph-related tasks is critically important. Firstly, LLMs often struggle to provide accurate answers when key information is deeply embedded within lengthy contexts(Gandhi et al., 2024) or when specialized knowledge extends beyond the pre-training corpora (Jiang et al., 2024b), particularly with up-to-date information. Secondly, LLMs lack interpretability, transparency, and accountability, which increases the risk of producing hallucinations (Zhou et al., 2024). Thirdly, few LLMs or frameworks operate in a human-like manner (Lee et al., 2024; Li et al., 2024). They do not think and respond in the thoughtful, deliberative way humans do when faced with challenging questions.

To address these three issues, researchers have proposed many methods to solve these problems recently, which can be roughly divided into two categories. The first approach focused on graph-related tasks by integrating LLMs with knowledge graphs (Luo et al., 2024; Jiang et al., 2024a; Sun et al., 2024; Xu et al., 2024). These methods fine-tune baseline models using large datasets of generated triplets and reasoning paths (Liu et al., 2024a). The second approach emphasizes training-free frameworks, which deliver high benchmark performance and broader generalizability, particularly for long-context processing, without compromising inference capabilities (Lee et al., 2024; Li et al., 2024). These frameworks also simplify knowledge updating (Han et al., 2024b; Wang et al., 2023), allowing users to add new information to the knowledge base without retraining the model (Ibrahim et al., 2024).

Given these challenges, we aim to design a generalized framework to enhance LLMs' capacity for processing long contexts and reasoning effectively. Our approach simulates the way humans

[*]Corresponding Author

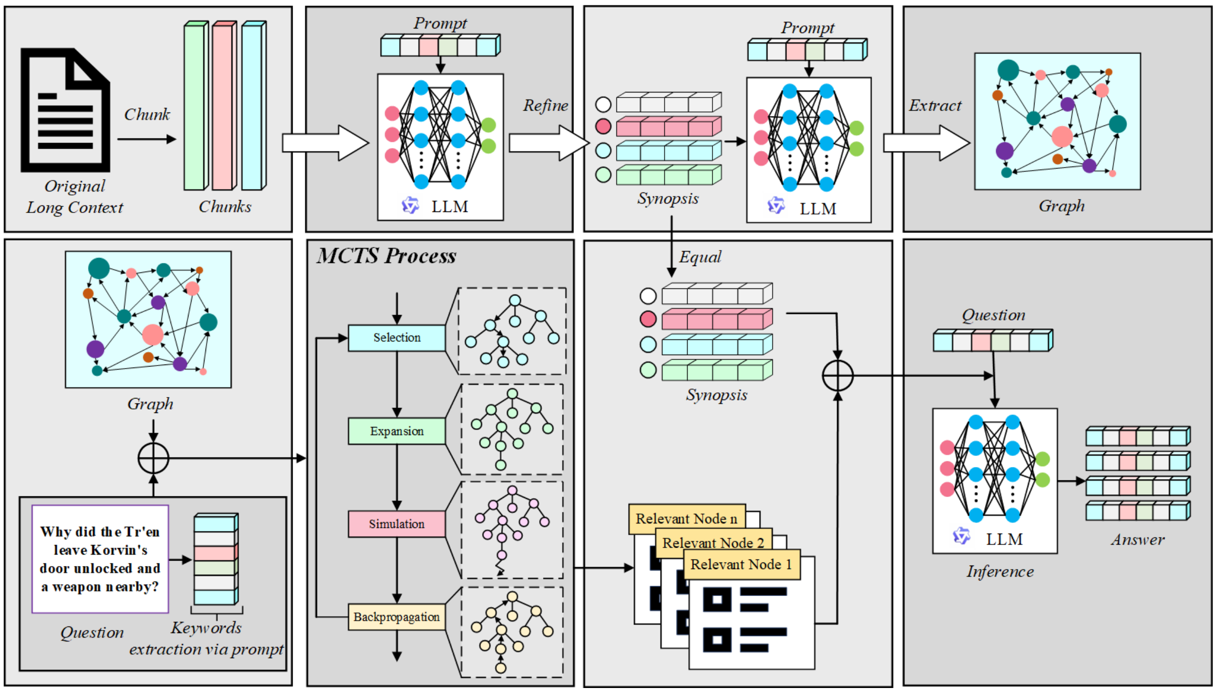


Figure 1: The overall framework of JERR consists of three main steps: 1) Given a question and a long context, LLMs are first prompted to segment the context into chunks. 2) Next, synopsis is generated through LLMs' prompts, followed by the construction of the graph. 3) Finally, reasoning is performed using relevant nodes, identified through the MCTS algorithm on the question and graph, along with the synopsis to generate the final answer.

utilize long-term memory by creating and memorizing summaries of key chunks as entities with interrelated attributes (Zhou et al., 2024). This framework addresses long-context comprehension in three main stages: synopsis extraction, graph construction, and relational reasoning.

For synopsis extraction, we leverage LLMs' summarization prompts to strategically segment the context (Dong et al., 2023; Huang et al., 2024). In graph construction, we employ algorithms that address issues like entity redundancy (Chen et al., 2024a) and attribute recurrence (Ning and Liu, 2024). This phase also involves building a directed acyclic graph (DAG) (Zhang et al., 2024) to allow models to explore complex reasoning paths while maintaining logical coherence. Finally, in the relational reasoning phase, Monte Carlo Tree Search (MCTS) (Zhao et al., 2024; Gao et al., 2024; Browne et al., 2012) enables precise retrieval of relevant nodes on the graph, enhancing the model's capacity to deliver accurate answers to complex queries.

To this end, we propose **JERR** (Figure 1), a **J**oint **E**nhancement of **R**elational **R**easoning for long-context LLMs.

2 Related Work

2.1 Long-Context LLMs

Recent progress in long-context LLMs highlights training strategies that extend context windows to improve model performance (Beltagy et al., 2020; Zhou, 2023; Zaheer et al., 2020; Ainslie et al., 2023; Chen et al., 2024b; Ding et al., 2024). Optimized Transformer attention mechanisms have also been developed to efficiently manage long contexts without extensive fine-tuning (Press et al., 2022; Jin et al., 2024; Chen et al., 2023; Xiao et al., 2024; Han et al., 2024a). However, studies show that model performance often declines with longer inputs, even within the context limit, due to the influence of distracting elements (Liu et al., 2024b; Shi et al., 2023). Our work addresses these issues by improving the effective context length and filtering irrelevant content, avoiding the need for architectural changes or retraining.

2.2 Retrieval

Retrieval-Augmented Generation (RAG) methods allow LLMs to access relevant information from large documents or text segments (Dinan et al., 2019; Izacard and Grave, 2021; Park et al., 2023;

Lewis et al., 2020). Research has examined retrieval granularity at levels such as tokens (Khandelwal et al., 2020), entities (Févry et al., 2020; de Jong et al., 2022), and chunks, with techniques ranging from traditional BM25 (Rasooli and Tetreault, 2015) to advanced learning-based approaches (Sachan et al., 2023). Although RAG enhances retrieval accuracy, it struggles with complex queries due to limited decision-making. Our approach leverages relational reasoning in a graph-based framework to better retrieve and integrate relevant information for tasks requiring long-context understanding.

2.3 Agent for Retrieval

Interactive agents have been employed to help LLMs navigate and process long texts. PEARL (Chen et al., 2024a) uses iterative prompting to improve understanding, and Self-note (Lanchantin et al., 2024) integrates notes with documents for better reasoning. LongRAG (Jiang et al., 2024c) introduces a "long retriever" and a "long reader", allowing the entire corpus to be processed into larger-sized units, which reduces the number of units needed during retrieval and alleviates the burden on the retriever. GraphRAG (Edge et al., 2024) introduces a graph-based retrieval augmentation framework that enhances large language models by explicitly modeling entity relationships and semantic associations within knowledge sources. GraphReader (Li et al., 2024) addresses these issues by structuring text into graphs and employing agents for autonomous exploration.

3 Methodology

This section details the design and implementation of our framework, JERR, an agent that integrates graph structures with long-context processing capabilities.

3.1 Overview of the Approach

As discussed in Section 1, there remains a significant challenge in balancing long-context processing with strong reasoning capabilities in LLMs. To address this, we propose a three-step approach that combines synopsis extraction, graph construction, and graph-based reasoning.

For synopsis extraction, we use the `autogen`[1] package to segment the original long context into chunks. Each chunk is then refined into a synopsis through targeted prompts specifically designed for summarization.

For graph construction, we extract nodes and attributes via prompts of information atoms, core components and attributes extraction. Then we incorporate both exact and similar deduplication techniques. We utilize a Bloom Filter with Trie for exact match and SimHash for similarity-based deduplication. Subsequently, inspired by the core approach in Shi et al.'s (2023) work, we construct a Directed Acyclic Graph (DAG), which significantly enhances the efficiency of graph traversal, allowing the agent to conduct more focused exploration during the reasoning process.

For graph-based reasoning, we identify the top-k nodes most relevant to a given question via employing the Monte Carlo Tree Search (MCTS) algorithm. The resulting subset of relevant nodes, along with the extracted synopsis, is used to prompt the agent. This allows the agent to identify specific sections of the original text that should be revisited and replaced. Finally, the combination of the extracted synopsis and selected passages is used to generate a well-informed response to the query.

3.2 Synopsis Extraction

To extend the capacity of large language models (LLMs) for long-context processing, a foundational approach involves segmenting the context into manageable chunks through prompt-based or conventional chunking methods.

The `autogen` chunking function allows for various customization parameters, including a maximum token limit per chunk, a specified chunking mode, an option to enforce breaks at empty lines, and an overlap line count to ensure continuity between consecutive chunks. Given an input context T, the function can be expressed as follows:

$$chunk\left(T\right) = \{t_1, t_2, ..., t_{\mathrm{n}}\} \tag{1}$$

In the chunking phase, the input text is divided into a set of segments, denoted as $\{t_1, t_2, ..., t_n\}$. This segmentation is followed by a synopsis extraction prompt (in §A.1) with specific parameter settings, designed to facilitate efficient storage and rapid retrieval of the most relevant information from the extended context. This process provides a structured approach to distilling key information from each chunk. The resulting set of synopsis segments can be represented as:

$$S = \{s_1, s_2, ..., s_{\mathrm{n}}\} = \overset{\mathrm{n}}{\underset{i=1}{U}} p_{\mathrm{syn}}\left(t_{\mathrm{i}}\right) \tag{2}$$

[1] https://github.com/microsoft/autogen

where p_{syn} denotes the synopsis extraction prompt. Each synopsis segment s_{n} thus captures the essential content of its corresponding chunk t_i.

3.3 Graph constructing

To maximize the effectiveness of synopsis, we developed a graph construction method that structures an information database, capturing key entities and relationships within the text. This approach provides a robust pipeline for entity deduplication and the construction of a Directed Acyclic Graph (DAG) that effectively encapsulates semantic relationships across the text corpus.

Initially, entities and core elements are extracted from synopsis using the corresponding prompts (in §A.2 and §A.3). Then we process exact deduplication, where each entity is passed through a Bloom Filter[2] and Trie structure to retain only unique entities. The set of exact deduplicated elements is defined as:

$$C = \{c_{\mathrm{i}} \,| c = ddp_{\mathrm{exact}} \left(s_{\mathrm{j}}\right), \ \forall s_{\mathrm{j}} \in S\} \qquad (3)$$

where c_{i} represents the result of deduplication for each synopsis segment s_{j} and ddp_{exact} denotes the exact deduplication function.

A second deduplication step is performed on the unique entities from the previous stage using the SimHash[3] algorithm. SimHash approximates similarity between entities by generating hash values, which are compared to determine near-duplicates. Entities with a bitwise difference below a set threshold (e.g., three bits) are considered similar and are combined. Similarly, the set of similar deduplicated elements is denoted as:

$$V = \{v_{\mathrm{i}} \,| v = ddp_{\mathrm{sim}} \left(c_{\mathrm{j}}, \theta\right), \ \forall c_{\mathrm{j}} \in C\} \qquad (4)$$

where v_{i} denotes the final set of candidate nodes after two rounds of deduplication, θ is the similarity threshold and ddp_{sim} represents the SimHash-based deduplication function.

The final step involves constructing graph. In this stage, unique entities from deduplication stages are iteratively added as nodes in the graph. Each node is annotated with the corresponding synopsis derived from the segmented context. For each pair of entities within a chunk, a directed edge is proposed based on the existence of relational attributes between them. These relational attributes are determined using an edge attribute generation prompt (in §A.4), and the resulting set of attributes can be expressed as:

$$E = \{(v_{\mathrm{i}}, \ v_{\mathrm{j}}) \,|\, prompt_{\mathrm{ea}} \left(v_{\mathrm{i}}, \ v_{\mathrm{j}}\right) \neq 0, \ \forall v_{\mathrm{i}}, v_{\mathrm{j}} \in V\} \quad (5)$$

Thus, graph G is defined as $G = (V, E)$.

3.4 Graph-based Reasoning

After constructing the graph G, we identify the most relevant nodes using prompt-extracted keywords from query and the Monte Carlo Tree Search (MCTS) as a graph search mechanism. Upon retrieving the relevant nodes, a prompt-based mechanism guides the agent in selecting key chunks from the original context. The agent then uses both the retrieved nodes and the selected context chunks to perform an inference step and generate a response to the user's query.

The Monte Carlo Tree Search (MCTS) mechanism searches for relevant nodes by iteratively simulating paths from a root node set V to potential child nodes within the synopsis graph. During the selection phase, it traverses down the tree, choosing child nodes with the highest win/visit ratio to maximize exploration of promising nodes. In the expansion phase, it adds unexplored neighbor nodes as children if the selected node lacks children. The simulation phase computes scores by evaluating keyword matches between the query and the nodes in the simulated path, approximating relevance based on keyword overlap. Finally, in backpropagation, these scores are updated along the path to the root, guiding the search toward nodes that yield high relevance scores in future iterations. Given a specified number of returned nodes k, graph G and user query q, the resulting set of candidate nodes after the MCTS algorithm can be defined as:

$$R = MCTS \left(G, \ q, \ k\right) \qquad (6)$$

The above MCTS process is detailed as §B.

4 Experiment

In this section, we describe the experimental setup and present our main results with an ablation study, case study of our proposed approach (cost analysis is in §C.2).

4.1 Experiment Setup

Dataset We conduct experiments on three types of long-context QA benchmarks, QuALITY, MuSiQue and NarrativeQA, shown in Table 1. See more details in §C.1.

[2]https://pypi.org/project/pybloom/
[3]https://pypi.org/project/simhash/

Table 1: The statistic of benchmark in our experiments to conduct three types of evaluation on our agent

Task	Dataset	Avg Tokens	Max Tokens	Samples
Multi-choice QA	QuALITY	4.1k	6.0k	230
Multi-hop QA	MuSiQue	15.5k	16.0k	200
Single-hop QA	NarrativeQA	29.7k	63.7k	200

Baselines We evaluate the proposed approach against several established baselines:

Retrieval Augmented Generation: As discussed in Section 2, retrieval-augmented generation (RAG) is a widely used approach for accessing extensive text collections beyond the LLM's context window. In our study, we implement two RAG variants: one utilizing BM25 (2020) and the other leveraging neural retrieval with the Qwen API's text-embedding v3 (2024) to retrieve the chunks most relevant to the user's query. The retrieved chunks are then processed by the qwen-plus 128k (2024) model to generate answers.

Long-context LLM: Given the cost constraints for handling extensive long-context benchmarks, we choose not to use GPT-4, instead selecting the high-performance qwen-plus-128k model to directly process the entire input passage. This choice aligns well with our selected datasets, which fit within the qwen-plus-128k input window capacity.

Agent-based Method: For agent-based retrieval, we use ReadAgent (Lee et al., 2024), a model inspired by human reading processes that performs interactive retrieval and reading, demonstrating effective handling of long-context QA tasks. We replace the PALM-2L model used in the ReadAgent study with the qwen-plus-128k model. GraphReader (Li et al., 2024) was not incorporated as a comparative baseline in the present study due to the absence of public source code, thus lacking its operational integration with qwen-plus-128k. We've clarified the differences among JERR, GraphReader and GraphRAG in §D. While GraphRAG (Edge et al., 2024) and LongRAG (Jiang et al., 2024c), as excellent agent-based methods, can be compared during experiments.

Evaluation Settings To assess the long-context summarization capabilities of our model, we employ several automatic evaluation metrics, including ROUGE (R-1, R-2, and R-L) and F_1 score. While these automatic metrics offer high efficiency, the accuracy may vary depending on the model's response format. To address this, we incorporate LLM Raters within ReadAgent to evaluate answer correctness by prompting a comparison to ground truth responses. The LLM Raters include LR-1 (strict version) and LR-2 (permissive version). The specific prompts of them can be seen in (Lee et al., 2024).

Table 2: Comparison of Different Methods on **QuALITY** Datasets

Method	ACC
BM25 Retrieval with qwen API	
Top-1	58.68%
Top-2	66.97%
Top-3	73.21%
Top-4	74.59%
Top-5	78.00%
Top-6	79.91%
Neural Retrieval with qwen API	
Top-1	66.30%
Top-2	73.92%
Top-3	78.14%
Top-4	79.43%
Top-5	81.30%
Top-6	83.32%
qwen-plus-128k	84.80%
ReadAgent with qwen API	83.80%
LongRAG with qwen API	84.91%
GraphRAG with qwen API	85.02%
JERR with qwen API	**86.39%**

Implementation Details In the RAG baseline, chunk sizes are tailored to the specific characteristics of each dataset: 2000 tokens for NarrativeQA and MuSiQue, and 600 tokens for QuALITY, to account for differences in passage lengths. The top-k parameter in MCTS is set to 5, a choice informed by ablation study results. To identify relevant pages across datasets, we adapt a lookup prompt approach inspired by ReadAgent (Lee et al., 2024), applying dataset-specific configurations to effectively address questions in QuALITY, NarrativeQA, and MuSiQue, as outlined in §A.5 and §A.6.

4.2 Overall Performance Comparison

The comparative results of the four baseline methods and the proposed JERR framework on multiple-choice question answering, multi-hop, and single-hop long-context question-answering benchmarks are presented in Table 2 and Table 3.

RAG Methods BM25 and Neural Retrieval approaches demonstrate varying effectiveness across different benchmark tasks. In the QuALITY benchmark (Table 2), both retrieval methods underperform compared to direct reading approaches, suggesting that chunking and ranking processes may introduce unnecessary complexity. The retrieval methods show consistent accuracy improvements

Table 3: Comparison of Different Methods on **MuSiQue** and **NarrativeQA** Datasets

Dataset	**MuSiQue**						**NarrativeQA**					
Method	**LR-1**	**LR-2**	**R-1**	**R-2**	**R-L**	F_1	**LR-1**	**LR-2**	**R-1**	**R-2**	**R-L**	F_1
BM25 Retrieval with qwen API												
Top-1	0.175	0.275	0.147	0.094	0.139	0.253	0.165	0.300	0.100	0.076	0.104	0.182
Top-2	0.260	0.380	0.184	0.149	0.181	0.291	0.255	0.430	0.134	0.108	0.122	0.194
Top-3	0.325	0.47	0.161	0.116	0.153	0.251	0.345	0.520	0.138	0.112	0.135	0.221
Top-4	0.355	0.505	0.172	0.137	0.168	0.233	0.340	0.575	0.143	0.103	0.131	0.217
Top-5	0.425	0.535	0.159	0.112	0.156	0.267	0.420	0.585	0.150	0.143	0.150	0.238
Top-6	0.41	0.525	0.160	0.109	0.154	0.286	0.445	0.645	0.155	0.129	0.150	0.237
Neural Retrieval with qwen API												
Top-1	0.185	0.295	0.072	0.068	0.070	0.235	0.290	0.410	0.075	0.053	0.069	0.156
Top-2	0.275	0.405	0.079	0.062	0.05	0.269	0.340	0.525	0.077	0.056	0.070	0.160
Top-3	0.335	0.450	0.088	0.068	0.084	0.231	0.390	0.570	0.081	0.051	0.073	0.162
Top-4	0.395	0.535	0.087	0.074	0.083	0.288	0.445	0.635	0.087	0.059	0.079	0.178
Top-5	0.425	0.565	0.094	0.076	0.091	0.295	0.445	0.645	0.089	0.055	0.081	0.166
Top-6	0.405	0.525	0.091	0.074	0.089	0.272	0.475	0.685	0.091	0.061	0.083	0.165
qwen-plus-128k	0.425	0.515	0.095	0.075	0.091	0.328	0.515	0.735	0.104	0.066	0.094	0.251
ReadAgent with qwen API	0.405	0.530	0.206	0.169	0.201	0.448	0.520	0.735	0.230	0.209	0.212	0.247
LongRAG with qwen API	0.415	0.545	0.209	0.174	0.203	0.473	0.520	0.735	0.228	0.204	0.201	0.249
GraphRAG with qwen API	0.410	0.550	0.211	0.185	0.205	0.488	0.525	0.745	0.221	0.205	0.207	0.254
JERR with qwen API	**0.455**	**0.595**	**0.226**	**0.212**	**0.218**	**0.505**	**0.540**	**0.760**	**0.234**	**0.215**	**0.216**	**0.269**

up to Top-5/Top-6 chunks, after which returns diminish.

BM25 retrieval consistently outperforms neural retrieval in lexical-based metrics (ROUGE-1, ROUGE-2, ROUGE-L) across MuSiQue and NarrativeQA benchmarks in Table 3. For instance, in MuSiQue, BM25 achieves an R-1 score of 0.159 at Top-5, significantly higher than neural retrieval's 0.094. This performance gap highlights BM25's strength in capturing lexical overlap.

For LR metrics, both retrieval methods exhibit similar patterns: starting with low Top-1 scores, they improve gradually with additional chunks, typically peaking at Top-5 or Top-6. In MuSiQue, both methods reach comparable maximum scores (LR-1: 0.425, LR-2: 0.55), while in NarrativeQA, neural retrieval achieves peak performance with Top-6 chunks (LR-1: 0.475, LR-2: 0.685). However, increasing retrieval chunks beyond these points shows diminishing returns, indicating a trade-off between comprehensive coverage and precision.

Long-Context LLMs qwen-plus-128k demonstrates distinct capabilities across different benchmark tasks. In the QuALITY benchmark, the model achieves strong accuracy through direct question-answering without context segmentation, effectively avoiding the "lost in the middle" issue (Liu et al., 2024b) when input lengths fall within its context window.

For complex summarization tasks in MuSiQue and NarrativeQA, qwen-plus-128k shows mixed performance patterns. While achieving relatively lower ROUGE scores (R-1: 0.095-0.104, R-2: 0.066-0.075, R-L: 0.091-0.094) compared to retrieval-based approaches, the model demonstrates stronger performance in LR metrics. Specifically, it achieves LR-1 scores of 0.425-0.515 and LR-2 scores of 0.515-0.735 across these benchmarks, indicating robust semantic understanding of long-form content.

The F_1 scores (0.328 in MuSiQue, 0.251 in NarrativeQA) suggest balanced precision-recall performance, though generally lower than BM25 retrieval approaches. However, a key advantage of qwen-plus-128k lies in its ability to process entire documents holistically, achieving comparable recall to chunking-based approaches without the overhead of document segmentation and selection.

Agent-based Methods Our framework, JERR, consistently demonstrates superior performance across diverse benchmarks by leveraging agent-based approaches for reasoning and information retrieval. In the QuALITY benchmark (Table 2), JERR outperforms other baselines in multiple-choice question-answering tasks. By employing the Monte Carlo Tree Search (MCTS) algorithm for graph construction and exploration, JERR achieves robust graph comprehension and precise relational reasoning. This enables it to efficiently retrieve and synthesize relevant information, minimizing information loss. In contrast, methods such as ReadAgent, which rely on compressing context into gist memories, face limitations in accuracy due to significant information loss during compression.

For the MuSiQue benchmark (Table 3), JERR demonstrates a clear advantage in long-range recall tasks, achieving superior LR-1 and LR-2 scores of 0.455 and 0.595, respectively, compared to ReadAgent's 0.405 and 0.53. Additionally, JERR surpasses ReadAgent across automatic evaluation metrics, including ROUGE and F_1 scores, showcasing its effectiveness in handling multi-sentence reasoning tasks.

In the NarrativeQA benchmark (Table 3), JERR achieves the highest ROUGE, F_1, and LR scores, excelling in both detailed lexical matching and maintaining narrative coherence. This highlights its ability to adapt to complex narrative structures, effectively preserving key elements from the source text while ensuring a cohesive understanding of the storyline. Across all datasets, JERR is a reliable and versatile approach for long-text reasoning and retrieval tasks.

The results suggest that JERR's graph-based approach offers an advantage in processing and reasoning over extended narrative information, making it a highly effective solution for retrieval-based tasks on narrative datasets.

4.3 Ablation Study

The Effect of MCTS

A critical step in JERR's processes is the MCTS algorithm setting. Thus, in Table 4, we compare JERR's performance using the MCTS algorithm with that of the PageRank algorithm on the QuALITY benchmark. The results clearly show that JERR achieves better performance with MCTS, boosting accuracy by approximately 5% and demonstrating a notable improvement.

Table 4: w. / w.o. MCTS accuracy Comparison on QuALITY of JERR

Method	Accuracy
JERR (via MCTS algorithm)	86.39%
JERR (via PageRank algorithm)	81.69%

Impact of the Number of Relevant Nodes

Another essential factor in JERR's performance is the selection of the top-k values. We conducted experiments with various top-k values on the QuALITY dataset to assess their impact. As shown in Figure 2, performance improves as top-k values increase, peaking at an optimal value of 5, which we adopt as the default setting. Beyond this point, however, performance declines, likely due to an overload of retrieved information, which may hinder the model's inference capability.

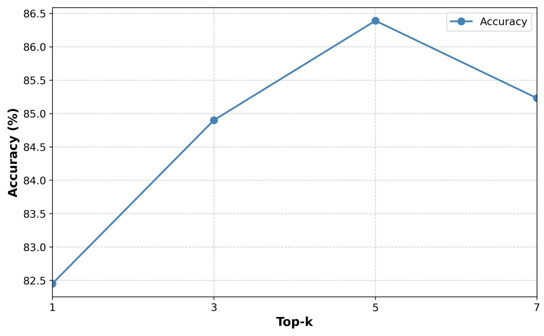


Figure 2: Performance of JERR with different top-k relevant nodes text on QuALITY.

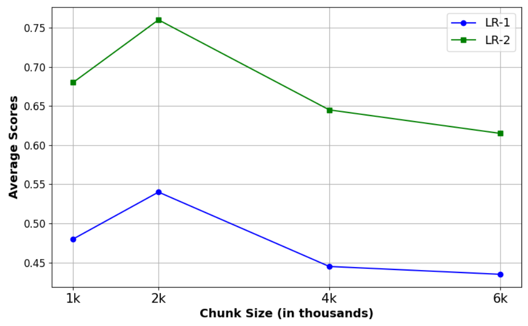


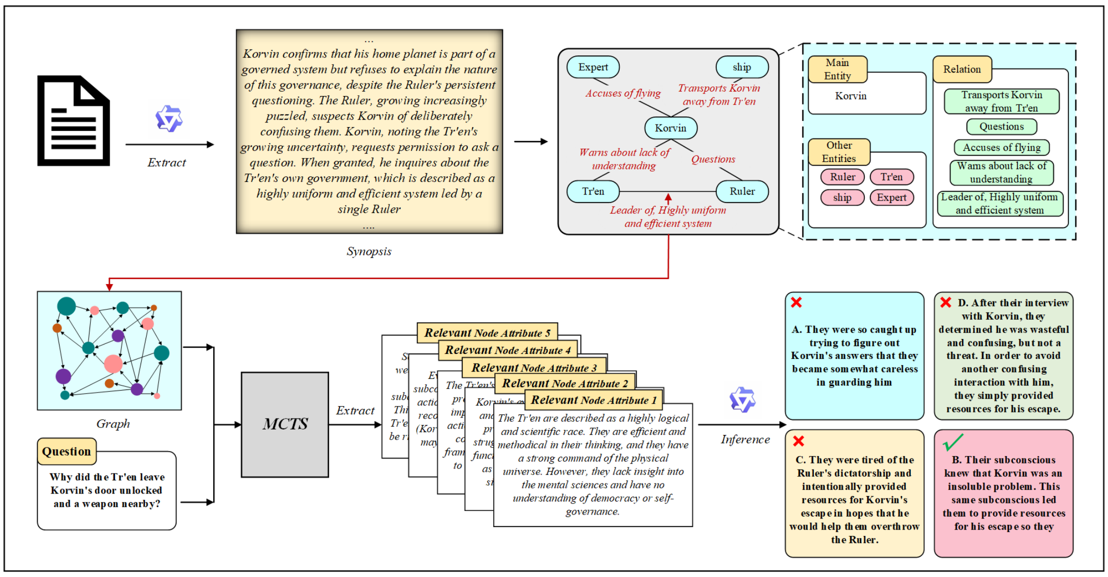
Figure 3: Performance of JERR with different chunk size on NarrativeQA.

Impact of Chunk Size For different setting of chunk size, we conduct experiments comparing LR-1 and LR-2 to evaluate JERR's performance on NarrativeQA dataset. As shown in Figure 3, the best performance is achieved with chunk size of 2k. It is evident that performance declines when the chunk size is set too large, which we attribute to model's inability to capture all details within longer chunks. Smaller chunk sizes allow for more accurate extraction of the synopsis. Therefore, we select chunk size of 2k as default.

In the QuALITY dataset, due to its shorter average context length, we use chunk sizes under 1000 tokens to maintain an optimal number of segments. Consistent with our findings on the NarrativeQA dataset, Figure 5 demonstrates that accuracy decreases when chunk size exceeds an optimal threshold. Peak performance is observed at a chunk size of 600, which we therefore set as the default for QuALITY.

4.4 Case Study

This section demonstrates JERR's workflow through a case study from QuALITY, analyzing a 4,168-word passage about Korvin's interactions


Extract
Synopsis
Expert
ship
Accuses of flying
Transports Korvin away from Tr'en
Korvin
Warns about lack of understanding
Questions
Tr'en
Ruler
Leader of, Highly uniform and efficient system
Main Entity
Korvin
Other Entities
Ruler
Tr'en
ship
Expert
Relation
Transports Korvin away from Tr'en
Questions
Accuses of flying
Warns about lack of understanding
Leader of, Highly uniform and efficient system
Graph
Question
MCTS
Extract
Relevant Node Attribute 5
Relevant Node Attribute 4
Relevant Node Attribute 3
Relevant Node Attribute 2
Relevant Node Attribute 1
Inference

...
Korvin confirms that his home planet is part of a governed system but refuses to explain the nature of this governance, despite the Ruler's persistent questioning. The Ruler, growing increasingly puzzled, suspects Korvin of deliberately confusing them. Korvin, noting the Tr'en's growing uncertainty, requests permission to ask a question. When granted, he inquires about the Tr'en's own government, which is described as a highly uniform and efficient system led by a single Ruler
....

Why did the Tr'en leave Korvin's door unlocked and a weapon nearby?

The Tr'en are described as a highly logical and scientific race. They are efficient and methodical in their thinking, and they have a strong command of the physical universe. However, they lack insight into the mental sciences and have no understanding of democracy or self-governance.

✗ A. They were so caught up trying to figure out Korvin's answers that they became somewhat careless in guarding him

✗ D. After their interview with Korvin, they determined he was wasteful and confusing, but not a threat. In order to avoid another confusing interaction with him, they simply provided resources for his escape.

✗ C. They were tired of the Ruler's dictatorship and intentionally provided resources for Korvin's escape in hopes that he would help them overthrow the Ruler.

✓ B. Their subconscious knew that Korvin was an insoluble problem. This same subconscious led them to provide resources for his escape so they

Figure 4: Study of case from the second passage of QuALITY dev set with the corresponding question (1 out of 9)

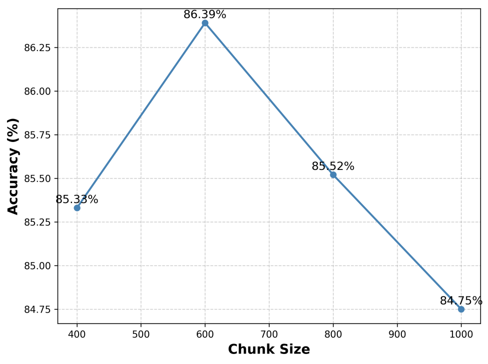


Figure 5: Performance of JERR with different chunk size on QuALITY.

with the Tr'en society, which lacks understanding of democratic principles and mental sciences.

As illustrated in Figure 4, the original passage undergoes refinement into synopsis chunks that preserve essential information. JERR then constructs a knowledge graph where nodes represent key concepts and edges indicate their relationships, enabling systematic mapping of textual information for efficient analysis.

JERR processes the question: *"Why did the Tr'en leave Korvin's door unlocked and a weapon nearby?"* using Monte Carlo Tree Search (MCTS), which simulates various paths within the graph to identify the most pertinent nodes for answering the question.

Based on the MCTS traversal, relevant nodes (highlighted in red boxes) are extracted, encompassing critical concepts including the Tr'en's democratic incomprehension, their inability to consciously resolve problems outside their governance model, and subconscious actions stemming from their mental limitations.

The framework synthesizes these nodes to generate an inference, suggesting that the Tr'en's subconscious influenced their decision to leave Korvin's door unlocked, driven by their desire to eliminate the *"problem"* he represented due to cognitive dissonance within their society.

The generated inference aligns with the correct answer (marked with a green check mark), confirming that the Tr'en subconsciously recognized Korvin as an insoluble problem and facilitated his escape to avoid further societal contradiction.

4.5 Comparison to GraphReader on GPT-4

We've conducted Experiment, which has the same settings in the paper of GraphReader, on MusiQUE and NarrativeQA to compare JERR with GraphReader and other baselines (e.g., ReadAgent, Pearl, LongRAG and GraphRAG) using GPT-4-128k. The table is presented in additional TABLE 5 as below (the statistics in the table is aligned with Table 2 in GraphReader paper, MQ and NR stand for MuSiQue and NarrativeQA, respectively).

The result demonstrates that JERR outperforms GraphReader and other baselines using the base model GPT-4, which shows great robustness and proves that JERR Framework doesn't depend on a specific LLM.

Table 5: Comparison to GraphReader

Methods	**LR-1 (MQ)**	**LR-2 (MQ)**	F_1 **(MQ)**	**LR-1 (NR)**	**LR-2 (NR)**	F_1 **(NR)**
BM25 (Top-1)	33	36.5	23.9	29.5	34.5	11.3
BM25 (Top-3)	43.5	49.5	31.1	44.5	52.5	20.5
Ada-002 (Top-1)	34.5	37	26.6	37.5	46.5	15.5
Ada-002 (Top-3)	40	45.5	32.1	45.5	53	19.5
GPT-4-128k	52	59.5	42.7	63.5	77	29.4
ReadAgent	54.5	61	45.1	63.5	75.5	18.9
Pearl	45	51.5	33.3	43.5	48	16.2
LongRAG	49	54.5	40.3	60.5	69.2	27
GraphRAG	46.5	56	31.2	52	66.5	23.1
GraphReader	59	63.5	47.4	65	80	29.8
JERR	**60.5**	**64**	**49.2**	**68**	**83**	**30.1**

5 Conclusion

We have introduced JERR, a graph with MCTS algorithm based agent, designed to combine effective long context and relational reasoning graph smoothly in LLMs. JERR transfers the input long context into synopsis, builds DAGs and employs MTCS algorithm to search the relational nodes information, thus guaranteeing high performance on long-context tasks versus all the baselines.

6 Limitations

Our current framework JERR has so far been validated exclusively on QuALITY, MuSiQue and NarrativeQA. Experimental results demonstrate that JERR has high performance on such long-context benchmarks. However, JERR has limitations mainly on three aspects:

For task scalability, although JERR performs well on the selected datasets, its ability to generalize to other reasoning tasks in different knowledge domains remains uncertain. Further validation on a broader range of datasets is necessary to assess its scalability.

For knowledge graph construction, the framework relies on the Qwen API to extract graph nodes, which helps automate knowledge representation. However, constructing a refined knowledge graph still requires a complex pipeline which can make large-scale and high-precision graph building a challenging task.

For task complexity, JERR applies a graph-based approach and Monte Carlo Tree Search (MCTS) to improve knowledge reasoning and retrieve relevant text more effectively. However, its effectiveness on simpler tasks has not been systematically evaluated. Future studies should compare its performance on both simple and complex tasks to provide a more complete assessment of its capabilities.

References

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontañón, Siddhartha Brahma, Yury Zemlyanskiy, David C. Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, Yun-Hsuan Sung, and Sumit Sanghai. 2023. Colt5: Faster long-range transformers with conditional computation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 5085–5100. Association for Computational Linguistics.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Okapi BM25. 2020. Title of the webpage. https://github.com/dorianbrown/rank_bm25.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

Shengyuan Chen, Qinggang Zhang, Junnan Dong, Wen Hua, Qing Li, and Xiao Huang. 2024a. Entity alignment with noisy annotations from large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024b. Longlora: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Fei Sha, and William W. Cohen. 2022. Mention memory: incorporating textual knowledge into transformers through entity mention attention. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

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 *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Guanting Dong, Rumei Li, Sirui Wang, Yupeng Zhang, Yunsen Xian, and Weiran Xu. 2023. Bridging the kb-text gap: Leveraging structured knowledge-aware pre-training for kbqa. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 3854–3859.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski. 2020. Entities as experts: Sparse memory access with entity supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4937–4951. Association for Computational Linguistics.

Kanishk Gandhi, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah Goodman. 2024. Understanding social reasoning in language models with language models. *Advances in Neural Information Processing Systems*, 36.

Zitian Gao, Boye Niu, Xuzheng He, Haotian Xu, Hongzhang Liu, Aiwei Liu, Xuming Hu, and Lijie Wen. 2024. Interpretable contrastive monte carlo tree search reasoning. *arXiv preprint arXiv:2410.01707*.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024a. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3991–4008.

Jiuzhou Han, Nigel Collier, Wray L. Buntine, and Ehsan Shareghi. 2024b. Pive: Prompting with iterative verification improving graph-based generative capability of llms. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 6702–6718. Association for Computational Linguistics.

Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. 2023. A survey of knowledge enhanced pre-trained language models. *IEEE Transactions on Knowledge and Data Engineering*.

Wenyu Huang, Guancheng Zhou, Hongru Wang, Pavlos Vougiouklis, Mirella Lapata, and Jeff Z. Pan. 2024. Less is more: Making smaller language models competent subgraph retrievers for multi-hop KGQA. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 15787–15803. Association for Computational Linguistics.

Nourhan Ibrahim, Samar Aboulela, Ahmed Ibrahim, and Rasha Kashef. 2024. A survey on augmenting knowledge graphs (kgs) with large language models (llms): models, evaluation metrics, benchmarks, and challenges. *Discover Artificial Intelligence*, 4(1):76.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 874–880. Association for Computational Linguistics.

Boran Jiang, Yuqi Wang, Yi Luo, Dawei He, Peng Cheng, and Liangcai Gao. 2024a. Reasoning on efficient knowledge paths: Knowledge graph guides large language model for domain question answering. In *IEEE International Conference on Knowledge Graph, ICKG 2023, Shanghai, China, December 1-2, 2023*, pages 142–149. IEEE.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024b. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 1658–1677. Association for Computational Linguistics.

Ziyan Jiang, Xueguang Ma, and Wenhu Chen. 2024c. Longrag: Enhancing retrieval-augmented generation with long-context llms. *arXiv preprint arXiv:2406.15319*.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. 2024. LLM maybe longlm: Selfextend LLM context window without tuning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jack Lanchantin, Shubham Toshniwal, Jason Weston, Sainbayar Sukhbaatar, et al. 2024. Learning to reason and memorize with self-notes. *Advances in Neural Information Processing Systems*, 36.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John F. Canny, and Ian Fischer. 2024. A human-inspired reading agent with gist memory of very long contexts. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Qing Li and Guanzhong Wu. 2025. Explainable reasoning over temporal knowledge graphs by pre-trained language model. *Information Processing & Management*, 62(1):103903.

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, Miami, Florida, USA, November 12-16, 2024*, pages 12758–12786. Association for Computational Linguistics.

Haochen Liu, Song Wang, Chen Chen, and Jundong Li. 2024a. Few-shot knowledge graph relational reasoning via subgraph adaptation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3346–3356.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024b. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yansong Ning and Hao Liu. 2024. Urbankgent: A unified large language model agent framework for urban knowledge graph construction. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Author Name qwen-plus 128k. 2024. Title of the webpage. `https://qwen.readthedocs.io/en/latest/`.

Mohammad Sadegh Rasooli and Joel Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *arXiv preprint arXiv:1503.06733*.

Devendra Singh Sachan, Mike Lewis, Dani Yogatama, Luke Zettlemoyer, Joelle Pineau, and Manzil Zaheer. 2023. Questions are all you need to train a dense passage retriever. *Transactions of the Association for Computational Linguistics*, 11:600–616.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Author Name text-embedding v3. 2024. Title of the webpage. `https://help.aliyun.com/zh/model-studio/developer-reference/text-embedding-quick-start-1`.

Xintao Wang, Qianwen Yang, Yongting Qiu, Jiaqing Liang, Qianyu He, Zhouhong Gu, Yanghua Xiao, and Wei Wang. 2023. Knowledgpt: Enhancing large language models with retrieval and storage access on knowledge bases. *arXiv preprint arXiv:2308.11761*.

Xinyi Wang, Alfonso Amayuelas, Kexun Zhang, Liangming Pan, Wenhu Chen, and William Yang Wang. 2024. Understanding reasoning ability of language models from the perspective of reasoning paths aggregation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Guang Liu, Jun Zhao, and Kang Liu. 2024. Generate-on-graph: Treat LLM as both agent and KG for incomplete knowledge graph question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 18410–18430. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yifan Zhang, Yang Yuan, and Andrew Chi-Chih Yao. 2024. On the diagram of thought. *arXiv preprint arXiv:2409.10038*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2024. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36.

Le Zhou. 2023. Longt5-mulla: Longt5 with multi-level local attention for a longer sequence. *IEEE Access*, 11:138433–138444.

Tong Zhou, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Cogmg: Collaborative augmentation between large language model and knowledge graph. *arXiv preprint arXiv:2406.17231*.

A Prompts

A.1 Synopsis Extraction Prompt

Please transfer the following chunk of a passage into synopsis.
Just give me a synopsis version. No extra explanation.
Passage: {SYNOPSIS TEXT}

A.2 Entities Extraction Prompt

You are a summarizing agent. Given a chunk of paragraph, summarize it into information atoms. (Information Atoms: Brief statements represent the most fundamental, indivisible facts, covering ideas like propositions, theories, entities, concepts, and underlying aspects such as reasoning, cause-effect relationships, event sequences, social interactions, timelines, and similar elements.)

A.3 Elements Extraction Prompt

You are a keyword extracting agent. Given a chunk of paragraph of a story, extract only 3 core components.
(Core components: The fundamental nouns (e.g., people, moments, occurrences, settings, quantities), verbs (e.g., activities), and adjectives (e.g., conditions, emotions) that are central to the story's progression.)

A.4 Edge Attributes Prompt

Based on the atomic facts: '{INFORMATION ATOMS1}' and '{INFORMATION ATOMS2}', what are the attributes between {ELEMENT1} and {ELEMENT2}?
(Information Atoms: smallest, indivisible truths extracted from text chunks.)
Use no more than three words to answer.

A.5 QuALITY Answer Prompt

Read the following article and answer a multiple choice question. For example, if (C) is correct, answer with "Answer: (C) ...". No extra explanation please.
Article: {CONTEXT}
Question: {QUESTION} {OPTIONS}

A.6 NarrativeQA / MuSiQue Answer Prompt

{RELEVANT NODES TEXT & SYNOPSIS}
Question: {QUESTION}
Answer the question based on the above relevant content and extracted synopsis. Your answer should be short and concise.

B MCTS Process

Initialize the Search Tree We define the search tree nodes $\mathcal{T}$, where each tree node corresponds to an informational node within the graph. Each node's state $s_{\mathrm{j}} \in S$ is associated with a graph node $v_{\mathrm{j}} \in V$ in graph G. The visit count $N(s)$ represents the number of times a node's state s_{j} has been visited, while the total reward $W(s)$ reflects the cumulative reward accumulated by visits to node's state s_{j}. The average reward of node's state s_{j}, utilized during the selection stage, is calculated as:

$$Q\left(s\right) = \frac{W\left(s\right)}{N\left(s\right)} \tag{7}$$

Selection The selection strategy employs the Upper Confidence Bound (UCB) to balance exploration and exploitation. An exploration coefficient κ is introduced to regulate this balance. The UCB for a node s is defined as:

$$UCB\left(s\right) = Q\left(s\right) + \kappa \cdot \sqrt{\frac{lnN\left(parent\left(s\right)\right)}{N\left(s\right)}} \tag{8}$$

The selection criterion then chooses the node with the highest $UCB\left(s\right)$ in the current set of tree nodes $\mathcal{T}$:

$$s_{next} = \underset{s \in \mathcal{T}}{argmax}\, UCB\left(s\right) \tag{9}$$

Algorithm 1 MCTS Relevant Nodes Extraction Algorithm

```
Input: Graph $G$, Query text $Q$, Start nodes $S$, Number of simulations $N$
Output: Top nodes based on keyword relevance
 1: Initialize root node $R$ with state $S[0]$
 2: Extract keywords $K$ from $Q$ as set of lowercase words
 3: for $i = 1$ to $N$ do
 4:   Selection: Set current node $n \leftarrow R$
 5:   while $n$ has children do
 6:     Select the best child of $n$ based on win/visit ratio
 7:     if $n$ has no selectable children then
 8:       break
 9:     end if
10:   end while
11:   Expansion:
12:   if $n$ has no children then
13:     Expand children for $n$ using neighbors in $G$
14:   end if
15:   Simulation:
16:   if $n$ has children then
17:     Set current node $n$ to best child of $n$
18:   end if
19:   Initialize score score $\leftarrow 0$
20:   for depth $d = 1$ to 10 do
21:     if $n$ exists in $G$ then
22:       Increment score by matching keywords $K$ with state of $n$
23:       Set $n$ to best child of $n$
24:       if $n$ has no children then
25:         break
26:       end if
27:     end if
28:   end for
29:   Backpropagation:
30:   while $n$ is not null do
31:     Update visits and wins for $n$ based on score
32:     Set $n$ to parent of $n$
33:   end while
34: end for
35: Sort root children by win/visit ratio and return top nodes
36: return Top relevant nodes based on MCTS exploration
```

Expansion If the selected node s_{next} has not reached a terminal state or attained a sufficient visit count, the expansion step continues by adding its corresponding graph node $v_{next} \in V$ to its neighboring nodes. The appropriate set of nodes for expansion is defined as:

$$\mathcal{E}\left(s_{next}\right) = \left\{s' \in \mathcal{N}\left(v_{next}\right) \cap s' \notin \mathcal{T}\right\} \qquad (10)$$

Simulation In this stage, a random simulation is conducted from one of the newly expanded nodes s' to estimate the potential reward along this path. The reward for state s' is evaluated based on its relevance to the query, which is determined by the maximum simulation depth d and a set of keywords extracted from the query. The reward function is given as:

$$r\left(s'\right) = \sum_{i=1}^{d} \left|K \cap \mathcal{E}\left(s'\right)\right| \qquad (11)$$

Backpropagation The final step of MCTS is the Backpropagation phase, where the reward $r\left(s'\right)$ obtained from the simulation is propagated back through each node along the traversal path, updating both the total reward $W(s)$ and visit count $N(s)$ for each node:

$$N(s_n) = N(s_{n-1}) + 1 \qquad (12)$$

$$W(s_n) = W(s_{n-1}) + r\left(s'\right) \qquad (13)$$

As a result, the set of *top* $- k$, ranked by relevance, is used as the set of retrieved nodes R:

$$R = \left\{ \left(r_i, \frac{W_i}{N_i} \right) \middle| i = 1, 2, ..., k \right\} \qquad (14)$$

In summary, the entire MCTS procedure comprises five stages, as illustrated in Algorithm 1. The final retrieval of relevant nodes is expressed as:

$$R = MCTS\left(G,\ q,\ k\right) = \left\{r_1,\ r_2,\ ...,\ r_{\mathrm{k}}\right\},\ r_{\mathrm{i}} \in V \qquad (15)$$

C Experiment

C.1 Dataset list

QuALITY[4] is a four-way multiple-choice QA challenge comprising text data from diverse sources, evaluated based on accuracy. The dev set of QuALITY includes 230 long-context samples with corresponding questions, answer options, and other details.

[4]https://github.com/nyu-mll/quality

MuSiQue[5] serves as a multi-hop long-context QA benchmark, with an average of 15.5 thousand tokens across 200 samples. MuSiQue's questions include various content types, such as narratives, expository texts, and factual material. This diversity challenges models to generalize across various genres and structures, strengthening their capacity for comprehensive language understanding and reasoning.

We also include the **NarrativeQA**[6] dataset as a single-hop long-context benchmark, averaging 29.7 thousand tokens across 200 samples. This dataset focuses on narrative comprehension in a long-form context, requiring models to answer questions based on an understanding of entire narratives, such as books or movie scripts, rather than isolated text segments. This differentiates NarrativeQA from typical QA datasets, which generally focus on shorter text segments.

C.2 Cost Analysis

Table 6 compares the average token consumption across methods used in our experiments. Neural retrieval demonstrates the highest token usage, as it requires converting context chunks into embeddings with the text-embedding-v3 model, adding significant computational overhead. BM25 also has high token consumption, greater than that of the qwen-plus-128k method alone, due to the additional step of evaluating and selecting Top-k chunks for answer generation. Our agent-based method, JERR, consumes 1.23 times more tokens than ReadAgent but achieves superior performance.

Table 6: comparison of token consumption per question among all methods on MuSiQue LR evaluation, where "Avg. Ctx. #Tokens" refers to the average token number of the original dataset. The "Avg. Cost #Tokens" comprise both input tokens and output tokens during exploration

Method	Avg. Ctx # Tokens	Avg. Cost #Tokens
JERR	15.5 k	98.54 k
JERR (w.o. Graph Construction)	15.5 k	44.33 k
ReadAgent	15.5 k	79.98 k
Neural Retrieval	15.5 k	138.90 k
BM25	15.5 k	39.18 k
qwen-plus-128k	15.5 k	16.99 k

Additionally, once JERR has constructed its graphs, it can reuse these structures efficiently without requiring reconstruction for each new query. This capability significantly reduces JERR's token cost, averaging 44.33k tokens per query when operating without additional graph construction. This efficiency marks a substantial improvement over the token consumption required by ReadAgent.

D Differences among JERR, GraphReader and GraphRAG

GraphReader constructs undirected graphs where nodes represent text chunks or atomic facts. While this captures local relationships, undirected edges lack explicit hierarchical or causal dependencies, leading to inefficient exploration (e.g., cycles or backtracking). While JERR introduces a directed acyclic graph (DAG) to model causal and hierarchical relationships explicitly. For example, edges in JERR's DAG are weighted by relational attributes (e.g., "causes," "belongs-to"), allowing the agent to prioritize paths with stronger semantic relevance. This structure inherently avoids cycles and supports efficient traversal, whereas GraphReader's undirected graph requires heuristic rules to prevent redundant exploration.

GraphRAG relies on entity extraction and hierarchical community detection (via Leiden algorithm) to group nodes, followed by community-level summarization. While effective for global query-focused summarization, this approach treats communities as isolated modules, limiting cross-community reasoning. Instead of partitioning nodes into communities, we perform synopsis extraction—an LLM-guided summarization process that preserves logical dependencies across chunks. This generates a condensed yet interconnected set of entities and relationships, avoiding information fragmentation. Unlike GraphRAG's static community summaries, JERR's synopses retain hierarchical relationships (via DAGs), enabling multi-hop reasoning across distant text segments.

[5]https://github.com/stonybrooknlp/musique
[6]https://github.com/google-deepmind/narrativeqa