Graph of Records: Boosting Retrieval Augmented Generation for Long-context Summarization with Graphs

Haozhen Zhang[1*] and **Tao Feng** and **Jiaxuan You**
University of Illinois at Urbana-Champaign
{haozhenz,taofeng2,jiaxuan}@illinois.edu, [1]wazhz14@gmail.com

Abstract

Retrieval-augmented generation (RAG) has revitalized Large Language Models (LLMs) by injecting non-parametric factual knowledge. Compared with long-context LLMs, RAG is considered an effective summarization tool in a more concise and lightweight manner, which can interact with LLMs multiple times using diverse queries to get comprehensive responses. However, the LLM-generated historical responses, which contain potentially insightful information, are largely neglected and discarded by existing approaches, leading to suboptimal results. In this paper, we propose *graph of records* (**GoR**), which leverages historical responses generated by LLMs to enhance RAG for long-context global summarization. Inspired by the *retrieve-then-generate* paradigm of RAG, we construct a graph by establishing an edge between the retrieved text chunks and the corresponding LLM-generated response. To further uncover the intricate correlations between them, GoR features a *graph neural network* and an elaborately designed *BERTScore*-based objective for self-supervised model training, enabling seamless supervision signal backpropagation between reference summaries and node embeddings. We comprehensively compare GoR with 12 baselines across four long-context summarization datasets, and the results indicate that our proposed method reaches the best performance (*e.g.*, 15%, 8%, and 19% improvement over retrievers w.r.t. Rouge-L, Rouge-1, and Rouge-2 on the WCEP dataset). Extensive experiments further demonstrate the effectiveness of GoR. Code is available at https://github.com/ulab-uiuc/GoR

1 Introduction

Large Language Models (LLMs) have recently achieved remarkable performance across sorts of language modeling tasks (Achiam et al., 2023; AI@Meta, 2024). Among them, the long-context global summarization task is of great importance, which requires ultra-long context understanding capabilities of LLMs (Li et al., 2024a; Liu et al., 2024b). Current attempts to accomplish this task mainly include long-context LLMs (Touvron et al., 2023; GLM et al., 2024; Li* et al., 2023; Tworkowski et al., 2023) and retrieval-augmented generation (RAG) (Ram et al., 2023; Yu et al., 2023; Trivedi et al., 2022; Jiang et al., 2023b; Asai et al., 2023). In comparison with long-context LLMs that expand their context window to accommodate long-context inputs, RAG performs a cost-effective *retrieve-then-generate* paradigm and provides a few retrieved short text chunks from a long document to LLMs. In a running RAG system, there are usually a large number of historical user queries and LLM-generated responses for a long document. Nevertheless, these historical responses, which contain informative task-related content, are mostly neglected without sufficient utilization by current RAG approaches.

Unfortunately, utilizing LLM historical responses for long-context global summarization presents two major challenges. (1) *Sophisticated yet implicit correlations between historical responses and text*. Given a long document, there will inevitably be complicated correlations among plentiful user queries (*e.g.*, logical correlations), which are further inherited by LLM-generated responses and the retrieved text chunks. However, uncovering these correlations is non-trivial since most text embeddings from language models (*e.g.*, SBERT (Reimers and Gurevych, 2019)) or retrievers (Karpukhin et al., 2020) concentrate on semantic similarity, which faces degrading performance in this case. (2) *Lack of supervision signal*. In contrast with local (*e.g.*, query-based) summarization (Zhong et al., 2021; Wang et al., 2022a) that includes golden reference text as labels, global summarization needs to be considered from the perspec-

[*]Work done as an intern at University of Illinois at Urbana-Champaign

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 23780–23799

tive of the long document as a whole and only has global reference summaries, which complicates the direct backpropagation of effective, accurate, and deterministic supervision signals to optimize the model towards a few relevant text chunks.

Based on the above observations, we propose *graph of records* (**GoR**), which utilizes and organizes LLM historical responses as a graph of records for enhancing long-context global summarization in RAG. In detail, we first leverage LLMs to simulate some user queries conditioned on arbitrary text chunks in a long document to obtain historical responses under the paradigm of RAG, and an edge is then created between the retrieved text chunks and the LLM-generated response to construct a graph of records. To learn fine-grained correlations among nodes, we employ a *graph neural network* and reuse the simulated user queries with the corresponding source text chunk as self-supervised training data. Intuitively, we hope the node embeddings can be adaptively learned to reflect the semantic and logical correlations with a given query. Inspired by the well-received BERTScore (Zhang et al., 2019) that quantifies the semantic similarity between two paragraphs of text, we rely on it to rank the nodes according to their similarity with the self-supervised label of a given simulated query. In this way, node embeddings can benefit the indirect supervision signal from the self-supervised labels and be flexibly optimized using a contrastive loss and a pair-wise ranking loss based on the node rankings. In the experiments, we adopt four long-context summarization datasets, and the results demonstrate the superiority and effectiveness of our proposed method. For example, we show that GoR outperforms retrievers by 15%, 8%, and 19% w.r.t. Rouge-L, Rouge-1, and Rouge-2, respectively, on the WCEP dataset. We also provide detailed comparisons and insightful analyses through extensive experiments, further showcasing the effectiveness of our approach. Our contributions are summarized as follows:

- We propose *graph of records* (**GoR**), which utilizes and organizes LLM-generated historical responses as a graph of records to strengthen RAG for long-context global summarization. We reveal that the fine-grained correlations between LLM historical responses and text chunks from long documents can be uncovered and utilized effectively to improve RAG performance.
- We leverage a *graph neural network* and design a *BERTScore*-based objective to optimize node embeddings, which can be adaptively learned in a self-supervised manner to reflect the semantic and complex correlations with input queries. Furthermore, the indirect supervision signal from self-supervised labels is crucial and conducive to the effective optimization of node embeddings.
- We evaluate our proposed method on four long-context summarization datasets, and the results show that GoR outperforms several competitive baselines by a significant margin. Extensive experiments and detailed analysis verify the superiority of GoR.

2 Graph of Records

In this section, we first present some necessary backgrounds in Section 2.1. Then, we describe our proposed method sequentially through three sections, *i.e.*, Graph Construction (Section 2.2), BERTScore-based Objective for Self-supervised Training (Section 2.3), and Retrieval from the Graph for Summarization (Section 2.4).

2.1 Preliminaries

Retrieval-augmented Generation. Retrieval-augmented Generation (RAG) (Ram et al., 2023) can typically be summarized into the following two processes. (1) *Retrieval*. Give a long document which consists of several split text chunks $\mathbf{C} = \{\mathbf{c_i}\}_{i=1}^{|\mathbf{C}|}$ as retrieval corpus, RAG first employs a retriever (*e.g.*, Contriever (Izacard et al., 2021)) to retrieve $\mathbf{K}$ text chunks that are most relevant to a given query $\mathbf{q}$ based on semantic similarity. The retriever typically embeds the query $\mathbf{q}$ and a text chunk $\mathbf{c}$ from $\mathbf{C}$ using a query encoder $\mathbf{E_q}(\cdot)$ and a context encoder $\mathbf{E_c}(\cdot)$, respectively, and quantify their semantic similarity by the dot product operation described as $\mathrm{Sim}(\mathbf{q}, \mathbf{c}) = \mathbf{E_q}(\mathbf{q})^T \cdot \mathbf{E_c}(\mathbf{c})$. (2) *Generation*. The retrieved text chunks are fed into LLMs with the query $\mathbf{q}$ to obtain the final response $\mathbf{r}$. The whole process can be described as:

$$\begin{aligned} \mathbf{r} &= \mathrm{Generation}(\mathbf{q}, \{\mathbf{c_1}, \cdots, \mathbf{c_K}\}), \\ \{\mathbf{c_1}, \cdots, \mathbf{c_K}\} &= \mathrm{Retrieval}(\mathbf{q}|\mathbf{C}). \end{aligned} \tag{1}$$

Graph Neural Networks. Graph Neural Networks (GNNs) (Kipf and Welling, 2016) stand out for their excellent representation learning ability

on graph data. GNNs update node embeddings iteratively by aggregating messages from their neighboring nodes. Generally, the l-th layer of GNNs can be formalized as:

$$\mathbf{h}_v^{(l)} = \mathrm{AGG}^{(l)}\Big(\mathbf{h}_v^{(l-1)}, \mathrm{MSG}^{(l)}\Big(\{\mathbf{h}_u^{(l-1)}, u \in N(v)\}; \theta_m^l\Big); \theta_a^l\Big). \quad (2)$$

where $\mathbf{h}_u^{(l)} \in \mathbb{R}^{d_l}$ is the embedding vector of nodes u in layer l and the dimension is d_l. $\mathrm{MSG}^{(l)}(\cdot)$ is a message computation function parameterized by θ_m^l and $\mathrm{AGG}^{(l)}(\cdot)$ is a message aggregation function parameterized by θ_a^l in layer l.

2.2 Graph Construction

In this section, we describe how to organize LLM historical responses into a graph of records by simulating user queries.

Query Simulation. User queries play a very critical role in the design of GoR since LLM historical responses generated by lots of repetitive, nonsense, or meaningless questions are inherently not beneficial for summarization. One solution is to use doc2query (Nogueira et al., 2019) to simulate queries for a long document, but the generated results inevitably suffer from simplicity and rigidity due to the limited text generation capabilities of T5 (Raffel et al., 2020). To this end, we directly turn to LLMs for query simulation with temperature sampling instead of greedy decoding for generating meaningful, insightful, and diverse questions. Specifically, we split a long document into several text chunks $\mathbf{C}$ following the standard procedure of RAG and prompt LLMs to generate a query $\mathbf{q^s}$ based on a randomly selected text chunk $\mathbf{c^s}$. We repeat the above process until a certain number of non-duplicate queries are generated, which are gathered in pairs with the corresponding text chunks to form a corpus $\mathbf{T} = \{(\mathbf{q}_i^\mathbf{s}, \mathbf{c}_i^\mathbf{s})\}_{i=1}^{|\mathbf{T}|}$ for further model training (Section 2.3).

Organize LLM Historical Responses into A Graph. After obtaining simulated queries, we utilize them to perform RAG on the long document. LLM-generated responses during this process include informative and valuable understanding, summarizing, and answering of retrieved text chunks in the long document. Moreover, since there may exist sophisticated correlations among simulated queries, the text chunks and responses can inherit these features and potentially assist in answering a more comprehensive query, especially global summarization that needs to be understood from a holistic perspective. Nevertheless, it is a significant challenge to find correlations among complex and massive text at the linguistic level and the embeddings from language models (*e.g.*, SBERT (Reimers and Gurevych, 2019)) or retrievers (Karpukhin et al., 2020) focus on semantic similarity, which also suffers from poor performance in this case. To this end, we propose to break out of this dilemma by organizing these historical responses into a graph.

Inspired by the *retrieve-then-generate* process of RAG, we can connect the retrieved chunks to the corresponding response generated by LLMs since they are naturally relevant in content. Sequentially, during the $\mathbf{i}$-th round RAG, given the simulated query $\mathbf{q_i^s}$, we expand the retrieval corpus $\mathbf{C}$ with previously generated responses $\{\mathbf{r_1}, \cdots, \mathbf{r_{i-1}}\}$ and then build an edge between each retrieved chunk $\mathbf{c_j} \in \{\mathbf{c_1}, \cdots, \mathbf{c_K}\}$ and the newly generated LLM response $\mathbf{r_i}$, resulting in $\mathbf{K}$ edges constructed in each round. Note that we append the responses generated by each round of RAG to the retrieval corpus because they contain more refined knowledge compared with the text chunks from $\mathbf{C}$ and can help LLMs generate comprehensive responses in a self-evolving manner. Formally, the $\mathbf{i}$-th round RAG on simulated queries $\{\mathbf{q}_i^\mathbf{s}\}_{i=1}^{|\mathbf{T}|}$ can be described as:

$$\begin{aligned} \mathbf{r_i} &= \mathrm{Generation}(\mathbf{q_i^s}, \{\mathbf{c_1}, \cdots, \mathbf{c_K}\}), \\ \{\mathbf{c_1}, \cdots, \mathbf{c_K}\} &= \mathrm{Retrieval}(\mathbf{q_i^s}|\mathbf{C}, \{\mathbf{r_1}, \cdots, \mathbf{r_{i-1}}\}). \end{aligned} \quad (3)$$

In this way, the LLM-generated responses serve as *bridges* to connect the originally scattered text chunks $\mathbf{C}$ so that the fine-grained and sophisticated correlations among them can be better modeled and explored. Furthermore, we can potentially leverage historical responses generated by LLMs and enhance the quality of future LLM responses.

2.3 BERTScore-based Objective for Self-supervised Training

So far, we have constructed a graph using LLM-generated historical responses during RAG given the simulated queries. The key in this section lies in designing a reasonable and effective objective function for model optimization. Considering that some random walk (Grover and Leskovec, 2016) or propagation-based (Zhu and Ghahramani, 2002) algorithms are not differentiable, we turn to graph neural networks (GNNs) for learning node embed-

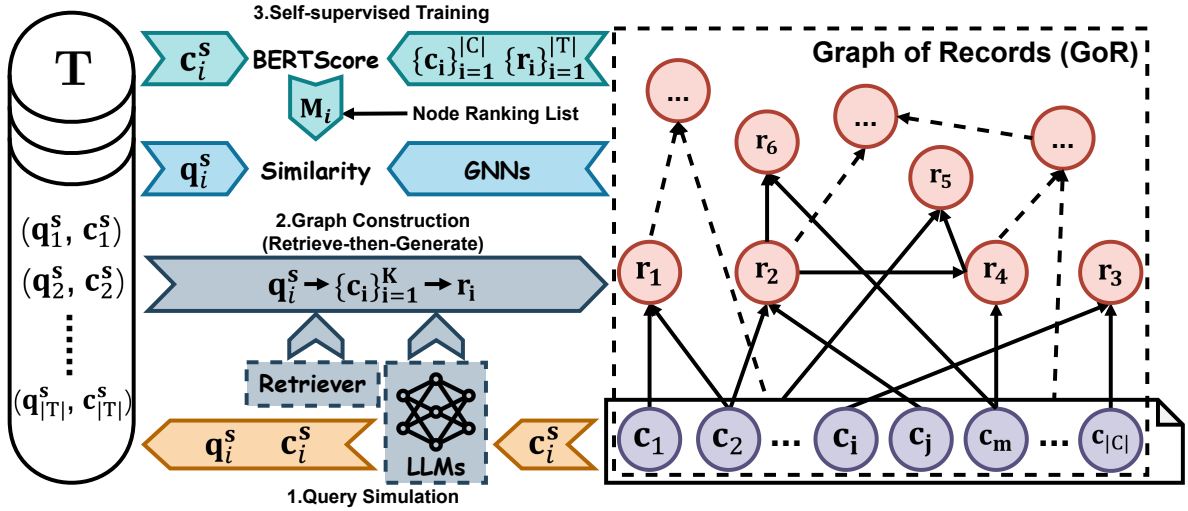

Figure 1: **GoR model architecture**. GoR randomly selects text chunks $\mathbf{c_i}$ from long documents to feed into LLMs for query simulation, which are saved as a self-supervised training corpus $\mathbf{T}$ and further used for graph construction inspired by the retrieve-then-generate paradigm in RAG. For model training, GoR leverages GNNs to obtain node embeddings and calculate their similarities to the query embedding. Finally, GoR features contrastive learning and pair-wise ranking objectives based on the node ranking list $\mathbf{M}_i$ derived from BERTScore calculation.

dings, which are backpropagation-friendly. Intuitively, given a global summarization query $\mathbf{q}$, our ultimate optimization goal is to make the learned node embeddings adaptively reflect the similarity with the query embedding $\mathbf{E_q}(\mathbf{q})$ by taking the complicated correlations among nodes into account. However, in global summarization tasks, there are essentially no text chunk indices as labels to indicate which nodes are most relevant for a query since it needs to consider the long document as a whole. Another naive solution is to use global reference summaries as labels, but there is a gap in supervision signal backpropagation between them and node embeddings because we still need to find out which nodes are most relevant to them.

Therefore, inspired by BERTScore (Zhang et al., 2019), which measures the semantic similarity between the reference and the generated text, we propose to use it to rank all nodes based on the similarity with reference summaries. By this means, BERTScore fills the gap in the backpropagation so that node embeddings can benefit the indirect supervision signal from the reference summaries. Nevertheless, global reference summaries contain broad information about long documents, making them highly semantically relevant to many nodes, which will confuse the model optimization direction and degrade the performance (we will discuss it in Section 3.5).

Contrastive Loss Driven by BERTScore. Based on the above observations, we directly reuse the simulated queries $\mathbf{T} = \{(\mathbf{q}_i^{\mathbf{s}}, \mathbf{c}_i^{\mathbf{s}})\}_{i=1}^{|\mathbf{T}|}$ to serve as self-supervised training data, in which the text chunk $\mathbf{c}_i^{\mathbf{s}}$ is highly relevant to the query $\mathbf{q}_i^{\mathbf{s}}$ and has more focused content. Given node embeddings output by the last L-th layer of GNNs, for the i-th query $\mathbf{q}_i^{\mathbf{s}}$, we rank them according to the similarity with the i-th text chunk $\mathbf{c}_i^{\mathbf{s}}$ and obtain a node embedding ranking list $\mathbf{M}_i$:

$$\mathbf{M}_i = [\mathbf{h}_+^{(L)}, \mathbf{h}_1^{(L)}, \cdots, \mathbf{h}_{|\mathbf{C}|+|\mathbf{T}|}^{(L)}], \tag{4}$$

where $\mathbf{h}_+^{(L)}$ stands for the node embedding with highest similarity. Note that we utilize the context encoder $\mathbf{E_c}(\cdot)$ from the retriever to initialize node embeddings for simplicity. Then, we regard $\mathbf{h}_+^{(L)}$ as the positive while the rest in $\mathbf{M}_i$ as negative samples to conduct contrastive learning using InfoNCE (van den Oord et al., 2018), which brings the query $\mathbf{q}_i^{\mathbf{s}}$ and the positive sample $\mathbf{h}_+^{(L)}$ closer in the semantic embedding space. We formulate the contrastive training objective as follows:

$$s(\mathbf{q}, \mathbf{h}) = \exp\left(\mathbf{E_q}(\mathbf{q})^\top \mathbf{h}/\tau\right), \tag{5}$$

$$\mathcal{L}_{\mathrm{CL}} = -\frac{1}{|\mathbf{T}|}\sum_{j=1}^{|\mathbf{T}|}\log\frac{s(\mathbf{q}_j^{\mathbf{s}}, \mathbf{h}_+^{(L)})}{s(\mathbf{q}_j^{\mathbf{s}}, \mathbf{h}_+^{(L)}) + \sum_{i=1}^{|\mathbf{M}_j-1|} s(\mathbf{q}_j^{\mathbf{s}}, \mathbf{h}_i^{(L)})}, \tag{6}$$

where τ is the temperature coefficient. Note that in the optimization pipeline of GoR, we conduct mini-batch training on the graph level, and each graph is associated with an independent self-supervised training dataset $\mathbf{T}$. We also leverage in-batch negatives from other graphs since the nodes in them are completely irrelevant content from other long documents (it is not shown in Formula 6 for brevity).

Auxiliary Pair-wise Ranking Loss. In the above-described contrastive loss $\mathcal{L}_{\text{CL}}$, although we impose constraints on positive and negative samples, the ranking of negative samples themselves is not well utilized. Inspired by LambdaRank (Burges, 2010), we further introduce an auxiliary pair-wise ranking loss on the ranking list $\mathbf{M}_i$, which can be formulated as:

$$\mathcal{L}_{\text{RANK}} = \frac{1}{|\mathbf{T}|}\sum_{k=1}^{|\mathbf{T}|}\sum_{\mathbf{h}_i^{(L)},\mathbf{h}_j^{(L)}\in\mathbf{M}_k} \mathbb{I}_{\text{r}(\mathbf{h}_j^{(L)})>\text{r}(\mathbf{h}_i^{(L)})}\log\left(1+\frac{s(\mathbf{q}_k^{\mathbf{s}},\mathbf{h}_j^{(L)})}{s(\mathbf{q}_k^{\mathbf{s}},\mathbf{h}_i^{(L)})}\right), \tag{7}$$

where $\text{r}(\cdot)$ denotes the ranking index (*e.g.*, $\text{r}(\mathbf{h}_+^{(L)}) < \text{r}(\mathbf{h}_1^{(L)})$). Given $\mathbf{h}_i^{(L)}, \mathbf{h}_j^{(L)} \in \mathbf{M}_k$ that satisfies $\text{r}(\mathbf{h}_j^{(L)}) > \text{r}(\mathbf{h}_i^{(L)})$, the pair-wise ranking loss will explicitly optimize in the direction of $\mathbf{E}_{\mathbf{q}}(\mathbf{q}_k^{\mathbf{s}})^\top\mathbf{h}_j^{(L)} < \mathbf{E}_{\mathbf{q}}(\mathbf{q}_k^{\mathbf{s}})^\top\mathbf{h}_i^{(L)}$, thus imposing stricter constraints to the pair-wise ranking.

Overall Training Objective. To sum up, the overall training objective can be formulated as:

$$\mathcal{L} = \mathcal{L}_{\text{CL}} + \alpha\cdot\mathcal{L}_{\text{RANK}}. \tag{8}$$

where $\alpha \in [0, 1]$ is a hyper-parameter. It is worth noting that GoR's training costs are lightweight since the only trainable module is GNNs, and no human-crafted labels are needed.

2.4 Retrieval from the Graph for Summarization

During the graph construction phase, we have already obtained a graph consisting of nodes that represent both the text chunks $\mathbf{c}$ from the long document and the responses $\mathbf{r}$ generated by LLMs during the RAG process. These nodes collectively form the retrieval corpus used by GoR during inference. After GNN training, each node is associated with a learned embedding vector that captures not only its semantic content but also its contextual and structural relationships with other nodes in the graph. These learned embeddings replace the original text embeddings produced by conventional retrievers (*e.g.*, Contriever (Izacard et al., 2021)), enabling the model to incorporate richer relational information among document chunks, which is especially beneficial for the summarization task. During inference for global summarization, the process begins with encoding the query into an embedding using the same retriever. We then compute the similarity (via inner product) between the query embedding and all node embeddings in the graph. The top-$\mathbf{K}$ most relevant nodes, comprising both document chunks and LLM-generated responses, are retrieved based on this similarity score. These selected nodes, along with the query, are then fed into the LLM to generate the final summary.

Overall, the retrieval process in GoR mirrors that of standard dense retrievers, with the key distinction being the use of graph-enhanced node embeddings and the inclusion of generated responses in the retrieval corpus. This approach allows GoR to better exploit both content and structural cues for improved summarization performance.

3 Experiments

3.1 Experimental Setup

Datasets. We evaluate our proposed method on four long-context summarization datasets, *i.e.*, AcademicEval (Feng et al., 2024), QMSum (Zhong et al., 2021), WCEP (Gholipour Ghalandari et al., 2020), and BookSum (Kryściński et al., 2021). Among them, AcademicEval collects scientific papers from arXiv for abstract writing, given the long inputs of its main body. QMSum is a query-based summarization dataset, and we **only use** "*general queries*" for evaluating global summarization. WCEP is a multi-document summarization dataset about news events, while BookSum features long-form narrative summarization. For metrics, we adopt Rouge-1 (R-1), Rouge-2 (R-2), and Rouge-L (R-L) (Lin, 2004) to assess the text alignment between the reference summaries and the predicted content generated by GoR.

Implementation Details. Following the standard procedure of RAG, we adopt TokenTextSplitter from LangChain to split each long document into text chunks. Each chunk has a size of 256, and the chunk overlapping is 32. We generate 30 queries for each long document using Mixtral-8x7B-Instruct-v0.1 (Jiang et al., 2024), and the

Model	QMSum			AcademicEval			WCEP			BookSum		
	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2
Node2Vec	18.5	31.8	6.3	19.3	38.3	10.6	13.9	20.1	6.3	13.6	27.4	4.6
BM25	18.4	32.1	6.1	20.4	39.6	11.3	15.5	22.6	7.3	13.7	26.7	4.9
TF-IDF	18.3	31.2	6.3	19.5	38.0	10.6	15.3	22.3	7.3	13.6	26.6	4.9
Contriever	19.1	32.7	7.7	23.6	44.8	16.0	15.7	23.5	7.7	14.4	29.8	5.5
DPR	18.6	32.1	6.7	20.9	41.4	13.2	15.6	22.5	7.5	13.8	27.1	4.8
Dragon	19.2	33.5	7.7	23.5	43.8	15.1	14.6	21.8	6.8	13.7	27.2	4.8
SBERT	19.0	33.0	7.4	23.4	45.2	15.8	13.7	20.5	5.5	14.4	29.5	5.4
BM25+DPR	18.3	31.8	6.6	19.9	39.0	10.8	15.7	22.1	7.6	14.1	28.9	5.4
Gemma-8K	**19.8**	33.5	7.3	21.9	42.0	12.9	15.6	21.9	7.7	12.8	23.4	4.2
Mistral-8K	19.6	31.2	7.2	21.6	41.6	13.1	16.7	24.2	8.8	13.5	26.2	5.3
Full Context	19.4	33.1	6.8	21.5	41.1	12.5	14.4	21.0	7.1	14.4	28.9	5.9
Thought-R	19.0	33.9	7.6	22.0	42.6	13.2	15.2	22.4	7.4	14.2	29.5	5.7
GoR (Ours)	**19.8**	**34.5**	**7.8**	**24.7**	**46.5**	**17.3**	**18.1**	**25.4**	**9.2**	**14.9**	**31.5**	**6.6**

Table 1: **Experimental results on QMSum, AcademicEval, WCEP, and BookSum datasets over long-context global summarization tasks w.r.t. Rouge-L (R-L), Rouge-1 (R-1), and Rouge-2 (R-2)**. Note that the average LLM input token length of GoR and retriever-based baselines is 6×256 ($\approx$1.5K). (**BOLD** indicates the best score)

temperature coefficient is set to 0.5 by default in the query simulation stage. For RAG, we use Contriever (Izacard et al., 2021) for query and document embedding and retrieve 6 text chunks by default, which are fed into LLaMA-2-7b-chat (Touvron et al., 2023) with greedy decoding to generate predicted summaries. In the training stage, we initialize the graph neural network as a two-layer graph attention network (GAT) (Veličković et al., 2017), with a 768-dim hidden dimension following the default setting of most retrievers.

Baselines. To have a comprehensive evaluation, we compare our proposed GoR with dozens of baselines, including (1) **Random Walk based Node Embedding** (*i.e.*, Node2Vec (Grover and Leskovec, 2016)), (2) **Sparse Retriever** (*i.e.*, BM25 (Robertson et al., 2009) and TF-IDF (Ramos et al., 2003)), (3) **Dense Retriever** (*i.e.*, Contriever (Izacard et al., 2021), DPR (Karpukhin et al., 2020), Dragon (Lin et al., 2023), and Sentence-BERT (SBERT) (Reimers and Gurevych, 2019)[1]), (4) **Hybrid Retriever** (*i.e.*, BM25+DPR with Reciprocal Rerank Fusion), (5) **Long-context LLMs** (*i.e.*, Gemma-8K (Team et al., 2024) and Mistral-8K (Jiang et al., 2023a)), (6) **Full Context** (*i.e.*, feeds all inputs to LLMs for summary generation[2]), and (7) **Thought Retriever** (Thought-R) (Feng et al., 2024). Appendix A elucidates more details.

3.2 Main Results

We conduct comprehensive experiments on QMSum, AcademicEval, WCEP, and BookSum datasets compared with dozens of baselines to evaluate the long-context global summarization capabilities of our proposed method. The results are shown in Table 1.

GoR consistently outperforms retriever-based methods. From Table 1, our proposed GoR beats sparse retrievers, dense retrievers, and hybrid retrievers in every aspect. Thanks to the constructed graph, which integrates text chunks from long documents and LLM historical responses into a whole, node embeddings can better reflect the complicated correlations with given queries, thus significantly improving the retrieval performance of GoR. Moreover, the informative content of historical responses may also enhance the summarization task.

GoR shows superiority over long-context LLMs. We compare Gemma-8K and Mistral-8K with a longer context window to accommodate long-context inputs. However, longer inputs may contain minor information, and long-context LLMs struggle with this situation. In contrast, GoR can effectively differentiate key and topic-related content in long texts using learned node embeddings and achieve better results with shorter input lengths.

Additional Findings. (1) Node2Vec produces unsatisfactory results, and the node embeddings

[1] We use all-MiniLM-L6-v2 as the backbone.

[2] If the input length exceeds the context window limit, we randomly sample continuous text spans of maximum length multiple times to feed into LLMs and calculate the avg. result.

QMSum		AcademicEval		WCEP		BookSum	
BM25	**GoR**	BM25	**GoR**	**BM25**	GoR	BM25	**GoR**
38.7%	**61.3%**	30.0%	**70.0%**	**70.4%**	29.6%	35.5%	**64.5%**
TF-IDF	**GoR**	TF-IDF	**GoR**	**TF-IDF**	GoR	TF-IDF	**GoR**
40.6%	**59.4%**	33.3%	**66.7%**	**66.7%**	33.3%	41.9%	**58.1%**
Contriever	GoR	**Contriever**	GoR	Contriever	**GoR**	**Contriever**	GoR
51.6%	48.4%	**53.3%**	46.7%	48.1%	**51.9%**	**54.8%**	45.2%
Gemma-8K	**GoR**	Gemma-8K	**GoR**	Gemma-8K	**GoR**	Gemma-8K	**GoR**
6.5%	**93.5%**	16.7%	**83.3%**	37.0%	**63.0%**	12.9%	**87.1%**
Mistral-8K	**GoR**	Mistral-8K	**GoR**	Mistral-8K	**GoR**	Mistral-8K	**GoR**
12.9%	**87.1%**	40.0%	**60.0%**	37.0%	**63.0%**	22.6%	**77.4%**

Table 2: **LLM Evaluation w.r.t. overall win rates on the QMSum, AcademicEval, WCEP, and BookSum datasets.** Note that there are very few test samples that contain some security-sensitive information that causes DeepSeek-R1 to be unable to return valid evaluation information. We directly skip these samples.

cannot be optimized effectively since it is based on a non-differentiable algorithm. (2) Although Thought Retriever demonstrates competitive results, it is still inferior to GoR due to the lack of exploration of the correlations between retrieved text chunks and LLM-generated responses. (3) Since the context window length limit of LLMs is exceeded, "Full Context" truncates the long-context input, thus losing some information that may be important for global summarization, resulting in suboptimal results.

Overall, GoR achieves the best results compared with various baselines, demonstrating the effectiveness of our proposed method.

3.3 LLM Evaluation

To enable a more comprehensive automatic evaluation of GoR, inspired by LLM-as-a-Judge (Gu et al., 2024), we adopt DeepSeek-R1 (Guo et al., 2025) to assess the summaries generated by GoR and competitive baselines. Following (Edge et al., 2024) and (Guo et al., 2024), we evaluate from three perspectives: comprehensiveness, diversity, and empowerment. The LLM is instructed to provide an overall judgment based on these criteria to determine the better summary.

To reduce evaluation costs, we select representative and strong baselines for LLM evaluation on the QMSum, AcademicEval, WCEP, and BookSum datasets. Table 2 reports the overall win rates, *i.e.*, the proportion of summaries judged better under pairwise comparison. Evaluation prompts are provided in Appendix D.

From Table 2, we observe the following. (1) GoR consistently outperforms other methods across all datasets, demonstrating stronger comprehensiveness, diversity, and informativeness. (2) Contriever performs comparably, with results close to GoR. (3) Long-context LLMs like Gemma-8K and Mistral-8K fall significantly behind, suggesting that GoR's graph-based retrieval yields more relevant and refined content, especially when input length is limited.

In summary, GoR delivers superior performance in LLM evaluation compared to other baselines.

3.4 Ablation Study

To investigate how each component of GoR contributes to its performance, we conduct an ablation experiment, and the results are shown in Table 3.

From Table 3, we can draw several conclusions. (1) Directly using the text embeddings from the retriever without training leads to degraded performance (*i.e.*, w/o train), highlighting the effectiveness of the learned node embeddings. (2) Both the contrastive loss $\mathcal{L}_{\text{CL}}$ and pair-wise ranking loss $\mathcal{L}_{\text{RANK}}$ significantly improve performance. The pair-wise ranking loss imposes stricter ranking constraints on node embeddings, making effective use of the indirect supervision signal from the self-supervised reference summaries. (3) In-batch negatives are crucial to the performance of contrastive learning. Removing in-batch negatives (*i.e.*, w/o in-b neg) leads to a significant drop in results. (4) Compared with self-supervised training, we utilize global reference summaries as labels to conduct supervised training (*i.e.*, w/ sup), and the results are significantly worse than the self-supervised setting. We will further discuss it in Section 3.5.

In general, GoR's reasonable module design enables it to achieve superior performance.

Variant	WCEP			BookSum		
	R-L	R-1	R-2	R-L	R-1	R-2
w/o train	15.3	22.4	7.4	13.7	27.7	4.7
w/o $\mathcal{L}_{\text{CL}}$	14.7	21.9	7.2	14.1	28.8	5.1
w/o $\mathcal{L}_{\text{RANK}}$	16.6	24.2	8.2	14.0	28.0	4.9
w/o in-b neg	17.2	24.9	8.8	13.3	26.3	5.2
w/ sup	15.5	22.8	7.3	13.8	29.0	5.2
GoR	**18.1**	**25.4**	**9.2**	**14.9**	**31.5**	**6.6**

Table 3: **Ablation study on the WCEP and BookSum datasets w.r.t. R-L, R-1, and R-2**.

3.5 Discussions

Impact of the Number of Simulated Queries During Training. Query Simulation is a crucial stage in our method design, and we will examine how the number of simulated queries used during training affects learning performance. In particular, we explore this effect by gradually increasing the number of simulated queries used in training. We present the results in Figure 2. Overall, R-L shows an upward trend as the number of simulated queries increases. Nevertheless, since fewer queries cover less relevant content from long documents, the curves of each dataset have some fluctuations, indicating the occurrence of underfitting.

In general, 30 simulated queries can optimize the model well across these four datasets, which indicates that our proposed GoR is cost-effective. Nevertheless, increasing the number of simulated queries may still potentially further improve the performance of the model. Due to budget constraints, we will leave this for future work.

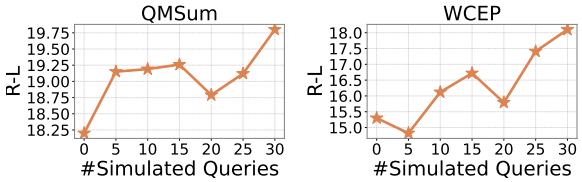

Figure 2: **Impact of the number of simulated queries during training w.r.t. R-L**. We show the results on the QMSum and WCEP datasets.

Supervised Training on Global Summarization Queries. To dive deeper into the differences between self-supervised and supervised training, we carry out additional experiments using global reference summaries. Specifically, we utilize global summarization queries and reference summaries to serve as a training corpus under the supervised setting. As there is only one global summarization query for each long document, we replicate it multiple times to match the quantity of self-supervised training data, thus eliminating the impact of the quantity difference. We present the results on the BookSum dataset in Figure 3, and the *Entropy* denotes the entropy of the similarity distribution between queries and node embeddings.

From Figure 3, it is evident that in the self-supervised setting, the loss is consistently lower than in the supervised setting. This suggests that the global reference summaries are highly correlated with many nodes, causing most nodes to exhibit a high semantic similarity with the global query. As a result, this confuses the model's optimization direction. Additionally, the entropy curve shows that the entropy in the supervised setting is consistently higher than in the self-supervised setting, indicating that the model struggles to select the most similar node. In contrast, the self-supervised label, derived from a specific part of a long document, contains more focused content, effectively guiding the model's optimization.

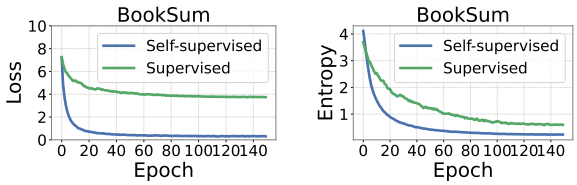

Figure 3: **Differences between self-supervised and supervised training w.r.t. loss and entropy on the BookSum dataset.**

Inference Efficiency Analysis. To investigate the inference efficiency of GoR, we conduct extensive experiments on the WCEP dataset and present the results w.r.t. the inference time per query in Table 4. Note that since the LLM used in our experiment is consistent, we ignore the inference time brought by the LLM itself.

From Table 4, we can draw the following conclusion. (1) Since GoR's only trainable module is GNN, GoR's inference efficiency is very high, and almost no additional noticeable latency is introduced. (2) Although GoR's inference time is longer than some baselines, it only increases by a few hundred milliseconds. Considering the significant performance improvement brought by GoR in Table 1, this tiny time overhead is almost negligible in practical applications.

Baselines	Node2Vec	BM25	Contriever	SBERT	BM25+DPR	Thought-R	**GoR (ours)**
Inference Time (s)	9.4	0.02	0.20	0.01	0.04	0.3	0.58

Table 4: **Inference efficiency analysis w.r.t. inference time per query on the WCEP dataset.**

4 Related Work

Long-context Summarization using LLMs. In recent years, LLMs have shown impressive capabilities in long-context modeling (Achiam et al., 2023; AI@Meta, 2024; Team et al., 2024; Jiang et al., 2024). Summarizing lengthy documents with LLMs primarily involves two approaches: retrieval-augmented generation (RAG) (Ram et al., 2023; Yu et al., 2023) and long-context LLMs (GLM et al., 2024; Li* et al., 2023; Tworkowski et al., 2023). Long-context LLMs feature a large context window length to accommodate long-context inputs. However, they may suffer from severe performance degradation when accessing some local key details in the middle of long contexts (Liu et al., 2024b). Conversely, RAG emerges as a promising approach for cost-effective long-context summarization. By equipping with a retriever (Karpukhin et al., 2020; Robertson et al., 2009), RAG can first perform a relevance search based on user queries and then feed the retrieved text into LLMs for summary. For a recent example, GraphRAG (Edge et al., 2024) conducts query-focused summarization by setting up a graph index and detecting graph communities for summary generation.

Nevertheless, most current RAG approaches still focus on enhancing LLMs' reasoning and question-answering capabilities, which only require retrieving locally relevant information (Trivedi et al., 2022; Jiang et al., 2023b; Asai et al., 2023; Li et al., 2023; Zheng et al., 2023). In comparison, our proposed method stands out from these methods by focusing on LLMs' global summarization capability of long-context inputs.

Graph-assisted Retrieval-augmented Language Models. As one of the effective structures for modeling data relations, graphs have recently been used to enhance the performance of retrieval-augmented language models on various QA tasks. EtD (Liu et al., 2024a) features a graph neural network (GNN) to traverse a knowledge graph hop by hop to discover more relevant knowledge, thus enhancing LLM generation quality. GNN-RAG (Mavromatis and Karypis, 2024) learns to reason over graphs using GNNs, and retrieves answer candidates for a given question. PG-RAG (Liang et al., 2024) constructs pseudo-graphs with a retrieval indexer by prompting LLMs to organize document knowledge in a self-learning manner. G-RAG (Dong et al., 2024) proposes to rerank documents by learning graph representation on abstract meaning representation graphs, while GNN-Ret (Li et al., 2024b) refines semantic distances between documents and queries by modeling relationships among related passages. ToG (Sun et al., 2023) and KGP (Wang et al., 2024) treat LLMs as agents to traverse and reason over knowledge graphs in an iterative way, while RoG (Luo et al., 2023) first generates plans for retrieval and then conducts reasoning. G-Retriever (He et al., 2024) transforms retrieved knowledge subgraphs into graph embeddings by training a graph encoder and textualizes subgraphs to serve as inputs of LLMs.

Different from the above, GraphRAG (Edge et al., 2024) sets up a graph index for query-focused summarization, which aligns more closely with our approach. Compared with GraphRAG, which is time-consuming and suffers from huge computational costs, GoR is lightweight and only draws on a few LLM historical responses with efficient training to achieve competitive performance.

5 Conclusion

In this work, we introduce a method named *graph of records* (GoR) to improve long-context global summarization in retrieval-augmented generation by utilizing LLM-generated historical responses. Intuitively, we establish connections between text chunks retrieved from long documents and LLM-generated historical responses to create a graph of records. To uncover complex correlations between these connections, we use a graph neural network and develop a BERTScore-based objective for self-supervised training, enabling seamless supervision signal backpropagation between self-supervised reference summaries and node embeddings. Our experiments on four long-context summarization datasets show that GoR significantly outperforms various baselines, demonstrating its effectiveness.

6 Limitations

Despite the superiority of our proposed method, GoR has some limitations. (1) Due to a limited budget, we only simulate and generate a small number of user queries, which may cause a bottleneck in further model optimization. (2) The simulated queries may not accurately reflect the real-world distribution, as they do not account for the possibility of users asking many meaningless questions. Therefore, a filtering process may be necessary, which we leave for future work.

To promote sharing and communication in the academic community, we also share some insights about simulated queries here. Given the powerful evaluation capabilities of LLMs, many works utilize LLM-as-a-Judge and treat LLMs as evaluators. Intuitively, in practical applications, we can first use some simple rule-based filtering strategies to preliminarily screen the meaningless questions raised by users, and then use LLM-as-a-Judge to evaluate the remaining questions and judge their quality from multiple dimensions (such as diversity, complexity, inspiration, etc.), which not only takes into account performance but also ensures a certain efficiency in real multi-user scenarios.

Acknowledgments

We sincerely appreciate the support from Amazon grant funding project #120359, "GRAG: Enhance RAG Applications with Graph-structured Knowledge", and Meta gift funding project "PERM: Toward Parameter Efficient Foundation Models for Recommenders"

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI@Meta. 2024. Llama 3 model card.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

Jialin Dong, Bahare Fatemi, Bryan Perozzi, Lin F Yang, and Anton Tsitsulin. 2024. Don't forget to connect! improving rag with graph-based reranking. *arXiv preprint arXiv:2405.18414*.

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. *arXiv preprint arXiv:2404.16130*.

Tao Feng, Pengrui Han, Guanyu Lin, Ge Liu, and Jiaxuan You. 2024. Thought-retriever: Don't just retrieve raw data, retrieve thoughts. In *International Conference on Learning Representations Workshop: How Far Are We From AGI*.

Demian Gholipour Ghalandari, Chris Hokamp, Nghia The Pham, John Glover, and Georgiana Ifrim. 2020. A large-scale multi-document summarization dataset from the Wikipedia current events portal. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1302–1308, Online. Association for Computational Linguistics.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM international conference on Knowledge discovery and data mining*, pages 855–864.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Yuanzhuo Wang, and Jian Guo. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv: 2411.15594*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma,

Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Annual Conference on Neural Information Processing Systems*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. *Preprint*, arXiv:2104.02112.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2021. Booksum: A collection of datasets for long-form narrative summarization. *arXiv preprint arXiv:2105.08209*.

Dacheng Li*, Rulin Shao*, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E. Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. 2023. How long can open-source llms truly promise on context length?

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024a. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2023. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. *arXiv preprint arXiv:2305.13269*.

Zijian Li, Qingyan Guo, Jiawei Shao, Lei Song, Jiang Bian, Jun Zhang, and Rui Wang. 2024b. Graph neural network enhanced retrieval for question answering of llms. *arXiv preprint arXiv:2406.06572*.

Xun Liang, Simin Niu, Sensen Zhang, Shichao Song, Hanyu Wang, Jiawei Yang, Feiyu Xiong, Bo Tang, Chenyang Xi, et al. 2024. Empowering large language models to set up a knowledge retrieval indexer via self-learning. *arXiv preprint arXiv:2405.16933*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Sheng-Chieh Lin, Akari Asai, Minghan Li, Barlas Oguz, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2023. How to train your dragon: Diverse augmentation towards generalizable dense retrieval. *arXiv preprint arXiv:2302.07452*.

Guangyi Liu, Yongqi Zhang, Yong Li, and Quanming Yao. 2024a. Explore then determine: A gnn-llm synergy framework for reasoning over knowledge graph. *arXiv preprint arXiv:2406.01145*.

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. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*.

Costas Mavromatis and George Karypis. 2024. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*.

Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to doctttttquery. *Online preprint*, 6(2).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *arXiv preprint arXiv:2307.07697*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2023. Focused transformer: Contrastive training for context scaling. *Preprint*, arXiv:2307.03170.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748v2*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Alex Wang, Richard Yuanzhe Pang, Angelica Chen, Jason Phang, and Samuel R Bowman. 2022a. Squality: Building a long-document summarization dataset the hard way. *arXiv preprint arXiv:2205.11465*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pages 6861–6871.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *International Conference on Learning Representations*.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. 2024. Buffer of thoughts: Thought-augmented reasoning with large language models. *arXiv preprint arXiv:2406.04271*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. 2023. Augmentation-adapted retriever improves generalization of language models as generic plug-in. *arXiv preprint arXiv:2305.17331*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H Chi, Quoc V Le, and Denny Zhou. 2023. Take a step back: Evoking reasoning via abstraction in large language models. *arXiv preprint arXiv:2310.06117*.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. *arXiv preprint arXiv:2104.05938*.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. *ProQuest number: information to all users*.

A Experimental Details

A.1 Dataset

We present dataset statistics in Table 5. Due to the limited budget, we randomly select training and test samples for the training and test set and calculate the average input and output token lengths using the LLaMA-2 tokenizer (Touvron et al., 2023) (samples with short input lengths are filtered out).

We evaluate our proposed method on four long-context summarization datasets, *i.e.*, AcademicEval (Feng et al., 2024), QMSum (Zhong et al., 2021), WCEP (Gholipour Ghalandari et al., 2020), and BookSum (Kryściński et al., 2021).

- **QMSum** (Zhong et al., 2021). QMSum is a query-based summarization dataset that features lengthy meeting transcripts, specific queries, and general queries. Specific queries focus on query-based summarization, and general queries are questions that summarize the entire meeting transcript, such as "Summarize the whole meeting." We **only use** "*general queries*" for evaluating global summarization.

- **AcademicEval** (Feng et al., 2024). AcademicEval collects scientific papers from arXiv for abstract and related work writing. We use the abstract writing subset, which provides the main body of a paper as input and generates the predicted abstract.

- **WCEP** (Gholipour Ghalandari et al., 2020). WCEP is a multi-document summarization dataset about news events, which requires comprehensive consideration of the contents of multiple documents.

- **BookSum** (Kryściński et al., 2021). BookSum features long-form narrative summarization, which covers source documents from the literature domain and includes highly abstractive human-written summaries.

A.2 Baselines

We present detailed descriptions of adopted baselines.

- **Node2Vec** (Grover and Leskovec, 2016). Node2Vec generates node embeddings for graphs by simulating biased random walks to capture both local and global structural properties of nodes.

- **BM25** (Robertson et al., 2009), **TF-IDF** (Ramos et al., 2003). BM25 ranks documents based on term frequency, inverse document frequency, and document length normalization, while TF-IDF evaluates the importance of a term in a document relative to a corpus by combining term frequency and inverse document frequency.

- **Contriever** (Izacard et al., 2021), **DPR** (Karpukhin et al., 2020), **Dragon** (Lin et al., 2023), **SBERT** (Reimers and Gurevych, 2019). Contriever is a self-supervised dense retriever that learns unsupervised document embeddings for information retrieval, DPR (Dense Passage Retriever) is a bi-encoder model that retrieves relevant passages by training on question-passage pairs, Dragon is a dense retrieval model optimized through diverse augmentation for generalizable dense retrieval, and SBERT (Sentence-BERT) is a modification of BERT that generates semantically meaningful sentence embeddings for tasks like similarity and clustering using a siamese network structure.

- **BM25+DPR**. BM25+DPR with Reciprocal Rerank Fusion is a hybrid retrieval method that combines the strengths of BM25's lexical matching and DPR's dense embeddings by reranking results from both models using a reciprocal rank fusion strategy to improve retrieval accuracy.

- **Gemma-8K** (Team et al., 2024), **Mistral-8K** (Jiang et al., 2023a). Gemma-8K and Mistral-8K are LLMs with relatively long context window lengths.

- **Full Context**. We feed all inputs to LLMs for summary generation. If the input length exceeds the context window limit, we randomly sample continuous text spans of maximum length multiple times to feed into LLMs and calculate the average result.

- **Thought-R** (Feng et al., 2024). Thought Retriever (Thought-R) generates thoughts for a series of simulated queries and appends them to the retrieval corpus as high-level knowledge.

Dataset	#Train	#Test	Average Input Token Length	Average Output Token Length
QMSum (Zhong et al., 2021)	162	30	17K	0.1K
AcademicEval (Feng et al., 2024)	400	30	13K	0.3K
WCEP (Gholipour Ghalandari et al., 2020)	400	30	11K	0.05K
BookSum (Kryściński et al., 2021)	400	30	16K	1K

Table 5: Dataset statistics

A.3 Additional Explanation on Training Objective

Given a graph that consists of document chunks and response nodes, we expect that the learned node embeddings $\mathbf{h}_v^{(L)}$ can adaptively reflect the semantic similarity to a given query $\mathbf{q}$. In other words, we expect that we can select the node $\mathbf{v}$ with the largest semantic similarity to $\mathbf{q}$ according to the formula $\mathrm{Sim}(\mathbf{q}, \mathbf{v}) = \mathbf{E_q}(\mathbf{q})^T \cdot \mathbf{h}_v^{(L)}$. To this end, we need to find out which node has the highest semantic similarity with $\mathbf{q}$ and use this as a supervision signal for model optimization. Therefore, we utilize BERTScore (Zhang et al., 2019) to obtain a node ranking list $\mathbf{M}_i$, which exactly serves as supervision signals.

$$\mathbf{M}_i = [\mathbf{h}_+^{(L)}, \mathbf{h}_1^{(L)}, \cdots, \mathbf{h}_{|\mathbf{C}|+|\mathbf{T}|}^{(L)}] \tag{9}$$

For contrastive loss in Equation 6, we regard $\mathbf{h}_+^{(L)}$ as the positive and $[\mathbf{h}_1^{(L)}, \cdots, \mathbf{h}_{|\mathbf{C}|+|\mathbf{T}|}^{(L)}]$ as the negatives to conduct contrastive learning (van den Oord et al., 2018). For a given query $\mathbf{q}$, the contrastive learning objective will bring $\mathbf{E_q}(\mathbf{q})$ and $\mathbf{h}_+^{(L)}$ closer in the semantic embedding space while increasing the distance between $\mathbf{E_q}(\mathbf{q})$ and $[\mathbf{h}_1^{(L)}, \cdots, \mathbf{h}_{|\mathbf{C}|+|\mathbf{T}|}^{(L)}]$ (the symbols are simplified here for convenience of description).

Similarly, in Equation 7, we expect to impose a stricter constraint on the learned node embedding based on the node ranking list $\mathbf{M}_i$. Although Equation 6 shortens the semantic distance between $\mathbf{E_q}(\mathbf{q})$ and $\mathbf{h}_+^{(L)}$, it does not take into account the relative ranking between negative samples. For example, the semantic similarity between $\mathbf{E_q}(\mathbf{q})$ and $\mathbf{h}_1^{(L)}$ is higher than that between $\mathbf{h}_2^{(L)}$. Formally, given $\mathbf{h}_i^{(L)}, \mathbf{h}_j^{(L)} \in \mathbf{M}_i$ that satisfies $\mathrm{rank}(\mathbf{h}_j^{(L)}) > \mathrm{rank}(\mathbf{h}_i^{(L)})$, Equation 7 will explicitly optimize in the direction of $\mathbf{E_q}(\mathbf{q})^\top \mathbf{h}_j^{(L)} < \mathbf{E_q}(\mathbf{q})^\top \mathbf{h}_i^{(L)}$, thus imposing stricter constraints to the pair-wise ranking.

A.4 Additional Implementation Details

In the stage of graph construction, due to the number and randomness of the simulated queries, there may be some isolated nodes, and we just keep them in the graph with self-loop edges. During model optimization, BERTScore is pre-computed for efficient training.

In the training stage, we use the Adam optimizer for model training and gradually decay the learning rate from 1e-3 to 0 with the LambdaLR scheduler. We present detailed hyper-parameters on QMSum, AcademicEval, WCEP, and BookSum datasets in Table 6. We implement our proposed method using PyTorch and Deep Graph Library (DGL), and all the experiments are conducted on a single RTX 3080 GPU. As for LLMs, we rely on API calling from Together AI[3] to obtain responses.

For metrics, we adopt Rouge-1 (R-1), Rouge-2 (R-2), and Rouge-L (R-L) (Lin, 2004) to assess the text alignment between the reference summaries and the predicted content generated by our proposed method. If a global summarization query has multiple reference summaries, we calculate the Rouge-L/1/2 of the predicted summary and all references, respectively, and take the maximum value as the final evaluation result. We follow this setting in all experiments, including the baseline evaluation.

A.5 More Comparison Experiments

We conduct extensive experiments on GovReport (Huang et al., 2021) and SQuALITY (Wang et al., 2022a) datasets, and the results are shown in Table 7, which demonstrate our proposed GoR is still competitive among baselines on these two datasets.

A.6 More Ablation Experiments

We conduct extensive ablation experiments on QMSum (Zhong et al., 2021) and AcademicEval (Feng

[3] https://www.together.ai/

Datasets	QMSum	AcademicEval	WCEP	BookSum
#GAT Layers	2	2	2	2
#GAT Heads	4	4	4	4
Batch Size	32	32	32	32
Epoch	150	150	150	150
Learning Rate	1e-3	1e-3	1e-3	1e-3
Hidden Dimension	768	768	768	768
Dropout Rate	0.2	0.0	0.1	0.2
Loss Coefficient α	0.9	0.6	0.7	0.2

Table 6: Hyper-parameters

Model	GovReport			SQuALITY		
	R-L	R-1	R-2	R-L	R-1	R-2
Node2Vec	18.1	36.7	12.4	17.0	32.9	7.7
BM25	18.2	39.2	13.0	17.0	31.4	8.1
TF-IDF	18.1	39.2	12.8	17.0	31.4	8.1
Contriever	20.2	39.8	**17.6**	16.8	32.6	8.3
DPR	19.1	39.4	15.5	17.4	33.1	8.4
Dragon	19.6	38.2	16.0	16.2	29.6	7.5
SBERT	20.0	39.8	15.8	17.1	32.1	7.8
BM25+DPR	19.4	37.4	15.0	16.6	31.5	7.4
Gemma-8K	17.4	33.8	11.4	12.9	19.7	5.8
Mistral-8K	16.0	28.9	9.4	16.9	32.2	8.1
Full Context	18.4	39.1	13.8	**17.8**	**34.0**	**8.8**
Thought-R	20.4	40.3	17.0	17.3	32.0	8.0
GoR (Ours)	**20.9**	**41.4**	16.8	**17.8**	**34.0**	8.5

Table 7: **Experimental results on GovReport and SQuALITY datasets over long-context global summarization tasks w.r.t. Rouge-L (R-L), Rouge-1 (R-1), and Rouge-2 (R-2)**. Note that the average LLM input token length of GoR and retriever-based baselines is 6×256, which is about 1.5K. (**BOLD** indicates the best score)

et al., 2024) datasets, and the results are shown in Table 8.

Variant	QMSum			AcademicEval		
	R-L	R-1	R-2	R-L	R-1	R-2
w/o train	18.2	33.0	7.6	23.3	45.0	15.5
w/o $\mathcal{L}_{\text{CL}}$	18.4	33.3	6.9	23.5	44.9	15.5
w/o $\mathcal{L}_{\text{RANK}}$	19.6	33.1	**7.8**	23.1	44.4	15.1
w/o in-b neg	**19.8**	**34.7**	**7.8**	24.5	46.4	16.5
w/ sup	18.1	32.3	6.9	21.4	43.3	13.9
GoR	**19.8**	34.5	**7.8**	**24.7**	**46.5**	**17.3**

Table 8: **Ablation study on QMSum and AcademicEval datasets w.r.t. R-L, R-1, and R-2**.

A.7 Impact of GNN Architectures

GNNs play a vital role in learning node embeddings. we explore various GNN architectures to study their impact on learning node embeddings, including GCN (Kipf and Welling, 2016), SGC (Wu et al., 2019), GIN (Xu et al., 2019), and GraphSAGE (Hamilton et al., 2017). Our findings, illustrated in Figure 4, show that GAT outperforms the other architectures. This is because GAT considers the significance of neighboring nodes when updating node embeddings, allowing the model to effectively capture essential information from the nodes. Among the other architectures, GraphSAGE performs poorly due to its unstable neighbor sampling mechanism.

Overall, GAT reaches the best results, which shows that considering the importance of neighboring nodes is effective in mining complicated correlations and is critical to improving performance.

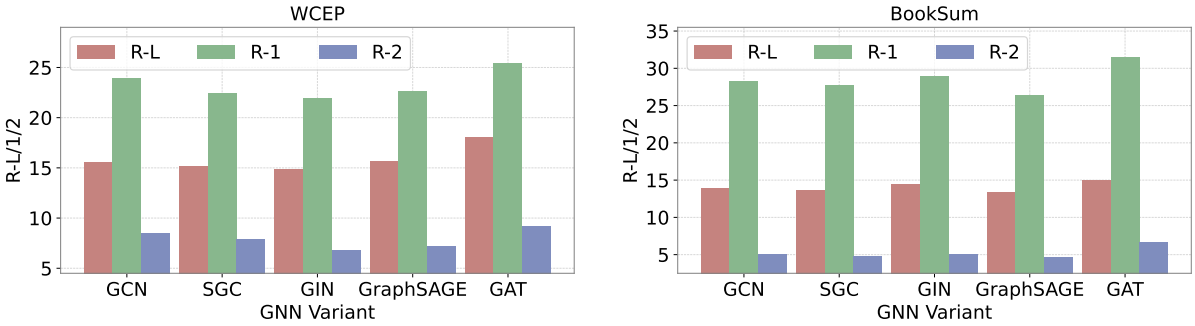

Figure 4: **Impact of GNN architectures w.r.t. R-L, R-1, and R-2**. The left figure shows results on the WCEP dataset, while the right one shows results with the BookSum dataset.

A.8 Impact of the Number of Simulated Queries During Training

We show additional results on the AcademicEval and BookSum datasets in Figure 5.

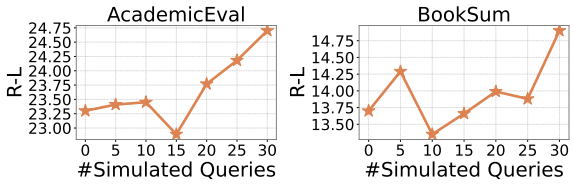

Figure 5: **Impact of the number of simulated queries during training w.r.t. R-L**. We show the additional results on the AcademicEval and BookSum datasets.

A.9 Supervised Training on Global Summarization Queries

We show additional results on the WCEP dataset in Figure 6.

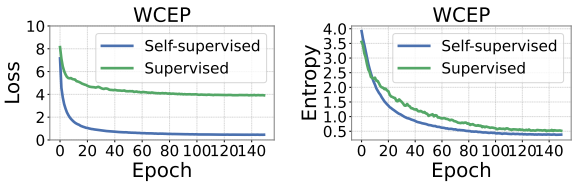

Figure 6: **Differences between self-supervised and supervised training w.r.t. loss and entropy**. We show the loss and entropy curve during training on the WCEP dataset.

B Additional Related Work

Historical Response Utilization of LLMs. Little work has been done on this under-explored topic. Thought-retriever (Feng et al., 2024) saves the historical responses of each user-LLM interaction as high-level and informative thoughts to expand the retrieval corpus for future user queries. However, the intricate correlations among thoughts are neglected, leaving room for further improvement.

Another line of work is the Chain-of-Thought (CoT), which is similar to our approach in terms of utilizing LLM historical responses and has been regarded as an effective means to enhance the reasoning ability of LLM during inference time in recent years. Few-shot CoT (Wei et al., 2022) and zero-shot CoT (Kojima et al., 2022) elicit intermediate reasoning paths by prompting LLMs with several demonstrations or just appending *"Let's think step by step."* Self-consistency (Wang et al., 2022b) samples diverse reasoning paths and conducts a majority vote to obtain the final answer. ToT (Yao et al., 2024) and GoT (Besta et al., 2024) take a further step by integrating a tree or graph structure to manage its historical reasoning paths, enabling more flexible reasoning and reflection of LLMs. By memorizing the solution paradigms of various queries into different templates, BoT (Yang et al., 2024) pushes LLM's reasoning ability to a new level. Although the above-mentioned CoT-series approaches improve the reasoning capabilities of LLMs by utilizing chains of intermediate reasoning responses, they usually concentrate on one specific QA-type query and cannot generalize to and benefit other queries. Moreover, only a small number of historical reasoning responses are retained for the final generation, while most of the rest are just discarded.

C Case Study

In this section, we provide a case study of GoR and other baseline methods' summarization.

The Summary Generated by GoR

An asteroid called 1999 KW4 will make a close approach to Earth this weekend, with its own small moon in tow. The asteroid is estimated to be around 1.5 km wide and will be roughly 3.2 million miles from Earth at its closest point. NASA astronomers are interested in studying the asteroid due to its close proximity and the fact that it is a binary system, meaning it has its own moon. The last time the asteroid will make a close approach to Earth will be in 2036. Additionally, a recent study using data from NASA's Kepler Space Telescope suggests that comets may have delivered water to Earth, as the ratio of two types of water molecules on comets matches that in Earth's oceans. The new algorithm used in the study is more sensitive to small planets the size of Earth and could help in the search for Earth-like planets.

The Summary Generated by Contriever

Asteroid 2019 JH7 recently flew past Earth, and NASA observed that the asteroid's trajectory falls under the "Earth Close Approach" category. The observations made by NASA's Jet Propulsion Laboratory (JPL) in Pasadena, California, suggest that comets could have delivered water to Earth. The study found that the ratio of two types of water molecules on the comet matches that in the Earth's oceans, which could imply that comets delivered a large fraction of water to the outer reaches of the solar system. The observations made using data from the SOFIA telescope suggest that all comets could have a heavy-to-regular water ratio similar to Earth's oceans, which could imply that comets delivered some water to Earth. Previously, measuring this ratio was difficult, and ground and space telescopes could only study this level of detail in comets when they pass near Earth.

Reference Summary

Binary Aten asteroid (66391) 1999 KW4 and its minor-planet moon make their closest-ever recorded flyby of Earth at 3.2 million miles away. The asteroid will approach even closer at 0.0155 AU (2,320,000 km) from Earth in 2036, and is the largest asteroid to approach Earth until (4953) 1990 MU in June 2027.

From the above example, we can draw conclusions. (1) GoR summarizes several keywords that appear in the reference summary, such as "1999 KW4" and "3.2 million miles", etc., but Contriever fails to extract this crucial information. (2) From a global perspective, the summary generated by GoR is more relevant and consistent with the reference summary. However, the summary generated by Contriever focuses too much on local details and ignores the main idea of the original article.

D LLM Prompts

In this section, we present LLM prompts used in GoR, including user query simulation, RAG, and LLM-as-a-Judge prompts.

D.1 LLM Prompts for User Query Simulation

Prompt for User Query Simulation

You are a great questioner of any text, and are adept at asking valuable and insightful questions. Your goal is to generate 1 summary question for the text provided below. The generated summary question should try to simulate the tone of human questions as much as possible, and make sure that the generated question must be interrogative sentences and a summary question. Important! Please make sure this text must be a complete and non-redundant answer to the generated summary question. Please directly output the generated summary question, do not output irrelevant text.

DOCUMENT:
{document}

D.2 LLM Prompts for RAG

RAG Prompt

Refer to the following supporting materials and answer the question with brief but complete explanations.

SUPPORTING MATERIALS:
{materials}

QUESTION:
{question}

D.3 LLM Prompts for LLM-as-a-Judge

We construct LLM-as-a-Judge prompts following (Edge et al., 2024) and (Guo et al., 2024) with some minor changes.

LLM-as-a-Judge Prompt - Instruction

—Role—
You are an expert tasked with evaluating two answers to the same question based on four criteria: Comprehensiveness, Diversity, and Empowerment.

—Goal—
You will evaluate two answers to the same question based on four criteria: Comprehensiveness, Diversity, and Empowerment.

Comprehensiveness: How much detail does the answer provide to cover all aspects and details of the question?
Diversity: How varied and rich is the answer in providing different perspectives and insights on the question?
Empowerment: How well does the answer help the reader understand and make informed judgments about the topic?

For each criterion, choose the better answer (either Answer 1 or Answer 2) and explain why. Then, select an overall winner based on these three categories.

LLM-as-a-Judge Prompt - Input

Here is the question: {query}

Here are the two answers: Answer 1: {answer1}; Answer 2: {answer2}

Evaluate both answers using the three criteria listed above and provide detailed explanations for each criterion.

Avoiding any potential bias and ensuring that the order in which the answers were presented does not affect your judgment.

Output your evaluation in the following JSON format:

{{ "Comprehensiveness": {{ "Winner": "[Answer 1 or Answer 2]", "Explanation": "[Provide explanation here]" }},
"Diversity": {{ "Winner": "[Answer 1 or Answer 2]", "Explanation": "[Provide explanation here]" }},
"Empowerment": {{ "Winner": "[Answer 1 or Answer 2]", "Explanation": "[Provide explanation here]" }},
"Overall Winner": {{"Winner": "[Answer 1 or Answer 2]", "Explanation": "[Summarize why this answer is the overall winner based on the three criteria]" }} }}

E Broader Impacts

In the era of LLMs, countless interactions take place between users and these models on a daily basis, resulting in the generation of a vast amount of historical responses. Our proposed method demonstrates that these historical responses hold significant potential and can be effectively leveraged to further improve the quality of future responses generated by LLMs. By analyzing and reusing these past outputs, we can not only refine and enhance the overall performance of the models but also reduce computational overhead. This approach highlights the untapped value of historical data in optimizing response generation while making the process more efficient and resource-friendly.