RAG for Source Text Summarization

Hamdi Berke GÖÇMEN
TUM CIT I11
Boltzmannstraße 3
D-85748 Garching
berke.gocmen@tum.de

Guangyao QUAN
TUM CIT I11
Boltzmannstraße 3
D-85748 Garching
guangyao.quan@tum.de

Luca Mattes WIEHE
TUM CIT I11
Boltzmannstraße 3
D-85748 Garching
luca.wiehe@tum.de

Abstract

In recent years, large pre-trained Language Models (LLMs) have garnered significant interest in the research community. While state-of-the-art LLMs achieve remarkable results in various tasks, they are prone to hallucination when responding to queries. Retrieval-Augmented Generation (RAG) is a framework that uses context chunks from a knowledge base to decrease the likelihood of hallucination.

This work adapts the RAG pipeline in two different ways to use it for source text summarization. In our first approach, we replace the user query with a topic description of the dominant topic. This topic description provides valuable information to retrieve meaningful chunks from the knowledge base. Our second approach uses extractive summarization to extract chunks similar to the extractive summary. This way, we make sure that the summarization is adapted using context information. Applying these approaches to the XSUM dataset demonstrates that performance is very similar across all RAG-based approaches. However, compared to a plain GPT baseline without augmented context, all RAG-based approaches show a significant gain in summarization quality.

1 Introduction

Natural Language Processing (NLP) is an application of Machine Learning to problems from humans' predominant form of communication: Language. In recent years, NLP has been dominated by Transformers (Vaswani et al., 2017), an architecture that proves to be particularly powerful for sequence-to-sequence modeling. Since language can be modeled as a sequence of words, the majority of state-of-the-art (SOTA) language models build on the Transformer architecture. Recently published language models include GPT-4 (Achiam et al., 2023), Mistral (Jiang et al., 2023), Llama3 (MetaAI, 2024), and Zephyr (Tunstall et al., 2023). Each of these LLMs demonstrates strong capabilities across tasks such as solving math problems, text understanding, and summarization tasks, among others.

Despite their strong performance across domains, hallucinations as part of LLMs are a well-known problem (Zhang et al., 2023; Rawte et al., 2023). We can pinpoint two main reasons for hallucination:

1. **The query is too short**: The query needs to contain more information to answer it solely based on the information contained in the query. In this case, the LLM will use knowledge it was trained on to fill gaps that cannot be answered based on the query only. This process will be lossful, resulting in hallucination because sequence-to-sequence models are probabilistic (Xu et al., 2024)
2. **The query is too long**: A long query introduces more complexity and noise. The model may incorrectly assess aspects' importance or infer relationships between text pieces that don't exist.

This problem is particularly relevant for tasks relying on correctness and factfulness. One of these tasks is source text summarization, i.e., extracting core information of long text in a few sentences or paragraphs. Source text summarization is the task domain that we want to focus on in this work.

When summarizing texts, we can generally distinguish two summarization types: extractive and abstractive. In extractive summarization, we aim to summarize text using only text snippets from the reference text. In abstractive summarization, we want to summarize text while allowing it to rephrase its content. Dealing with long text is a challenge in summarization tasks (Wibawa et al., 2024). Since LLMs are probabilistic when generating summaries, they are better suited for abstractive summarization. We have to pass a whole text as a query for summarization, resulting in a long query. As mentioned earlier, long queries imply a higher likelihood of hallucinations.

Retrieval-Augmented Generation (RAG) tries to tackle this issue and increase the groundedness of an LLM's response (Lewis et al., 2020). Traditionally, RAG is applied to the task of knowledge retrieval. Here, we aim to decrease the likelihood of hallucination by providing additional context that we consider particularly relevant to a user's query. There are several variations to the traditional RAG pipeline (Eibich et al., 2024), each of which has its own strengths and weaknesses. In this work, we want to apply the RAG pipeline to the task of source text summarization. We hypothesize that we can increase the groundedness of LLM-generated summaries by providing context chunks from our source documents. There is some concurrent work on this topic (Edge et al., 2024), but we will propose two new approaches that tackle the weaknesses of the traditional RAG pipeline.

2 Dataset

The primary objective of this study is to assess the compatibility of RAG with summarization tasks. While there is a wide range of summarization datasets available, we have chosen to conduct our evaluation using the XSUM dataset. This dataset, which contains 226,711 news articles and their human-written one-sentence abstractive summaries, is particularly suitable for our evaluation due to its comprehensive coverage and high-quality summaries.

Due to the brevity of the source texts, we can test our results with numerous samples while keeping token usage and computation times relatively low. At the same time, we can still benchmark the quality of our approaches with representative results.

3 Methods

In this section, we will introduce approaches that we will apply for source text summarization. We implemented baseline RAG methods that are not geared toward source text summarization and novel methods that tackle certain weaknesses of the baselines.

3.1 Baselines

We start by introducing our baselines. We outline the intuition for these techniques. Visualizations of the full architectures can be found in [Appendix A].

RAG Pipeline. In the traditional RAG pipeline, we receive a document (source text) that we split into chunks. We obtain a vector representation for each chunk using an embedding model and store text and vector in an index. Once we receive a query, we use the same embedding model to obtain a vector representation for the query. We can compare chunk vectors to the query vector to get the most similar chunk. This chunk is then passed to the LLM along with the query to provide the query with some context to avoid hallucination.

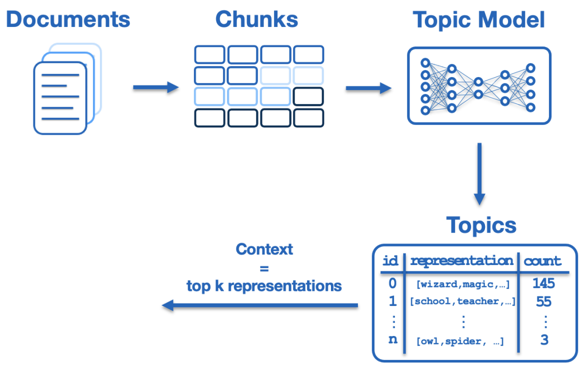
Sentence-Window Retrieval. Sentence-Window retrieval adapts the traditional RAG pipeline. Instead of chunks, we now obtain a vector representation for every sentence. Once we receive the query, we get its vector representation and compare it to each sentence. We then return the most similar sentence and a window of further sentences around the most similar sentences. This is because sentences often contain more fine-grained information than the whole chunk. An entire chunk also contains a lot of noise, making the retrieved chunk less relevant to the query.

Auto-Merging Retrieval. Auto-Merging Retrieval recursively divides the source text into chunks with decreasing sizes. We start by comparing the query to the smallest chunks. We then aggregate the most similar small chunks and use the parent chunk that contains the most similar child chunks. The remaining part of the pipeline remains as it is.

3.2 Novel Methods

All previous approaches have one weakness in common: They are not tailored to summarization tasks. The query "Summarize this source text" will al-ways result in the same embedding vector. The query doesn't provide a meaningful vector, and the chunks provided as a context will likely not improve overall summarization quality because the context is irrelevant.

Topic-based Retrieval. This motivates a new approach that removes the query as the basis for retrieving context. Instead, we will run a topic model to extract a topic for every single chunk in the database. We will then perform a majority vote to obtain the most relevant chunks. Depending on how many chunks we want to pass as context to the LLM, we then obtain the topic summaries for the k most frequent topics. We then use pairwise similarities between the topic summaries and chunks to obtain the most similar chunks. We hope that the obtained chunks contain a diverse set of information that is well-suited to summarize the text as a whole. The architecture is summarized in figure 1:



id	representation	count
0	[wizard,magic,...]	145
1	[school,teacher,...]	55
⋮	⋮	⋮
n	[owl,spider, ...]	3

Figure 1: Obtaining Chunks based on Topic Models

We will use LDA and BERTopic as topic models, which differ slightly. LDA requires us to specify the number of topics we want to obtain, while BERTopic can automatically infer the number of topics. Both topics return a summary that is not a complete sentence but a list of words. We concatenate these words and embed the sequence of words.

Knowledge Graph Retrieval. We introduce another approach that uses knowledge graphs to pass context to the LLM. Knowledge graphs summarize knowledge of a text in a graph. The graph consists of nodes representing entities (words) and edges that describe the relationship between these entities as shown in Figure 2.

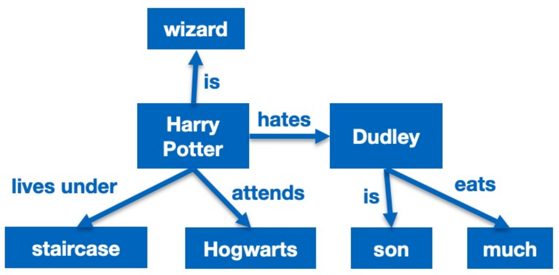


Figure 2: A simplified knowledge graph obtained from the Harry Potter book as a source text. Nodes are entities; edges describe relationships between entities.

Once more, we know that the user's query will not add meaningful information to the context. Therefore, we replace the query-based topic retrieval using a highest-degree heuristic. In a knowledge graph, nodes with high degrees are words in relationships with many other words. Words with a high degree are in a relationship with many words promising to contain the most insightful context for summarization. As a chunk, we concatenate all extracted relationships in a single string that will be passed to the LLM (Figure 3). If more chunks are intended, one can pass the top k highest degree nodes instead of a single one.

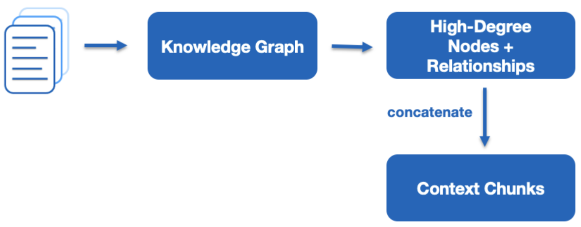


Figure 3: Chunk Creation. Given the knowledge graph, we extract the highest-degree nodes. We concatenate all relationships of these nodes to obtain context chunks.

Extractive Retrieval. Extractive Retrieval synergizes Extractive Summarization and Retrieval Augmented Generation (RAG). This methodology offers a significant advantage by eliminating the traditional need for indexing and searching within such an index. Instead, we summarize the source document based solely on its intrinsic content.

The process starts with Extractive Summarization using a BERT (Bidirectional Encoder Representations from Transformers) model to identify critical sentences in the source text. First, the source text is processed using a BERT Tokenizer to encode the data. Next, we use a K-Nearest Neighbors

(KNN) algorithm on these embeddings, selecting a predefined number of sentences, denoted by 'K', that best represent the text's core ideas. These sentences are chosen for their closeness to the centroid of each cluster formed in the embedding space.

The selected key sentences are then used as the retrieved context in an RAG process, enhancing the language model's ability to generate relevant, contextually rich text (Figure 4). This approach avoids the complexities and overhead caused by indexing, enabling faster and more focused information retrieval.

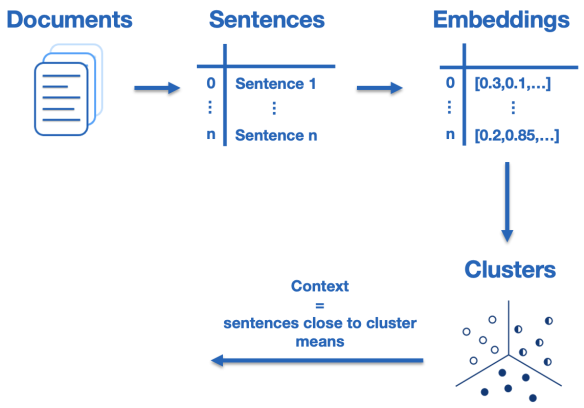


Figure 4: Obtaining chunks using extractive summarization.

Integrating these critical sentences into the language model makes the generation process more efficient and accurate, ensuring the output is highly relevant to the original text's intent.

BART as LLM. In our experiments, we have noticed that the effectiveness of our retrieval approach is consistently tied to the performance of the LLMs we employ. To address this issue, minimize costs for the service provider, and reduce the model size, we intend to conduct experiments using BART. BART is a task-specific model, unlike the larger LLMs we have previously used. One drawback of using BART is its limited context window compared to LLMs.

4 Experiments

In this section, we briefly introduce our experimental setup and the metrics we use to assess the effectiveness of our methods.

4.1 Experimental Setup

As mentioned earlier, we will use XSUM to assess summarization performance. Due to computational and monetary limitations, we will not evaluate the full data set but only 200 randomly selected news articles/samples. Each sample is used to create a new index specific to that sample. We then use a wrapper function to decorate the original document with an appropriate prompt template in figure 14 in [Appendix C] and compare the response to a reference summary from the dataset itself.

4.2 Evaluation Metrics

We utilize two categories of evaluation metrics: those assessing the effectiveness of RAG and those evaluating the quality of the generated summaries. The rationale is to first validate the performance of RAG through its specific metrics, ensuring its reliability before determining its suitability for summarization tasks. This dual approach allows for a comprehensive evaluation of both the method and its application.

4.2.1 RAG Metrics

As described in the methods section, the RAG process can be divided into distinct stages: generating a query, extracting context based on that query, and delivering a context-enhanced response. The RAG triad is an evaluation metric that assesses the effectiveness of RAG by evaluating each of these stages, providing three distinct scores as visualized in figure 5. Context Relevance measures the pertinence of the retrieved context to the query, Groundedness evaluates whether the context is effectively integrated into the response, and Answer Relevance assesses the response's relevance to the original query.

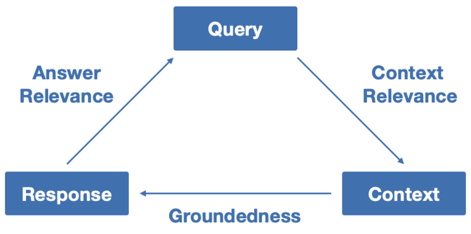


Figure 5: The RAG Triad for performance evaluation

4.2.2 Summarization Metrics

Assessing summarization quality is challenging due to the lack of standardized metrics across the literature, making it difficult to consistently compare different summarization techniques. We utilize a range of metrics to ensure broad compatibility with existing and future research.

The first metric, LDFACTS (Bishop et al., 2024), evaluates the factuality of abstractive summaries and combines the assessment of RAG with the assessment of summarization quality. Additionally, we include BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005), which, although traditionally used in translation tasks, are applicable here due to their focus on meaning preservation. These metrics are well-known and straightforward, making them useful reference points for evaluation.

BERTScore (Zhang et al., 2020) and BARTScore (Yuan et al., 2021) are used to assess the semantic similarity between the generated and reference texts, which is critical for evaluating the quality of summaries. Even though they are not specifically designed for summarization, they effectively gauge how closely a summary aligns with the original content.

Finally, we employ ROUGE (Lin, 2004), a set of metrics that evaluates summarization quality based on n-gram overlaps, word sequences, and word pairs between the generated and reference summaries. This comprehensive approach allows for a detailed assessment of summarization performance.

5 Results

In this section, we present the results of our analysis, focusing on the key findings that emerged from the experiments. These findings highlight the main trends and differences between our approaches.

5.1 RAG Quality

We start by evaluating the performance within the RAG pipeline. Note that the Extractive Retrieval approach is not included in this evaluation, as it does not adhere to the traditional steps of the RAG framework. The results are summarized in Table 1.

Approach	**Context Relev.**	**Ground-edness**	**Answer Relev.**
Standard RAG	0.10	0.99	0.92
Sentence-Window	0.12	0.98	0.91
Auto-Merging	0.18	0.98	0.92
Extractive	N/A	N/A	N/A
Topic-based	0.02	0.61	0.92
Knowledge Graph	0.12	0.57	0.95

Table 1: RAG Triad Performance. Each metric is measured on a scale from 0 (bad) to 1 (good).

Low context relevance across approaches. Our findings indicate that standard approaches outperform our specialized architectures regarding context relevancy, with Knowledge Graph-based retrieval systems excelling over summary-specialized architectures. However, we can observe that context relevance is generally low across approaches. This proves our initial hypothesis that in a summarization setting, *the query does not give rise to a specific chunk* that is relevant for summarization. If this was the case, query and obtained chunk would be very similar, resulting in a high context relevancy, in particular for classical RAG approaches.

Lower groundedness for novel approaches. Traditional summarization methods outperform novel approaches in terms of groundedness. We attribute this to the more coherent and focused nature of full text-chunks compared to the diverse and fragmented context provided for topic-based and graph-based retrieval. Due to the significantly shorter length of summaries relative to their context, it is challenging to comprehensively cover a diverse set of aspects compared to a single coherent chunk. Importantly, this does not reflect negatively on the quality of summarization produced by novel approaches. In fact, a close alignment between query and context can even hinder performance when cnotext relevance is low as previously observed.

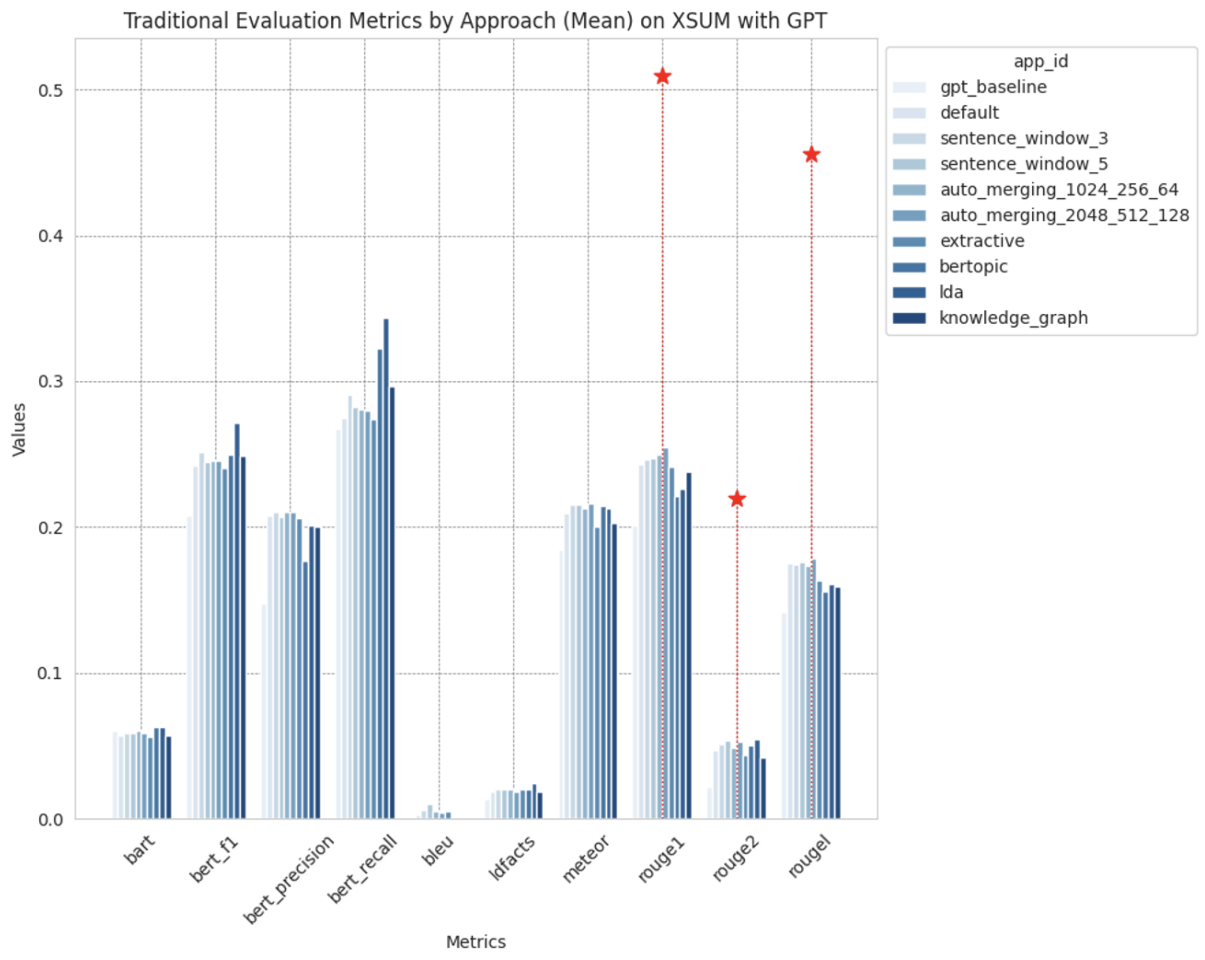


Figure 6: Summarization Metrics for all Approaches. Red star shows SOTA performance.

High answer relevance. All methods demonstrate high performance regarding answer relevance, though the Knowledge Graph approach slightly outperforms the others. This suggests that responses remained closely related to the source text, even when groundedness was lower. Despite the lower context relevance, the Knowledge Graph approach showed the best overall performance, indicating that dense information retrieval can compensate for lower-quality data.

5.2 Summarization Quality

Figure 6 demonstrates the results on summarization quality. All RAG-based approaches, including the unoptimized baselines, significantly outperform the plain GPT model across all metrics. Interestingly, all RAG-based approaches perform similarly, showing no substantial differences among them. The BERT-based metrics indicate that the topic-based approaches and the Knowledge Graph approach achieve the best results, while other scores do not vary significantly. Despite these improvements, the results suggest that we are still far from achieving state-of-the-art performance in summarization metrics. The key takeaway is that adding context through RAG approaches markedly enhances summarization quality compared to the same non-RAG methods.

As shown in [Appendix B], using BART as an LLM leads to significantly worse results across metrics and approaches. This is likely due to the limited context size, which prevents the LLM from incorporating all provided information equally.

5.3 Computation Times

This section discusses the computation times associated with the different approaches as shown in table 2.

Approach	Indexing	Inference
Standard RAG	4.26s	1.12s
Sentence-Window	5.03s	1.24s
Auto-Merging	4.59s	1.41s
Extractive	0.50s	2.26s
Topic-based	4.02s	1.21s
Knowledge Graph	7.52s	1.98s

Table 2: Indexing and Inference Speed

Indexing times are approximately equal for all baseline and topic-based retrieval methods, as they all utilize a vector index. However, building the Knowledge Graph takes approximately 50% longer, which could pose a challenge, especially for larger source texts. Extractive Retrieval stands out for its significantly faster indexing times because it does not store data and instead embeds it on the go during retrieval. Conversely, inference times are the longest for Extractive Retrieval due to the combined embedding and nearest neighbor classification processes during inference.

5.4 Ablation Study

To ensure our RAG-based summarization pipeline's performance is not limited to specific datasets, we conducted an evaluation using the Multi-News dataset. The resulting score distributions did not exhibit significant differences or noticeable shifts compared to the XSUM dataset, as shown in figure 12 in [Appendix B]. Although minor fluctuations were present, they can likely be attributed to the small sample size used in this ablation study. These results suggest that our pipeline maintains consistent performance across different downstream datasets, demonstrating its general applicability and robustness.

5.5 Limitations

In our project, we encountered several limitations that influenced the effectiveness of our methodologies. One significant issue was with topic-based retrieval using BERTopic, which often collapsed multiple topics into a single summary. This was largely due to the model's pre-training, which caused it to merge subtopics within the same domain, resulting in a loss of detailed information.

Another challenge arose with Latent Dirichlet Allocation (LDA), which required perfectly preprocessed text to function optimally. Inadequate preprocessing, particularly the presence of recurring words like author names, led to these words dominating the topic distribution, thereby distorting the intended focus of the analysis.

The nature of our source material, especially short news articles in the XSUM dataset, also presented difficulties. Information in these articles is typically spread across the entire text, and our approach of extracting specific chunks risked highlighting some aspects while neglecting others. Although our method showed promise with longer articles, it struggled to balance the representation of all content in shorter pieces.

Finally, the capabilities of large language models (LLMs) significantly impacted summarization quality. Due to financial and hardware constraints, we could not utilize state-of-the-art models or APIs, which limited the performance of our system. Additionally, the need to pass complete texts to the LLM to avoid hallucination increased the complexity and length of the inputs, further challenging the summaries' accuracy.

6 Conclusion and Future Work

This project proposes a modular framework to adapt standard RAG pipelines for source text summarization. Our findings indicate that RAG-based architectures enhance factuality and overall summarization quality compared to standard GPT-models. It is worth noting that our specialized adaptations are able to outperform baseline RAG methods with respect to some metrics but not consistently across summarization metrics. Additionally, RAG-based approaches still fall short of state-of-the-art summarization techniques. A significant factor influencing summarization quality is LLM choice. Future work could explore whether utilizing the latest and most advanced LLMs, instead of older models like GPT-3.5Turbo, could achieve SOTA performance.

Furthermore, the development of large-context window LLMs, such as LongRoPE (Ding et al., 2024), presents an intriguing avenue for research. Investigating how RAG frameworks perform with these models, which can accommodate entire source texts, could further provide insights into improving summarization quality.

Disclaimer

While we wrote the whole report ourselves, we used Grammarly [1] and ChatGPT [2] to proofread and reformulate the report's initial version. We used Github Copilot[3] to generate documentation for our code.

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*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Jennifer A. Bishop, Sophia Ananiadou, and Qianqian Xie. 2024. LongDocFACTScore: Evaluating the factuality of long document abstractive summarisation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10777–10789, Torino, Italia. ELRA and ICCL.

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. *arXiv preprint arXiv:2402.13753*.

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*.

Matouš Eibich, Shivay Nagpal, and Alexander Fred-Ojala. 2024. Aragog: Advanced rag output grading. *arXiv preprint arXiv:2404.01037*.

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. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

MetaAI. 2024. Introducing meta llama 3: The most capable openly available llm to date. `https://ai.meta.com/blog/meta-llama-3/`. Accessed: 2024-05-22.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Vipula Rawte, Amit Sheth, and Amitava Das. 2023. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Aji Prasetya Wibawa, Fachrul Kurniawan, et al. 2024. A survey of text summarization: Techniques, evaluation and challenges. *Natural Language Processing Journal*, 7:100070.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *CoRR*, abs/2106.11520.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

[1]https://app.grammarly.com/
[2]https://chatgpt.com
[3]https://github.com/features/copilot

A Appendix: Full Baseline Visualizations

A.1 Standard RAG Pipeline

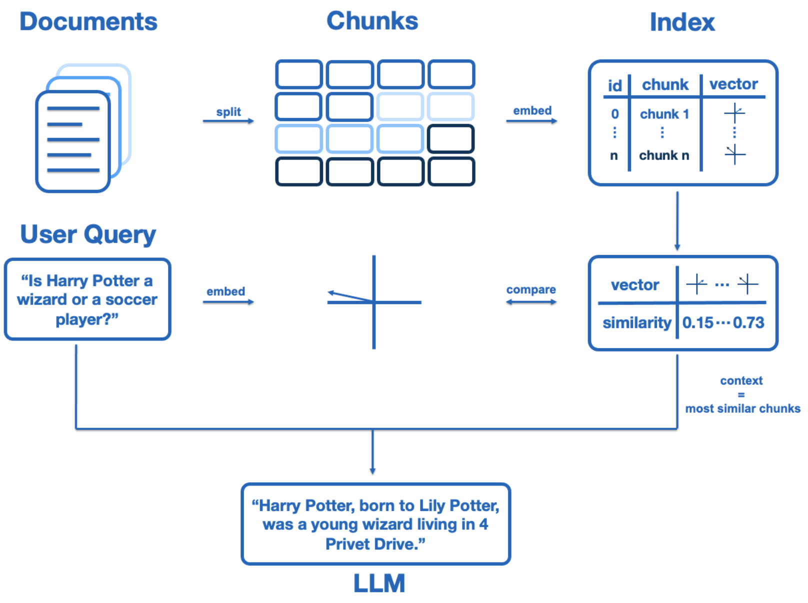

Documents
split
Chunks
embed
Index
id
chunk
vector
0
chunk 1
n
chunk n
User Query
"Is Harry Potter a wizard or a soccer player?"
embed
compare
vector
similarity
0.15···0.73
context = most similar chunks
"Harry Potter, born to Lily Potter, was a young wizard living in 4 Privet Drive."
LLM

Figure 7: The standard RAG pipeline divides documents into chunks and creates an index as a lookup. Once we receive a user query, we compare the query embedding to chunk embeddings in the index. The most similar chunks are passed as a context to the LLM.

A.2 Sentence-Window Retrieval

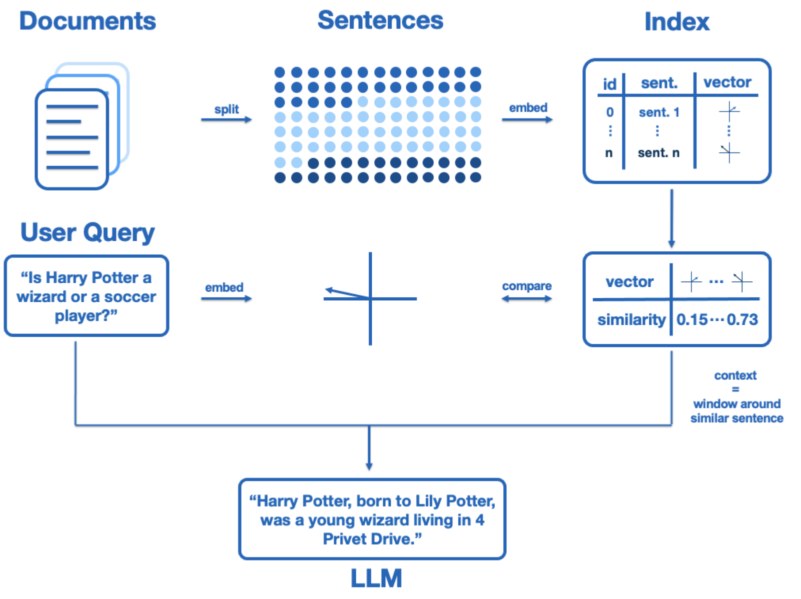

Documents
split
Sentences
embed
Index
id
sent.
vector
0
sent. 1
n
sent. n
User Query
"Is Harry Potter a wizard or a soccer player?"
embed
compare
vector
similarity
0.15···0.73
context = window around similar sentence
"Harry Potter, born to Lily Potter, was a young wizard living in 4 Privet Drive."
LLM

Figure 8: Sentence-window retrieval adapts the standard RAG pipeline by embedding sentences instead of chunks. The query is then compared to the query and the window around the most similar sentence is passed as context to the LLM.

A.3 Auto-Merging Retrieval

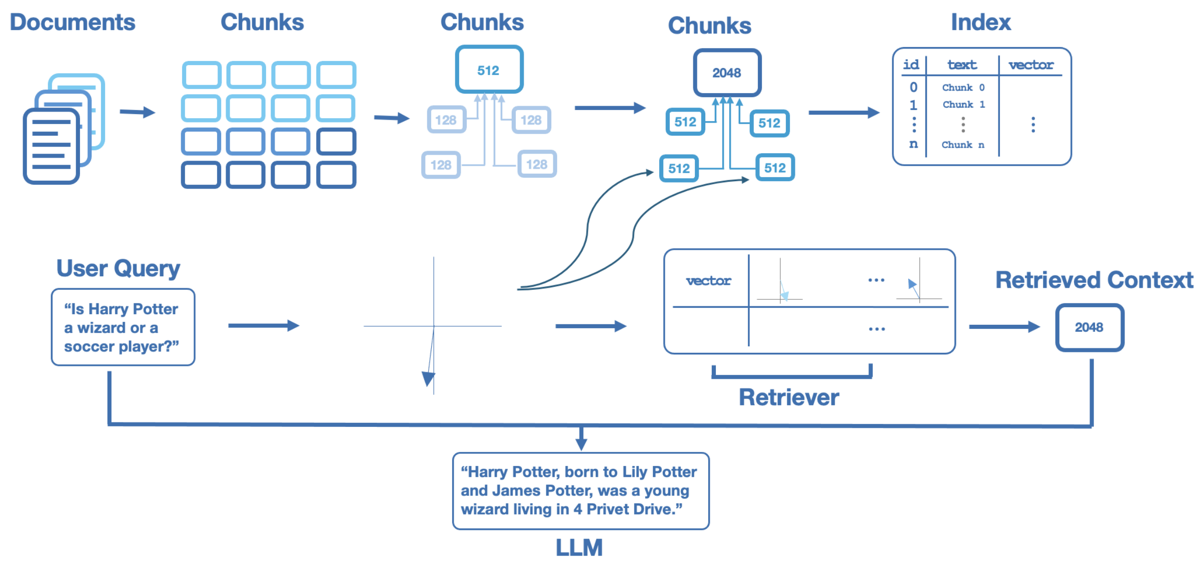


Figure 9: Auto-Merging Retrieval uses a recursive approach to find the most similar chunk. The idea is similar to sentence-window retrieval, i.e., similarity in smaller chunks is more expressive than in large chunks.

B Appendix: Extended Results

B.1 RAG Triad using GPT3.5-Turbo on XSUM

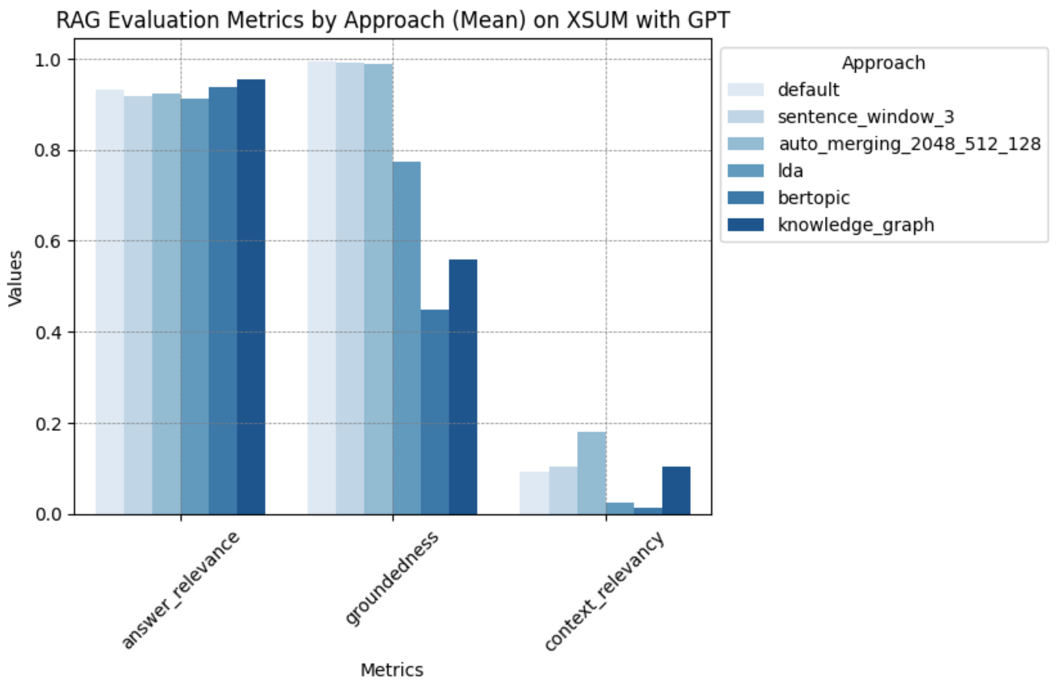


Figure 10: Full Results of RAG Triad with GPT3.5-Turbo as LLM on XSUM.

B.2 RAG Triad using BART on XSUM

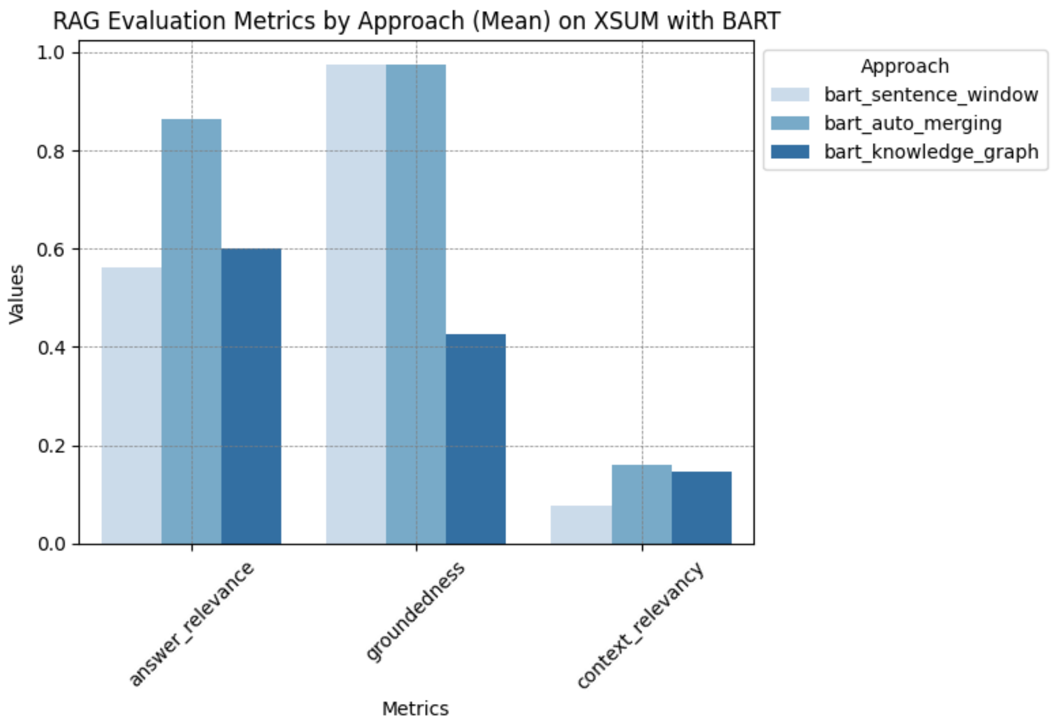


Figure 11: Full Results of RAG Triad with BART as LLM on XSUM.

B.3 RAG Triad using GPT3.5-Turbo on Multi-News

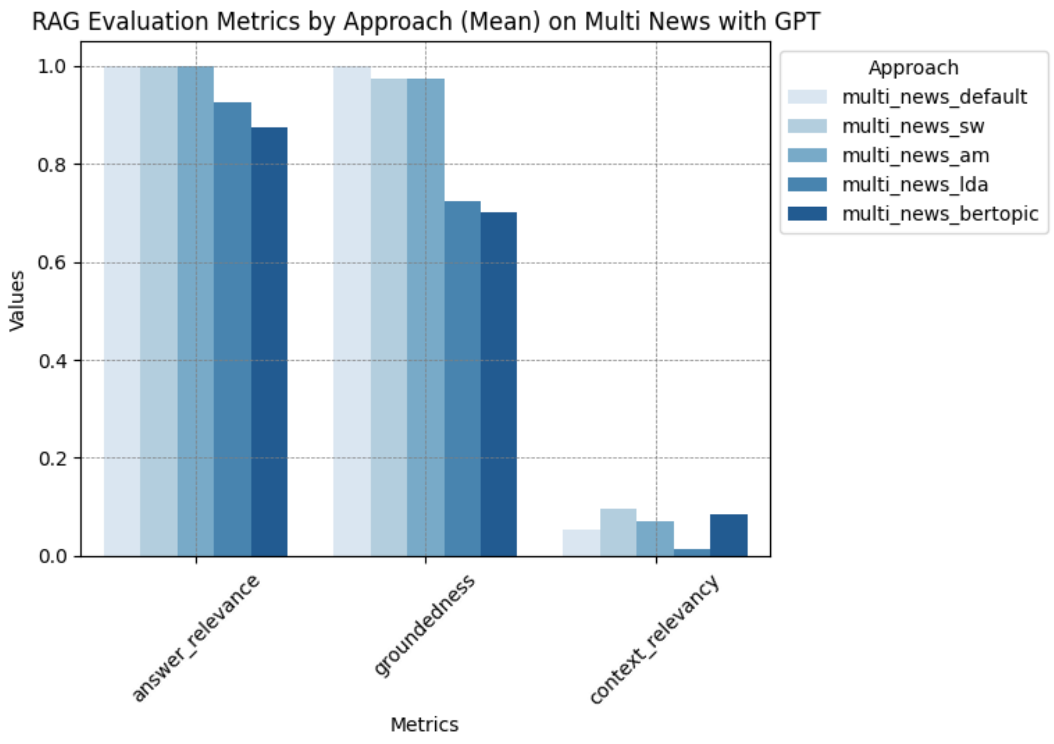


Figure 12: Full Results of RAG Triad with GPT3.5-Turbo as LLM on Multi-News.

B.4 Summarization Metrics using BART

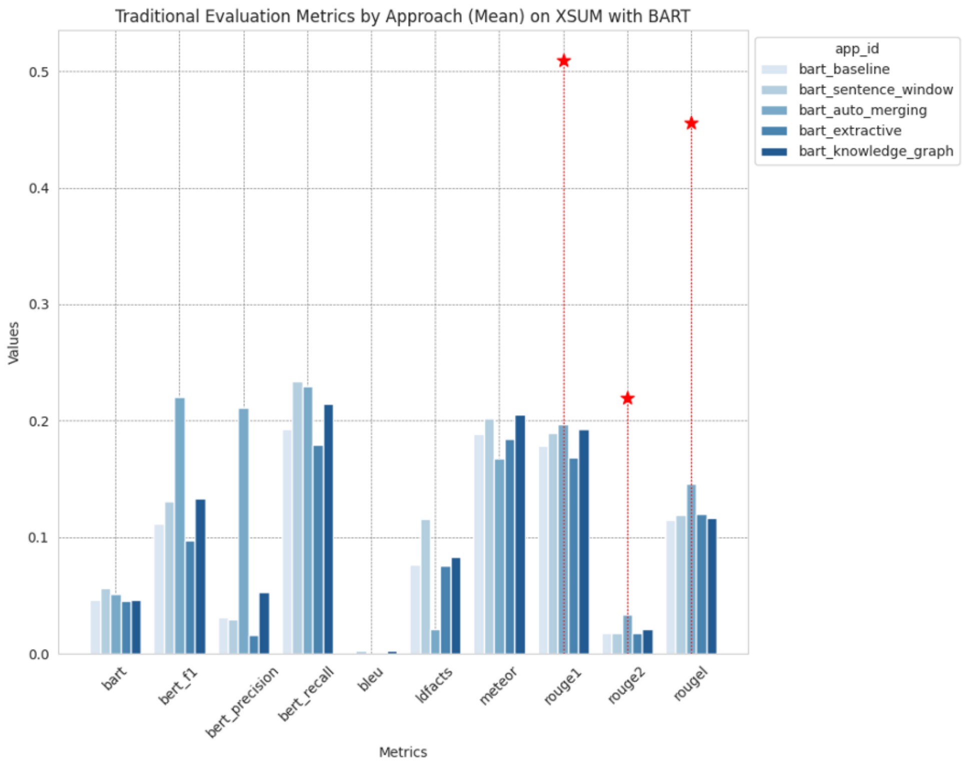


Figure 13: Full Results of summarization metrics with BART as LLM.

C Appendix: Prompt Template

```
"## Summarization Prompt Template\n\n"

"We have provided the original document to be summarized below.\n"
"---------------------\n"
"### Original Document:\n"
"{original_document}\n"
"---------------------\n\n"

"### Instructions:\n"
"Using the above document, create a concise and precise summary. "
"Focus on the key points and essential information, "
"ensuring that the summary is as brief as possible while still capturing the main ideas.\n\n"

"### Requirements:\n"
"1. **Brevity**: Keep the summary IN ONE (EXTREMELY) SHORT SENTENCE and to the point.\n"
"2. **Clarity**: Ensure the summary is clear and easy to understand.\n"
"3. **Coverage**: Include all critical information from the original document.\n"
"4. **Relevance**: Exclude any extraneous details.\n\n"

"---------------------\n"
"### Example Document:\n"
"Media playback is unsupported on your device 8 May 2015 Last updated at 10:28 BST During the war, "
"families would have to ration their food and had little communication in their homes. Luxuries "
"like chocolate and fruit were very difficult to find and families had to grow their own food to "
"survive. Watch Martin's report to find out more.\n\n"
"### Example Summary:\n"
"Martin went to the German Occupation Museum to see what life was like for a family living on Guernsey in "
"World War II.\n"
"---------------------"
```

Figure 14: The template for summarization prompts