Graph-Augmented Open-Domain Multi-Document Summarization

Xiaoping SHEN
Hong Kong University of Science and Technology
xshenat@connect.ust.hk

Yekun Chai*
Baidu
chaiyekun@gmail.com

Abstract

In the open-domain multi-document summarization (ODMDS) task, retrieving relevant documents from large repositories and generating coherent summaries are crucial. However, existing methods often treat retrieval and summarization as separate tasks, neglecting the relationships among documents. To address these limitations, we propose an integrated retrieval-summarization framework that captures global document relationships through graph-based clustering, guiding the re-ranking of retrieved documents. This cluster-level thematic information is then used to guide large language models (LLMs) in refining the retrieved documents and generating more accurate, coherent summaries. Experimental results on the ODSUM benchmark demonstrate that our method significantly improves retrieval accuracy and produces summaries that surpass those derived from the oracle documents. These findings highlight the potential of our framework to improve both retrieval and summarization tasks in ODMDS.

1 Introduction

Traditional multi-document summarization (MDS) tasks typically involve a small, fixed set of documents, and the summarizer does not need to verify the relevance of them to the query. However, in ODMDS (Ji et al., 2013), the scale of the document repository is immense, such as the entirety of Wikipedia. This vastness makes it impractical to use the entire Wikipedia as the input set or to directly locate all documents relevant to a given query. To address this, a "retrieve-then-summarize" architecture has been proposed (Xu and Lapata, 2020; Giorgi et al., 2023) as illustrated in Figure 1.

During the retrieval phase, identifying a truly relevant set of documents from an extensive repository based on a brief query is challenging. Traditional retrieval methods, such as BM25 (Robertson and Zaragoza, 2009) and text embeddings, calculate similarity scores between the query and each document for retrieval. These methods treat documents as discrete, independent entities, ignoring the interconnections between them and failing to adopt a global perspective of the entire repository. Additionally, during the summarization phase, the most popular tools are LLMs, which possess strong semantic understanding and text generation capabilities, making them well-suited for summarization tasks (Ouyang et al., 2022). However, due to the input length limitations of LLMs, even if a highly relevant set of documents is retrieved, it is challenging to input all of them (Yang et al., 2023a).

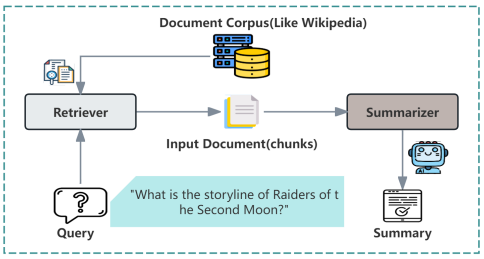


Figure 1: Retrieve-then-summarize pipeline.

Previous research has largely focused on enhancing the accuracy of retrieval systems (Karpukhin et al., 2020; Izacard et al., 2022) and improving the ability of summarization systems to distill refined summaries from verbose texts (Pasunuru et al., 2021; Yasunaga et al., 2017). However, these efforts were not specifically designed for the ODMDS task and did not integrate retrieval and summarization into a unified framework.

To address these gaps, we propose a retrieval-summarization framework based on a key assumption: "*In ODMDS tasks, documents with the same topic or contextual relationships should be considered in parallel.*" This assumption is grounded in the observation that documents relevant to a query are typically clustered around a few specific topics rather than dispersed across unrelated topics.

*Corresponding author.

Proceedings of the 31st International Conference on Computational Linguistics: Industry Track, pages 318–330
January 19–24, 2025.

Contribution The main contributions of this work are as follows:

- We proposed a retriever that builds graphs to capture connections between documents, clusters documents based on context and topic, and uses this clustering and topic information to guide re-ranking and subsequent reflection and refinement modules of the summarizer.
- We proposed a summarizer that can accept a large number of candidate documents and refine them to varying degrees based on the cluster information output by the retriever, allowing the final summary generation to focus on texts that are more relevant to the issue.
- We conducted comprehensive experiments and ablation studies, exploring the performance improvements in the retriever and summarizer, demonstrating its superior performance compared to the baseline model.

2 Related Work

2.1 Open-Domain MDS

Several attempts have addressed the challenges of ODMDS. Giorgi et al. (2023) proposed a two-stage process: document retrieval followed by summarization. They built their index from four datasets and used pseudo-queries (summaries as queries). However, this led to less targeted summaries and retrieval of only 2.7 relevant documents out of ten on average. Liu* et al. (2018) introduced the WikiSum dataset, generating Wikipedia sections from titles and reference documents using a mix of extractive and abstractive techniques, though it was constrained by a small index compared to open domain. Zhang et al. (2023b) used a pretrained dense passage retriever and T5 summarizer, testing on a proprietary dataset. Lastly, Zhou et al. (2023) created the ODSUM benchmark, converting summarization datasets into ODMDS formats, emphasizing evaluation improvements to enhance retrieval performance and robustness.

2.2 LLMs in Retrieval and Summarization

LLMs have excelled in zero and few-shot learning, outperforming traditional retrieval methods like BM25 and self-supervised models like Contriever (Brown et al., 2020; Chowdhery et al., 2022; Izacard et al., 2021). Their strength in low-supervision scenarios is well-documented (Schick and Schütze, 2020; Winata et al., 2021; Bonifacio et al., 2022). In summarization, LLMs, when guided by task descriptions, produce more accurate summaries, addressing common issues like factual inaccuracies (Goyal et al., 2023; Zhang et al., 2023a; Yang et al., 2023b; Zhao et al., 2023). These models also excel in automatic evaluation (Shen et al., 2023; Mao et al., 2023; Liu et al., 2023b). Given their effectiveness, LLMs are key to the entire ODMDS pipeline, as demonstrated in our method.

3 Methods

3.1 Retriever

Our retriever is designed to perform retrieval from a graph-based perspective, considering the interrelationships between chunks. This involves constructing a graph structure, conducting GAE training, and partitioning chunk clusters. This process guides an adaptive rerank of the retrieved chunks, resulting in a chunk list that provides more accurate contextual information for the summarizer.

3.1.1 Graph Construction

Given a series of documents, we first split them into chunks that comply with the input length limitations of the text embedding model. A long document might be divided into several text chunks, with edges established between neighboring chunks to indicate contextual relationships. Additionally, edges can be based on citation relationships between documents if such properties exist, like in Wikipedia or academic paper repositories.

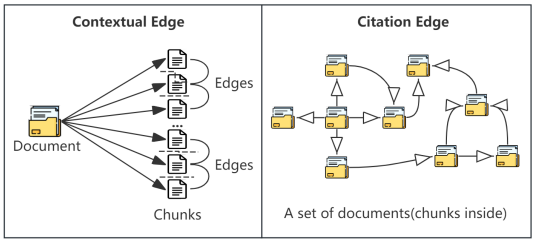


Figure 3: Illustration of edge construction.

In this graph, each node represents a text chunk, and edges indicate contextual or citation connections. Node features are constructed as follows: Firstly, we perform TF-IDF mapping for each text chunk to obtain the embedding vector $v_{\text{TF-IDF}}$. Secondly, use a text embedding model[1] to obtain the embedding vector v_{gpt}. Then apply Principal Component Analysis (PCA) (Hotelling, 1933) to reduce both vectors to the same dimension d:

$$\mathbf{v}^{d}_{\text{TF-IDF}} = \text{PCA}\left(\mathbf{v}_{\text{TF-IDF}}, d\right) \quad (1)$$

$$\mathbf{v}^{d}_{\text{GPT}} = \text{PCA}\left(\mathbf{v}_{\text{GPT}}, d\right) \quad (2)$$

[1] https://platform.openai.com/docs/guides/embeddings

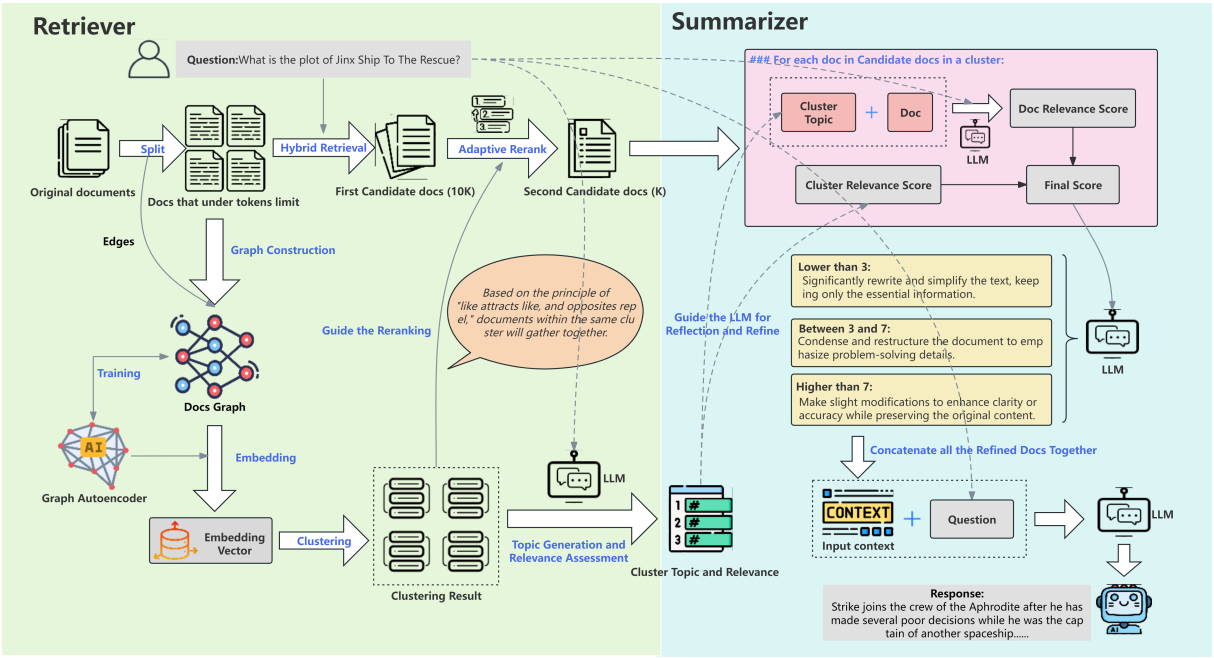


Figure 2: Illustration of proposed method.

Finally, perform a weighted sum of the two vectors to obtain the final feature vector:

$$\mathbf{v}_{\text{final}} = \alpha \cdot \mathbf{v}^d_{\text{TF-IDF}} + \beta \cdot \mathbf{v}^d_{\text{GPT}} \tag{3}$$

Through these steps, feature vectors for each text chunk are constructed, forming a graph model for the entire document collection.

3.1.2 Graph Embedding and Clustering

In the graph constructed through preprocessing, each node represents a text chunk, and edges represent their contextual relationships or citation links. To extract deep connections of structure and content from these text chunks, we used an unsupervised learning approach with a Graph Autoencoder (GAE) (Kipf and Welling, 2016) for effective node embedding. Then we cluster these embedding vectors to get the cluster division of the chunks set.

GAE Architecture and Training We employed a Graph Convolutional Network (GCN) (Kipf and Welling, 2017) as the core encoder in our GAE. The GAE operates in two primary phases: encoding and decoding, working together to learn meaningful node embeddings in an unsupervised manner by reconstructing the graph structure.

In the encoding phase, the transformation from $\mathbf{X}$ to $\mathbf{Z}$ is performed. $\mathbf{X} \in \mathbb{R}^{N \times F}$ denotes the initial feature matrix, where N is the number of nodes, and F is the dimensionality of the input features. Along with the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, the input $\mathbf{X}$ is passed through multiple GCN layers sequentially.

$$\mathbf{Z} = \text{GCN}_L(...\text{GCN}_1(\mathbf{X}, \mathbf{A})) \tag{4}$$

Each GCN layer aggregates information from the local neighborhood of each node via the adjacency matrix $\mathbf{A}$ and updates the node features. The operation of a single GCN layer is defined as:

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathbf{A}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{5}$$

Here, $\mathbf{H}^{(l)}$ represents the node feature matrix at layer l, with $\mathbf{H}^{(0)} = \mathbf{X}$. $\mathbf{W}^{(l)}$ is the learnable weight matrix at layer l, and σ denotes the ReLU.

The encoding process outputs $\mathbf{Z}$, a matrix where each row corresponds to a node's latent embedding. These embeddings integrate both the intrinsic features of nodes and the structural relationships captured by the graph. This rich representation is subsequently used for clustering to reveal the underlying structure of the text chunks.

In the decoding phase, the reconstructed adjacency matrix $\hat{\mathbf{A}}$ is predicted by computing the inner product of the learned node embeddings:

$$\hat{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^\top) \tag{6}$$

where σ denotes the sigmoid activation function. This prediction strategy is standard for edge prediction tasks, where the inner product captures the similarity between node embeddings in the latent space, reflecting the likelihood of edges.

The graph autoencoder is trained by minimizing the reconstruction loss of the adjacency matrix $\hat{\mathbf{A}}$. This involves optimizing the model parameters to reduce the difference between the predicted adjacency matrix $\hat{\mathbf{A}}$ and the actual adjacency matrix $\mathbf{A}$, using the following loss function:

$$\mathcal{L} = \|\mathbf{A} - \hat{\mathbf{A}}\|_F^2 \tag{7}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. During training, the node embeddings $\mathbf{Z}$ are computed through forward propagation in the encoder, and

the adjacency matrix is reconstructed in the decoder. The reconstruction loss is calculated, and the parameters are updated via backpropagation. This process results in node representations $\mathbf{Z}$ that effectively capture both structural and attribute information of the nodes and their neighborhoods.

Clustering Analysis After training, we use the trained encoder of GAE to obtain the embedding representation for each node. Then, we apply traditional clustering algorithms, such as K-Means(MacQueen, 1967) and DBSCAN(Ester et al., 1996), to cluster the nodes. Through clustering, we obtain the division of chunk clusters, where each cluster contains document chunks with high semantic and thematic similarity.

3.1.3 Re-ranking Based on Clusters

Firstly, we use a hybrid retrieval method to determine an initial set of candidate chunks. Specifically, we calculate the BM25 score and the GPT-Embedding cosine similarity score between the question and each chunk, then take a weighted sum of the two to obtain the final similarity score. Assuming the number of chunks we ultimately input to the summarizer is K, we select the top $10K$ chunks with the highest scores as the candidate set.

Next, we use the cluster information obtained earlier to re-rank these candidate chunks. Assuming the score of chunk i under hybrid retrieval is S_i, the re-ranking score R_i for each chunk can be calculated using the following formula:

$$R_i = S_i + \sum_{j \in W_i} (S_j \times P_j \times F_j) \tag{8}$$

where W_i represents the set of chunks in the cluster containing document i, and S_j, P_j, and F_j represent the original score, position factor, and weight factor of chunk j, respectively. If W_i is empty, the reranking score R_i equals S_i. The position factor P_j considers the order of documents in the initial retrieval results, giving higher-ranked documents more influence:

$$P_j = \frac{1}{\log(1 + \text{rank}_j)} \tag{9}$$

where rank_j is the rank of chunk j in the initial retrieval results. The weight factor F_j adjusts the contribution of each chunk to the reranking score:

$$F_j = \frac{S_j}{\sum_{k \in W_i} S_k} \tag{10}$$

In this method, cluster guides the re-ranking of the initial set of retrieved chunks provided by the retrieval system. Figure 4 illustrates this process. The re-ranking is based on the principle of "*like attracts like, unlike repels*", which causes documents within the same cluster to gather together and elevates the ranking of more important documents within each cluster. Consequently, the re-ranked document list becomes cluster-dominated, reflecting group relationships, which is distinctly different from the independent and discrete relationships before re-ranking. By considering both the initial relevance scores and the relationships within clusters, this approach enhances the chunk set's alignment with the query's intent, providing higher quality input to the summarizer.

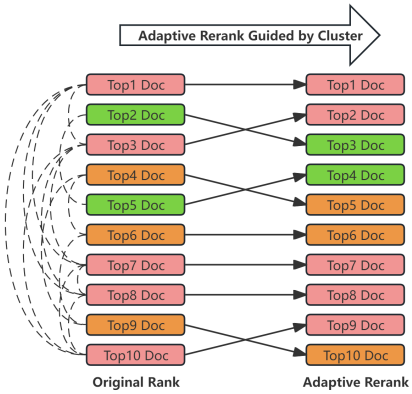


Figure 4: Adaptive Rerank Guided by Cluster.

3.2 Summarizer

After obtaining the output chunks from the retriever, it may not be reasonable to directly concatenate these chunks and pass them into the LLM to generate a summary, as these chunks contain a lot of irrelevant information. Therefore, we designed a summarizer where the LLM reflects on both the topic of the cluster and the content of the chunk themselves, assessing the relevance of chunk. Then, it performs a refinement by selectively extracting and condensing the chunk based on relevance. Through this preprocessing step, we ensure that the LLM receives more precise and concise context, enabling the generation of higher-quality summaries.

3.2.1 Topic and Relevance of Cluster

After obtaining the clusters of the retrieved chunks, we carefully designed a prompt, shown in Figure 9 in Appendix B, to guide the LLM to generate a topic for each cluster and a relevance score between the cluster and the query, to guide the subsequent reflection and refinement module. The purpose of this step is to capture the topic information at the cluster level and obtain the overall relevance of the cluster to the query from a more macro perspective.

3.2.2 Document Reflection and Refinement

After obtaining the topic and relevance scores of the clusters, we further utilize LLM to reflect and

refine each chunk. The reflection process first generates a relevance score for each candidate chunk. Next, we weight and sum the chunk relevance score with the cluster's topic relevance score to obtain the final score. We rewrite the chunks to varying degrees based on this score to achieve length refinement. The prompt is shown in Figure 10. The text generated by the reflection and refinement module not only meets the input length requirements but is also highly relevant and information-dense. These refined chunks are then embedded into carefully designed prompts to generate the final summary, responding to the posed questions with high accuracy and clarity. For more detail in Appendix B.

4 Experiments

This section conducts experimental studies on the effectiveness of our framework's retriever and summarizer. We tested its performance on the ODSUM. See Appendix C for specific experimental setup.

4.1 Dataset

The ODSUM(Zhou et al., 2023) dataset is a benchmark dataset designed specifically for the ODMDS task. It consists of two sub-datasets: ODSum-Story and ODSum-Meeting. The statistics of this dataset are shown in Table 1.

Dataset	ODSum-Story	ODSum-Meeting
Document Number	1,190	232
Avg Doc Length	808.54	7176.21
Queries	635	436
Avg Query Length	10.79	32.89
Avg Related Docs	9.37	2.97
Reference Summary	4	1
Avg Summary Length	273.80	185.17

Table 1: Dataset statistics.

ODSum-Story is adapted from the SQuALITY(Wang et al., 2022) dataset, comprising 127 stories split into 1190 chapters (documents), with an average of 808 tokens per document. It includes 635 summary questions, each requiring about 9.37 documents for context. ODSum-Meeting, based on the QMSum(Zhong et al., 2021) dataset, includes 232 meeting transcripts, averaging 7176 tokens per document. It contains 436 questions, each needing about 2.97 documents for context.

4.2 Baselines and Metrics

Retriever In our experiments, we explored different numbers of retrieval chunks, denoted as K (for more details see Appendix 5.4). Our baseline retrieval algorithms include BM25 and GPT-Embedding. Additionally, we tested a hybrid retrieval method that combines BM25 scores with embedding similarity scores in a weighted manner. This method leverages the advantages of BM25 in word frequency analysis and the strengths of text embeddings in semantic understanding. To evaluate the performance of our retrieval methods, we used three key metrics: Precision at K (P@K), Recall at K (R@K) and F1 score at K (F1@K).

Method	ODSum-Story			ODSum-Meeting		
	P@3	**R@3**	**F1@3**	**P@1**	**R@1**	**F1@1**
BM25	71.17	28.16	40.35	30.61	16.52	21.46
GPT Embedding	65.62	26.75	38.00	22.25	7.81	11.56
Hybrid Retrieval	72.18	28.63	41.00	28.12	15.30	19.82
Our Retriever	**74.04**	**29.98**	**42.68**	**31.24**	**16.78**	**21.83**
	P@8	**R@8**	**F1@8**	**P@3**	**R@3**	**F1@3**
BM25	50.62	48.48	49.53	22.32	33.12	26.76
GPT Embedding	42.62	42.24	42.43	17.03	17.26	17.14
Hybrid Retrieval	52.64	50.30	51.44	20.52	31.09	24.72
Our Retriever	**55.95**	**52.66**	**54.25**	**24.73**	**35.29**	**29.08**
	P@10	**R@10**	**F1@10**	**P@6**	**R@6**	**F1@6**
BM25	44.85	52.53	48.39	17.22	46.28	25.10
GPT Embedding	36.93	44.74	40.46	13.59	26.27	17.91
Hybrid Retrieval	45.23	53.13	48.86	16.94	45.84	24.74
Our Retriever	**48.92**	**56.74**	**52.54**	**18.45**	**50.84**	**27.07**

Table 2: Retrieval performance comparison. The best results are bolded, and the second-best are underlined.

Summarizer In the experiment, we used several LLMs as baseline summarizers. The output chunks from each baseline retriever were incorporated into carefully designed prompts and fed into LLMs to generate summaries. Additionally, we conducted experiments with oracle documents, where the truly relevant documents for each question were provided as inputs to the LLM to show the upper limit of summarizer performance in a perfect retrieval scenario. We adopted three different evaluation metrics: ROUGE(Lin, 2004) measures word overlap between candidate and reference summaries, while BERTScore(Zhang* et al., 2020) uses contextual word embeddings to assess similarity between them and G-Eval(Liu et al., 2023a), a framework that uses LLMs to assess summary quality. For more details on metrics, see Appendix D.

5 Results and Analysis

5.1 Retriever Evaluation

In our experiments, we evaluated the performance of our proposed retriever against several baseline methods on the ODSUM benchmark. The result are shown in Table 2. It indicate that for all top-k selections in both datasets, our proposed retriever consistently outperforms baseline retrieval methods. By leveraging the contextual relationships and topic clustering of documents, our retriever achieves the highest precision, recall, and F1 scores.

Additionally, we found that on Story, the performance of hybrid retrieval is the best among the baselines, but on the Meeting, it is surpassed by BM25. This is because the average document length in the Meeting is much longer than in the Story. Text embedding struggles with capturing information from long documents, while BM25 does not have this issue. Therefore, this further illustrates that our retriever is robust and applicable for retrieving both long and short documents.

5.2 Summarizer Evaluation

Table 3 shows the summarization results from different retrievers, revealing the following insights:

Oracle documents do not guarantee the best. Even if the retrieved documents are perfect, some content has to be discarded due to the input length constraints of the summarizer. As a result, the summarizer cannot access the complete reference text, leading to a decline in output.

G-Eval better reflects summarization quality. Summaries generated by gpt-4.0 scored lower on R-2 and BERTScore compared to llama-3-70b and gpt-3.5, contrary to expectations. Because these metrics evaluate summary quality based on lexical overlap and textual similarity between the generated and reference summaries. In contrast, G-Eval directly assesses the consistency and relevance of the generated summary to the input context, without relying on a reference summary. This makes it more suitable for the ODMDS task, where the key concern is whether the generated summary accurately captures the content of the input context rather than its similarity to a reference summary.

Our method outperforms baselines methods. Although our retriever is not as accurate as Oracle documents, it achieved the best results across all LLMs. This is because our framework captures relationships between documents by constructing graphs and improves the retriever's recall by leveraging cluster information. Additionally, our summarizer rewrites each text chunk, retaining essential information and eliminating unnecessary details, which significantly reduces the input length. This allows us to provide more reference chunks to the LLM within the input length constraints, compensating for the lack of Oracle documents.

5.3 Ablation Study

To validate the role of various modules in our proposed retrieval-summarization framework, we conducted ablation experiments on several key components. These included the **(1) graph construction** and clustering module (Section 3.1.1, 3.1.2) and **(2)** Cluster-based **adaptive re-ranking** module (Section 3.1.3) in the retriever, as well as the **(3)** Cluster-guided **reflection and refinement** module (Section 3.2.1, 3.2.2) in the summarizer. Each of them was individually removed for the experiment. We used gpt-3.5-turbo and gpt-4o for generation, with G-Eval as the evaluation metric. The experimental results for the ODSum-Story and Meeting datasets are shown in Figure 5 and 13.

The results show that removing any module reduced performance compared to the full framework, though still outperforming the baseline. Notably, removing the cluster-guided reflection and refinement module from the summarizer caused the largest drop, indicating its critical role. This suggests the retriever's impact on generation is less significant than the summarizer's. The retriever primarily improves recall by including all relevant chunks, while incorrect chunks matter less, as the summarizer's refinement module filters them out. This ensures that even with some incorrect retrievals, the summary quality remains high.

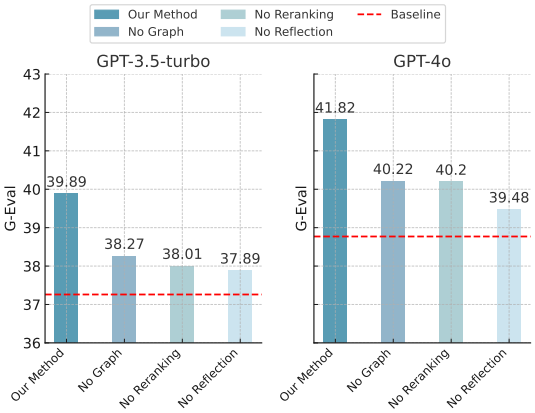


Figure 5: Ablation experiment on ODSum-Story.

5.4 Impact of Document Retrieval Quantity

In our retrieval experiments, we found that increasing the number of retrieved documents K results in higher recall but lower precision. Therefore, this section aims to explore how the quality of the final summary changes with the increase in the number of retrieved chunks K in our method. We conducted experiments on the ODSum dataset, and the results are shown in the Figure 6.

From the Table 1, we can see that the average number of relevant documents for the ODSum-Story and Meeting datasets are 9.37 and 2.97, respectively. The Figure 6 shows that the optimal

ODSUM-Story	LLAMA3-70B			GPT-3.5-Turbo			GPT-4.0-Turbo			GPT-4o		
	R-2	BS	G-Eval	R-2	BS	G-Eval	R-2	BS	G-Eval	R-2	BS	G-Eval
Oracle	**9.77**	84.78	37.91	**10.58**	84.85	39.23	**9.66**	84.37	**40.73**	**10.12**	85.02	41.21
BM-25	9.43	84.71	32.37	9.33	84.49	36.05	8.42	84.21	37.19	8.26	84.91	36.79
GPT-Embedding	9.27	85.01	34.46	8.58	84.62	36.33	7.31	84.47	38.91	7.99	84.25	37.62
Hybrid-Retrieval	9.69	**85.16**	35.81	9.42	84.73	37.26	9.33	**84.59**	39.07	9.07	**85.32**	38.77
Ours	9.63	84.77	**37.96**	9.40	**84.78**	**39.89**	8.96	84.19	40.66	8.74	84.22	**41.82**
ODSUM-Meeting	**LLAMA3-70B**			**GPT-3.5-Turbo**			**GPT-4.0-Turbo**			**GPT-4o**		
	R-2	BS	G-Eval	R-2	BS	G-Eval	R-2	BS	G-Eval	R-2	BS	G-Eval
Oracle	**8.34**	**85.15**	33.76	**12.13**	86.16	35.32	**10.73**	85.52	36.38	**10.65**	85.11	36.44
BM-25	7.98	84.16	29.35	9.42	85.91	35.05	10.64	84.19	35.67	9.32	84.10	35.57
GPT-Embedding	8.14	84.83	31.92	10.23	85.23	35.23	9.48	84.53	35.22	9.43	84.62	35.42
Hybird-Retrieval	8.26	84.75	32.34	10.17	**86.35**	34.91	10.33	84.71	35.86	10.16	**85.37**	35.79
Ours	8.09	85.12	**34.23**	11.96	85.69	**36.57**	10.27	**85.55**	**36.98**	9.33	84.97	**37.02**

Table 3: Summarization performance on ODSum-Story and Meeting. R-2 and BS means ROUGE-2 and BERTScore. The best results under the three indicators are bolded, and the second best result under the G-Eval is underlined.

K values are 20 and 6, respectively, which are exactly twice the actual number of relevant documents. This indicates that our framework has fault tolerance and can accept more input documents to capture the truly relevant ones while ignoring the irrelevant ones.

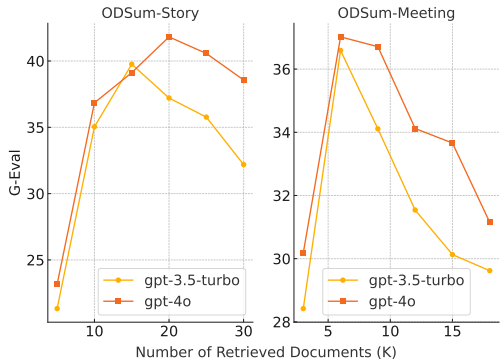


Figure 6: Experiments on the effect of the number of retrieved documents on summary generation.

This finding is also reflected in the two figures. The four curves rise rapidly and steeply before reaching the optimal effect, but the decline after the optimal effect is very gentle. This also confirms that our model tends to find truly relevant documents from a larger set of candidate documents. If the set of candidate documents is too small, it may lead to hallucination phenomena in the LLMs due to the lack of background text.

5.5 Human Evaluation

We randomly selected 50 summary questions from both the ODSUM-Stoty and Meeting datasets for human evaluation by three human volunteers. Our method was compared against baseline approaches, which consisted of a hybrid retrieval method paired with a native LLM summarizer. Both methods employed the same underlying LLM to ensure a fair comparison. Figure 7 shows that our method achieves an win rate of 71% on average. For a detailed evaluation criteria, refer to Appendix F.

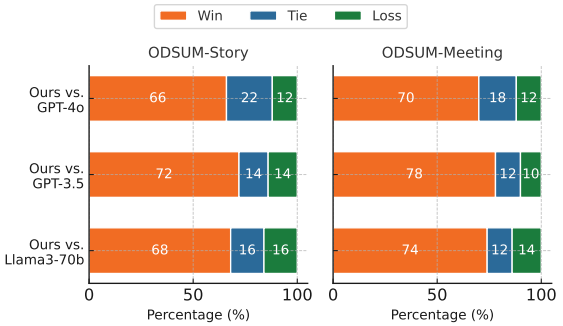


Figure 7: Human pairwise evaluation with GPT-4o, GPT-3.5, and Llama3-70B.

6 Conclusion

This paper tackles the challenge of retrieving relevant documents from large repositories and generating summaries from lengthy inputs in the ODMDS task. We propose an integrated retrieval-summarization framework, recognizing that ODMDS questions typically relate to topic-specific documents rather than isolated ones. Using a graph-based approach, we capture relationships between documents, enabling effective clustering. Our framework connects retrieval and summarization through these clusters, enhancing the retriever's re-ranking and the summarizer's reflection and refinement. Experiments on the ODSUM benchmark demonstrate that our method outperforms baseline strategies, improving retrieval accuracy and summary quality, even surpassing summaries from perfectly retrieved documents.

References

Luiz Henrique Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. Inpars: Unsupervised dataset generation for information retrieval. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press.

John Giorgi, Luca Soldaini, Bo Wang, Gary Bader, Kyle Lo, Lucy Wang, and Arman Cohan. 2023. Open domain multi-document summarization: A comprehensive study of model brittleness under retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8177–8199, Singapore. Association for Computational Linguistics.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2023. News summarization and evaluation in the era of gpt-3. *Preprint*, arXiv:2209.12356.

Harold Hotelling. 1933. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Preprint*, arXiv:2112.09118.

Heng Ji, Benoit Favre, Wen-Pin Lin, Daniel Gillick, Dilek Z. Hakkani-Tür, and Ralph Grishman. 2013. Open-domain multi-document summarization via information extraction: Challenges and prospects. In *Multi-source, Multilingual Information Extraction and Summarization*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Thomas Kipf and Max Welling. 2016. Variational graph auto-encoders. *ArXiv*, abs/1611.07308.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

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.

Peter J. Liu*, Mohammad Saleh*, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023a. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Yixin Liu, Alexander R. Fabbri, Pengfei Liu, Dragomir R. Radev, and Arman Cohan. 2023b. On learning to summarize with large language models as references. *ArXiv*, abs/2305.14239.

J. MacQueen. 1967. Some methods for classification and analysis of multivariate observations.

Rui Mao, Guanyi Chen, Xulang Zhang, Frank Guerin, and E. Cambria. 2023. Gpteval: A survey on assessments of chatgpt and gpt-4. *ArXiv*, abs/2308.12488.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Ramakanth Pasunuru, Mengwen Liu, Mohit Bansal, Sujith Ravi, and Markus Dreyer. 2021. Efficiently summarizing text and graph encodings of multi-document clusters. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4768–4779, Online. Association for Computational Linguistics.

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

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *ArXiv*, abs/2009.07118.

Chenhui Shen, Liying Cheng, Yang You, and Lidong Bing. 2023. Large language models are not yet human-level evaluators for abstractive summarization. In *Conference on Empirical Methods in Natural Language Processing*.

Alex Wang, Richard Yuanzhe Pang, Angelica Chen, Jason Phang, and Samuel R. Bowman. 2022. SQuALITY: Building a long-document summarization dataset the hard way. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1139–1156, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Genta Indra Winata, Andrea Madotto, Zhaojiang Lin, Rosanne Liu, Jason Yosinski, and Pascale Fung. 2021. Language models are few-shot multilingual learners. *ArXiv*, abs/2109.07684.

Yumo Xu and Mirella Lapata. 2020. Coarse-to-fine query focused multi-document summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3632–3645, Online. Association for Computational Linguistics.

Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. 2023a. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *Preprint*, arXiv:2304.13712.

Xianjun Yang, Yan Li, Xinlu Zhang, Haifeng Chen, and Wei Cheng. 2023b. Exploring the limits of chatgpt for query or aspect-based text summarization. *ArXiv*, abs/2302.08081.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462, Vancouver, Canada. Association for Computational Linguistics.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori Hashimoto. 2023a. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57.

Weijia Zhang, Svitlana Vakulenko, Thilina Rajapakse, Yumo Xu, and Evangelos Kanoulas. 2023b. Tackling query-focused summarization as a knowledge-intensive task: A pilot study. *Preprint*, arXiv:2112.07536.

Yilun Zhao, Zhenting Qi, Linyong Nan, Boyu Mi, Yixin Liu, Weijin Zou, Simeng Han, Ruizhe Chen, Xiangru Tang, Yumo Xu, Dragomir Radev, and Arman Cohan. 2023. QTSumm: Query-focused summarization over tabular data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1157–1172, Singapore. Association for Computational Linguistics.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zairi Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir R. Radev. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *North American Chapter of the Association for Computational Linguistics*.

Yijie Zhou, Kejian Shi, Wencai Zhang, Yixin Liu, Yilun Zhao, and Arman Cohan. 2023. Odsum: New benchmarks for open domain multi-document summarization. *Preprint*, arXiv:2309.08960.

A Limitations

Despite the promising performance of our retrieval-summarization framework in the ODMDS task, several limitations remain that present opportunities for future improvement.

Scalability Issues Our method may encounter computational and storage bottlenecks when handling larger-scale document collections. Although graph embeddings and clustering techniques effectively capture inter-document relationships, the computational cost and memory requirements increase significantly with the scale of the document corpus. Moreover, generating high-quality graph embeddings and clustering results becomes more challenging as the number of documents grows. Enhancing the scalability of our method is thus a crucial issue that needs to be addressed.

Consumes Substantial LLM Inference Resources Our framework relies on powerful LLMs such as GPT-3.5 and GPT-4.0, which excel in processing long texts and generating high-quality summaries. Additionally, the framework involves multiple calls to LLMs, such as generating topics and topic relevance scores for each cluster and performing reflection and refinement for each document. Consequently, each query consumes a significant amount of LLM inference resources, leading to high resource consumption and slower summary generation speeds. This reliance necessitates substantial computational resources and GPU support, increasing deployment and operational costs. The applicability of our method is limited for users who do not have access to advanced LLMs.

Cluster Structure Stability Our retriever depends on the document cluster structure, which can exhibit instability across different document collections and topics. Despite the use of graph embeddings and clustering algorithms, the quality and consistency of document clusters may be affected by noise and data distribution, impacting the final retrieval and summarization performance. Therefore, improving the stability and robustness of the cluster structure remains a significant research challenge.

Evaluation Metric Limitations Although the G-Eval metric partially reflects summary generation quality, it still relies on existing reference documents for evaluation and cannot fully measure the creativity and diversity of summaries. Moreover, traditional metrics such as R-2 and BERTScore may not accurately reflect the actual quality of the summaries in certain cases, indicating the need for further improvement in evaluation methods.

B Reflection and Refinetion Module

The most central module of our summarizer is the Reflection and Refinement module, whose complete framework is shown in Figure 8. The cluster topics and relevance scores are generated by *gpt-3.5-turbo* using prompt shown in Figure 9. This is the key cluster information that guides the adaptive rearrangement module and the reflection and refinement module.

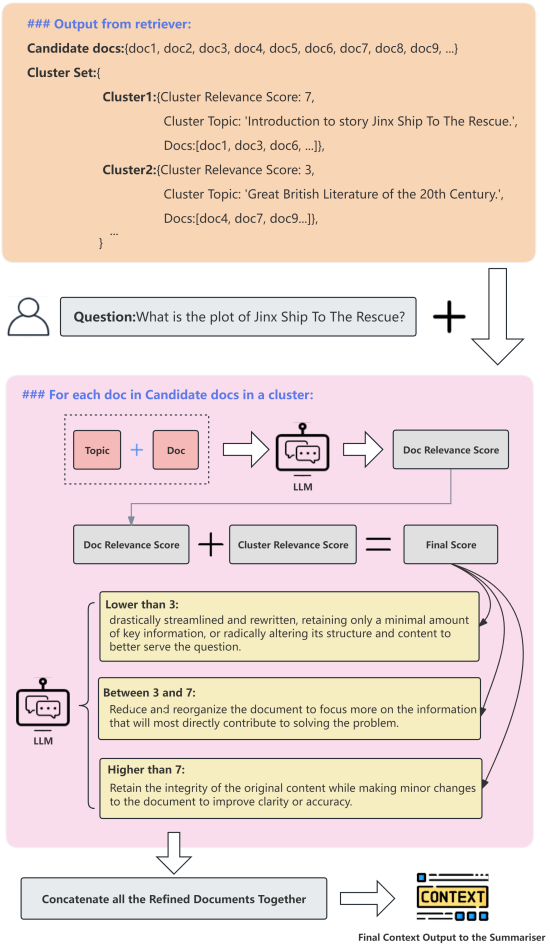


Figure 8: Framework of Refilection and Refine Module.

In the refinement phase, we classify documents into 3 categories based on the weighted sum of each document's cluster relevance score and document relevance score, executing three different rewriting strategies, the prompt is shown in Figure 10:

High Relevance Documents (scores 7-10) Perform minor edits to improve the accuracy and clarity of the document.

System:
You are a document processing and summarization specialist adept at distilling a unified theme from a collection of documents.

User:
Given a cluster of documents and a query, generate a topic of this cluster. Assess how relevant this topic is to the query and provide a relevance score (from 1 to 10). The output should be in JSON format, specifying the cluster identifier, the generated topic, and the relevance score.

Note:
- The cluster must have a generated topic that represents the collective content of the documents within it.
- The relevance score should reflect how well the generated topic addresses the query.
- Be aware of potential clustering inaccuracies; not all documents may align perfectly with the generated topic. Focus on synthesizing the most representative and accurate theme from the majority of the documents.

Cluster:
{All documents of this cluster}

Query:
{query}

Figure 9: Cluster Topic and Relevance Generation Prompt.

Medium Relevance Documents (scores 4-6) Restructure the document to focus on information directly related to the issue.

Low Relevance Documents (scores 1-3) Simplify significantly, retaining only the core information, or completely rewrite to meet query needs.

System:
You are skilled at modifying documents based on their relevance to specific questions, using pre-assigned relevance scores.

User:
Given a document, a relevance score (from 1 to 10), and a query, revise the document according to the relevance score:

- For a high relevance score (7-10), preserve the document's original integrity while making subtle modifications to increase clarity or accuracy.
- For a moderate relevance score (4-6), condense and reorganize the document to emphasize information that directly addresses the query.
- For a low relevance score (1-3), drastically simplify or completely revamp the document, retaining only crucial details or restructuring it to better suit the query.

Relevance Score:
{relevance_score}

Query:
{query}

Document:
{document}

Figure 10: Refinement According to Relevance.

C Experimental Setup

C.1 Data Preparation

To address the challenges of the ODMDS, we first performed meticulous preprocessing on the dataset. Using the *word_tokenize* method from the *nltk* library, we segmented each document into sub-documents no longer than 8912 tokens, and recorded these segmentation actions as undirected edges to maintain the natural connections between documents. Additionally, using OpenAI's text embedding model *text-embedding-ada-002*, we converted the text embeddings into high-dimensional vectors of 3912 dimensions. Next, we reduced the dimensionality of vectors generated by TF-IDF and GPT-embedding to 500 dimensions using PCA, and performed weighted summation to obtain the feature representation of each sub-document.

C.2 Graph Construction and Embedding

The next step in constructing the document graph is to implement a GAE. The first two GCN layers are followed by batch normalization and dropout with a 50% drop rate to prevent overfitting. The last convolutional layer does not include dropout to stabilize the learned embeddings. The model receives the input feature matrix and edge list, processes them through successive GCN layers, and uses the ReLU activation function for normalization after each layer. The output node embeddings are used to reconstruct the graph's adjacency matrix, and the reconstruction loss is calculated to evaluate model performance. The entire model is trained using the Adam optimizer with a learning rate of 0.01 and L2 regularization weight decay of 0.0005 to enhance generalization capability. The training process lasts for 300 epochs.

C.3 Clustering and Document Retrieval

For the clustering stage, we used the K-Means and DBSCAN algorithms, achieving the best results with DBSCAN configured with *eps* equal to 0.5 and *min_samples* equal to 2. We employed a hybrid retrieval method to determine the initial candidate document set, with the BM25 and GPT-Embedding scores weighted at 0.6 and 0.4, respectively. We selected 3, 8, and 10 documents as the final input for the summarizer in the Story dataset, and 1, 3, and 6 documents in the Meeting dataset.

C.4 LLM in the Summarizer Pipeline

We used LLM not only in the final output summary step but also multiple times throughout the summarizer framework. For example, in generating cluster topics and topic relevance scores, and in the document reflection and refinement modules. The LLM used in these modules is *gpt-3.5-turbo*. In the final summary generation process, we tested *llama-3-70b*, *gpt-3.5-turbo*, *gpt-4.0-turbo*, and *gpt-4o*. Their input length limits are 8k, 16k, 128k, and 128k respectively.

D Summarizer Metrics

For evaluating the quality of generated summaries, we adopt three different evaluation metrics:

ROUGE ROUGE assesses summaries by calculating the overlap of words between the candidate and reference summaries. Specifically, we report the F1 score of ROUGE-2, which considers the overlap of bigrams. This metric evaluates the quality of summaries by comparing the precision and recall between the generated and reference summaries.

BERTScore BERTScore calculates the similarity between the reference and generated summaries using contextual word embeddings. It employs the BERT model for word embeddings and uses the F1 score as the evaluation metric, which balances precision and recall by considering the semantic similarity between the generated and reference summaries.

G-EVAL G-Eval is a framework that uses LLMs combined with the Chain of Thought approach and form-filling paradigms to assess the quality of natural language generation outputs. Using *gpt-3.5-turbo* as the backbone, G-EVAL scores summaries based on dimensions such as consistency, coherence, relevance, and fluency. Due to input token limitations, it only compares predicted and reference summaries, with scoring criteria including consistency and relevance. After scoring, the average score of each example is calculated as the metric for the quality of the model-generated summaries. The prompt for consistency and relevance scores are shown in the Figure 11 and 12, and their scores range from 0-5.

System:
You are tasked with evaluating a summary of a news article based on its relevance. The goal is to ensure that the summary captures the essential information from the source document without including redundant or unnecessary details.

User:
Provide a numerical relevance score based on your assessment of the summary, using the evaluation criteria of Relevance (1-5), which measures the accuracy and conciseness of the summary in representing the key content of the source document. There is no need to explain your rating. Please review the instructions thoroughly and keep them accessible during your evaluation.

Evaluation Steps:
1. Read both the summary and the source news article carefully.
2. Identify the main points of the news article.
3. Evaluate how comprehensively the summary captures these main points and assess the presence of any irrelevant or excessive information.
4. Rate the summary on a scale of 1 to 5 based on its relevance to the main content of the news article.

NEWS ARTICLE:
{relevant_documents}

SUMMARY:
{response}

Figure 11: Relevance Score Prompt of G-Eval.

System:
You are tasked with evaluating the consistency of a summary based on its factual alignment with a news article. The primary goal is to ensure that the summary accurately reflects the facts presented in the source document without introducing any unsupported or incorrect details.

User:
Provide a numerical consistency score based on your assessment of the summary, using the evaluation criteria of Consistency (1-5), which measures the factual correctness and alignment of the summary with the original news article. Please refrain from explaining your rating. Carefully review these instructions and keep them handy during your evaluation.

Evaluation Steps:
1. Read the news article thoroughly to discern all the key facts and details it presents.
2. Examine the summary and compare it directly against the news article, checking for any factual discrepancies not supported by the article.
3. Assign a score for consistency based on how factually aligned the summary is with the source article.

NEWS ARTICLE:
{relevant_documents}

SUMMARY:
{response}

Figure 12: Consistency Score Prompt of G-Eval.

E Ablation Experiment on ODSum-Meeting

Figure 13 shows the experimental results under the ODSum-Meeting dataset after removing the three modules, and its results also support the analyses among the Section 5.3, consistent with the characteristics of the ODSum-Story dataset.

F Criteria for Human Evaluation

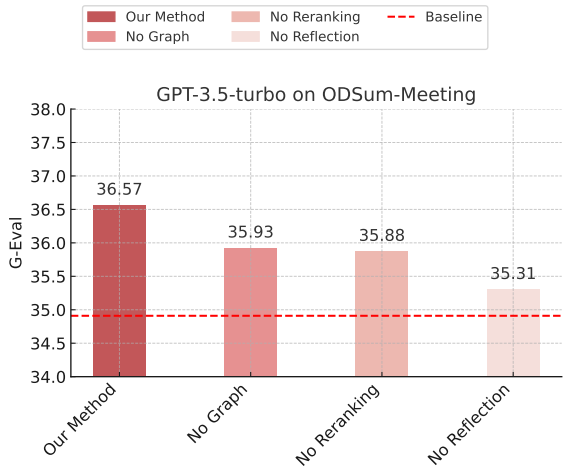
The quality of the generated summaries was assessed based on the following criteria:

Accuracy and Coverage The summary should accurately capture the core information from the original document, faithfully reflecting the main ideas without introducing errors or irrelevant information. It must correctly represent the key points, ensuring that no significant content is overlooked or misrepresented.

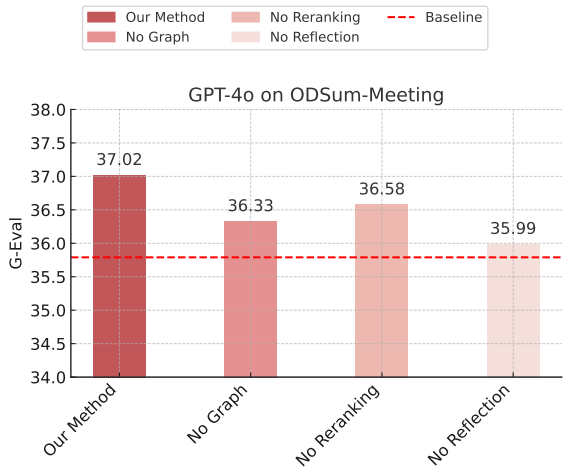
Conciseness and Coherence The summary should be concise and easy to understand, presenting information in a smooth, logical, and readable manner. It should avoid unnecessary length and redundancy, with sentences that are logically connected and flow naturally.

Relevance to the Questions The summary should directly answer the posed questions, including all relevant information while excluding unnecessary or irrelevant details. It should effectively address all the questions without digression.

Substantial and Meaningful Content The summary should provide substantial and useful information, avoiding empty or repetitive statements. It



(a) gpt-3.5-turbo on ODSum-Meeting.



(b) gpt-4o on ODSum-Meeting.

Figure 13: Ablation Experiment on Meeting Dataset. The baseline is Hybrid Retrieval, No Graph denotes the removal of the document graph construction and clustering module, No Reranking denotes the removal of the cluster-based adaptive re-ranking module, No Reflection denotes the removal of cluster-guided reflection and refinement module.

must not contain redundant, hollow, or meaningless content, ensuring that all included information is valuable.