Retrieval Enhanced Long Document Summarization with Question-Answering Blueprints

Litu Ou

Fourth Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2024

Abstract

Recent advances in large language models (LLMs) have significantly extended the context length, enabling long document summarization to be accomplished by directly processing the entire input. Nevertheless, it is often the case that the summary contains only the most relevant information from the source document while omitting less important details. This raises the question of whether all parts of the input are helpful for generating the summary. Therefore, the focus of this work is to explore how selecting salient information before summarization can improve the quality of generated summaries.

Since selecting salient information is similar to extractive summarization, we start by assessing the performance of oracle extractive summarization in long document summarization tasks. This exploration offers valuable insights into the potential gains from high-quality content selection. Subsequently, we investigate how to select salient information solely from the input document, without specific queries to focus on. Inspired by prior research that generates text plans in the form of question-answer (QA) pairs as *blueprints* that can be viewed as an outline representation, we propose utilizing retrieval against these blueprints to identify potentially relevant information. Our pipeline begins with generating text plans from the input document, then it retrieves relevant input passages based on these plans, and finally, it uses the selected passages to create the final summaries. Our results demonstrate that a simple setup involving blueprint generation and summarization, executed by fine-tuned end-to-end models combined with efficient lexical retrieval, can achieve competitive results on certain faithfulness metrics with a slight compromise on generation quality, highlighting the potential of our proposed pipeline.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Litu Ou*)

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Mirella Lapata, for her invaluable guidance throughout the completion of this undergraduate project. Since June 2023, I have had the honor of working on my undergraduate project under the supervision of Professor Mirella Lapata. Although she is very busy with her daily work, she patiently guides me through reading papers, reproducing them, analyzing experimental results, and proposing new ideas for improvement. Her rigorous attitude towards research and dedication to work have greatly influenced me. Both the successful completion of this project and my personal academic growth are attributed to her hard work.

Next, I would like to express my special gratitude to Dr. Yao Fu for leading me into the gates of Natural Language Processing two years ago. His patient guidance and wisdom made me accustomed to reading papers, reproducing results, etc. The knowledge I gained in this period of time was instrumental for equipping me with the knowledge needed to work on the undergraduate project.

Upon graduation from the University of Edinburgh, I would like to express my gratitude to my personal tutor, David Sterratt. Over the past four years, he has consistently provided assistance whenever I needed it. He is a kind, careful, and thoughtful tutor who works diligently to support his students.

Finally, I want to express my gratitude to my parents, for their huge and indispensable support over the years and being there to help me even at the hardest times.

Table of Contents

Chapter 1

Introduction

Due to the quadratic complexity of self-attention (Vaswani et al., 2017), traditional transformer-based language models (LMs) such as BERT (Devlin et al., 2019) and BART (Lewis et al., 2020) have limited context length despite being pretrained on large corpora. This limitation hinders the effectiveness of such models in tasks requiring long inputs, such as book-level summarization and open-domain question answering, where multiple documents are retrieved and concatenated. To extend the context length of transformer-based LMs, early work focused on approximating self-attention with combinations of local and global attention, reducing computational complexity while keeping performance loss minimal. More recently, with the rise of large language models (LLMs), the focus has shifted towards developing efficient fine-tuning methods to adapt existing LLMs to longer input lengths, thereby increasing the context length to beyond 100K tokens (Chen et al., 2024).

Previous methods for improving LMs primarily focused on processing longer sequences, mostly addressing the aspect of modeling efficiency. However, these advancements have paid little attention to two other crucial aspects: hallucination and context utilization, which are essential for accurate and efficient downstream deployment. The problem of hallucination, where models generate seemingly coherent but factually incorrect outputs, persists because the reasoning and verification mechanisms are still shrouded in opacity. Additionally, previous methods did not consider the fact that not all contexts provided to the model are necessarily helpful. Recent studies on context utilization showed that LLMs struggle to process information located in the middle of long input contexts (Liu et al., 2023). Focusing on selecting salient information, Xu et al. (2024) also found that by selecting context related to the input query, the performance of zero-shot LLMs improved significantly on multiple long context query-focused summarization and question-answering benchmarks.

Believing that addressing both hallucination and context utilization is critical for building more robust and trustworthy LMs for real-world applications, we explore, in this work, how content selection can prioritize salient and factual information. Drawing inspiration from existing work that generates blueprints - text plans in the form of question-answer (QA) pairs - as an intermediate step to long document summarization (Narayan et al., 2023), we envisage these blueprints as reflecting important aspects

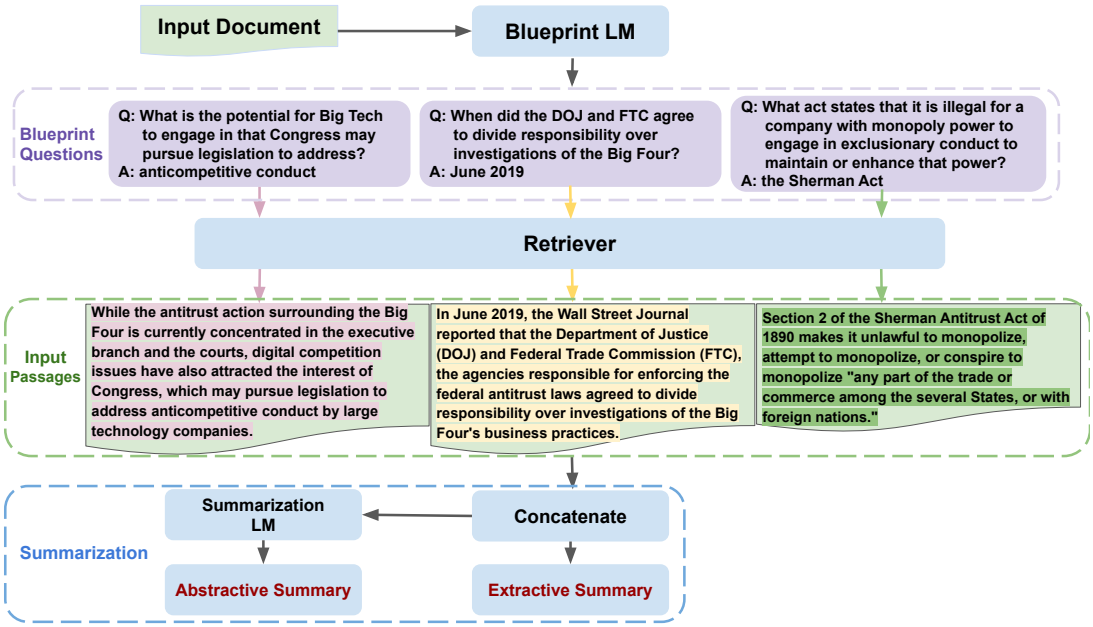

Figure 1.1: Overview of the whole summarization pipeline with QA blueprints and retrieval. The blueprint QA can be regarded as 1) an outline of the input document, which makes the generation less opaque, and 2) a set of facts derived from the input document, guiding the summary generation to be more faithful.

of the input content. We propose a pipeline (shown in Figure 1.1) that first generates such blueprints, then retrieves relevant information from the input using the blueprint questions, and finally performs summarization based on the retrieved information. One benefit of this pipeline is that the generated blueprints can be viewed as a gist of the input, making the process more interpretable. Our findings show that a straightforward approach, where blueprint generation and summarization are handled by fine-tuned end-to-end models coupled with efficient lexical retrieval, can achieve comparable results on various faithfulness metrics at the cost of a slight decline in the overall quality of the generated text.

1.1 The Potential of Content Selection

The idea of prioritizing salient information is similar to the task of extractive summarization, where the summary is composed by carefully selecting the most representative sentences from the input. However, most summarization datasets tend to provide abstractive summaries as the ground truth, rather than extracted sentences. As a result, previous work has investigated oracle extractive summarization, which uses reference abstractive summaries to create ground truth extractive summaries as high-quality training data for training extractive summarizers. These ground truth extractive summaries are often referred to as *oracle summaries*, since they are created based on reference abstractive summaries. To validate our hypothesis that selecting salient information is important for long document summarization, we evaluate the quality of oracle summaries to set an upper bound on content selection performance for our task.

Methods	GovReport			SummScreenfd		
	R-1	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
Greedy Beam Search (size 2)	**72.2**	**43.6**	27.2	**48.8**	**12.3**	17.8
LongT5-XL 4096	58.1	27.3	**29.7**	31.4	6.7	**18.8**

Table 1.1: Results of oracle extractive summarization using greedy selection with beam search of size 2, against LongT5-XL trained with the first 4096 input tokens on the GovReport and SummScreenfd datasets (details described in Section 4.1). R-1, R-2, and R-L are shorthands for the metrics ROUGE-1, ROUGE-2, and ROUGE-L, respectively.

We use the greedy selection with beam search oracle extractive summarization method proposed by Gu et al. (2022) in this experiment. For a beam size of k, this method maintains a beam of k partial summaries at each step. At every step, the method extends each partial summary in the beam by adding a new sentence from the input document. It selects the k extensions that maximize the current average ROUGE-1 and ROUGE-2 scores, which then form the new beam for the next step. This process continues until the addition of a new sentence does not increase the ROUGE score for any of the partial summaries in the beam, at which point the method concludes and outputs the highest-scoring partial summary as the final summary.

In our experiments, we utilize a beam size of 2, consistent with Gu et al. (2022), and compare the performance with the fine-tuned LongT5-XL model for abstractive summarization (Guo et al., 2022), which has demonstrated superior performance on multiple long-document summarization benchmarks. Table 1.1 summarizes the related results. We observe that the oracle method achieves significantly higher ROUGE-1 and ROUGE-2 scores compared to LongT5-XL, but lower ROUGE-L scores. This discrepancy can be explained by the oracle method's priority in maximizing ROUGE-1 and ROUGE-2 scores during sentence selection while neglecting ROUGE-L rewards. Nevertheless, the significant improvement in ROUGE-1 and ROUGE-2 scores indicates the potential of the content selection approach to outperform capable long context models.

1.2 Extracted Summary as Abstractive Input

Recent studies on long context models have discovered the *lost-in-the-middle* problem, where the ability of long context language models to process information in the middle of the input is significantly inferior compared to that at the start and end (Liu et al., 2023). Since traditional long document summarization models often utilize either the entire input or truncation that retains the first n tokens, the *lost-in-the-middle* issue can potentially impact these models, leading to a bias towards summarizing different parts of the input. Consequently, the final summary may lack comprehensiveness and faithfulness. This provides further motivation for employing content selection to reduce the context length and allow the model to focus on salient parts of the input in an unbiased manner.

Methods	GovReport			SummScreenfd		
	R-1	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
LongT5-XL 4096	58.1	27.3	29.7	31.4	6.7	18.8
Extract-then-summarize	59.1	32.3	30.1	33.5	8.2	19.2
Extract-then-summarize + oracle fine-tune	**67.3**	**36.8**	**32.8**	**39.7**	**11.1**	**22.0**
Extractive only	72.2	43.6	27.2	48.8	12.3	17.8

Table 1.2: Results using the extractive summary as input for abstractive summarization based on LongT5-XL. ”extract-then-summarize” is the method that uses extractive summary for abstractive summarization, with ”+ oracle fine-tune” representing the method with LongT5-XL fine-tuned on oracle extracted input sentences. We copy the LongT5-XL 4096 and extractive summarization results from Table 1.1 for clear comparisons.

To study the extent to which content selection can help long context models in focusing on important information in the input, we utilize the summary obtained from oracle extractive summarization as the input to an abstractive summarizer to establish an upper bound on the performance of salient content selection. We first test whether summarization performance can be improved by employing the same LongT5-XL model, replacing the truncated input document with the extractive summary. In addition, we fine-tune the same backbone model by replacing input documents in both training and validation data with the oracle-extracted inputs, allowing the model to learn how to better utilize the extracted inputs for summarization.

The results of the abstractive summarization over extractive summaries are shown in Table 1.2. We use ”extract-then-summarize” as an abbreviation to denote the approach using the LongT5-XL abstractive summarization model over oracle extracted input sentences at test time only, and ”+ oracle fine-tune” to denote fine-tuning LongT5-XL on oracle extracted input sentences. Compared to the vanilla LongT5-XL model, The extract-then-summarize method provides a considerable gain in all ROUGE metrics, demonstrating the effectiveness of prioritizing salient information in the input. More importantly, we observe a much larger gain in the ”extract-then-summarize + oracle fine-tune” method, increasing the ROUGE-1 score by almost 10 on both datasets and bridging the gap between oracle extractive summarization on both ROUGE-1 and ROUGE-2. We also notice that ”extract-then-summarize + oracle fine-tune” yields the best ROUGE-L scores among all experiments, while oracle extractive summarization has the lowest ROUGE-L scores, demonstrating the ability of oracle fine-tune to capture more informative content from the reference summaries.

1.3 Comparing Oracle Methods

In the previous sections, we demonstrated the advantage of the greedy selection with the beam search method in producing oracle summaries with high ROUGE scores under various settings. Nevertheless, previous literature has suggested that greedy oracle methods have a bias towards selecting sentences at the start of the input (Kedzie et al.,

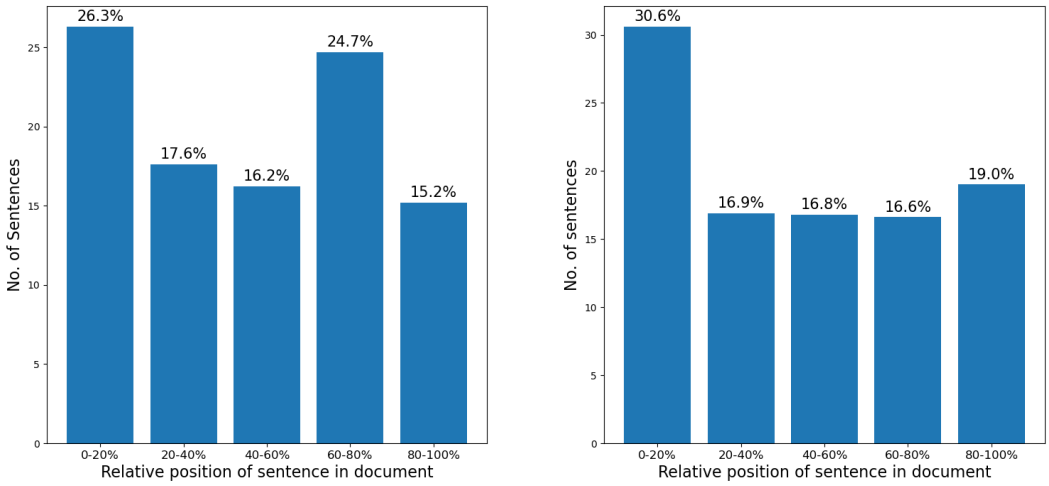

Figure 1.2: Proportions of extracted sentences at each range of relative positions for GovReport (left) and SummScreenfd (right). For example, the 30.6% on the 0-20% bar for SummScreenfd indicates that 30.6% of the extracted sentences are located in the first 20% of the input document.

2018; Grenander et al., 2019). As a background study, we explored the lead bias issue in oracle extractive summarization approaches on the GovReport (Huang et al., 2021) and SummScreenfd (Chen et al., 2022) datasets. Our results are shown in Figure 1.2. Specifically, we observe the lead bias problem in both datasets, where the proportion of extracted sentences in the first 20% of the input is the highest, with 26.3% in GovReport and 30.6% in SummScreenfd, each containing more than one-quarter of all sentences. The proportions for the rest of the positions are therefore lower than the average, with the only exception in the 60%-80% range for GovReport, which has 24.7% of sentences. Even though many oracle-extracted sentences lie in the front part of the input, it does not mean we can simply ignore the contents in the middle or at the end, as together they represents 73.7% and 69.4% of oracle sentences, respectively.

A more recent approach attempted to build a better sentence labelling scheme called OREO (**OR**acle **E**xpectati**O**n labeling) for extractive summarization (Xu and Lapata, 2022). OREO creates soft sentence labels by incorporating information from multiple high-quality summary hypotheses obtained using beam search. It estimates the expected summary quality score for each sentence based on its presence in the different high-quality summary hypotheses, rather than relying on a single greedy or beam search extraction method that considers only the overall score. To understand the effectiveness of OREO on long document summarization tasks, we conduct a comparison of these two approaches on our tasks of extractive summarization and abstractive summarization on extracted summaries (extract-then-summarize) to establish the optimal method for future experiments. In terms of the experimental setting, we use a lightweight setup consisting of BART-base as the backbone model for extract-then-summarize, as the focus here is to compare performances rather than achieving better ROUGE scores.

The results of this comparison are summarized in Table 1.3. For the task of extractive summarization, we observe that greedy selection with beam search significantly out-

Methods	GovReport			SummScreenfd		
	R-1	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
ExtSumm						
Greedy Beam Search (size 2)	**72.2**	**43.6**	**27.2**	**48.8**	**12.3**	**17.8**
OREO	59.4	36.8	25.4	40.5	11.5	16.1
Extract-then-summarize w/ fine-tune						
Greedy Beam Search (size 2)	**68.6**	**37.1**	**29.8**	**41.7**	**11.6**	**22.6**
OREO	60.9	31.7	27.3	33.1	8.7	19.3

Table 1.3: Results on extractive summarization (ExtSumm) and abstractive summarization w/ extracted summary and fine-tune (extract-then-summarize w/ fine-tune). Here *OREO* is the shorthand for the oracle expectation approach.

perform OREO on all ROUGE metrics. This is consistent with the results reported by Xu and Lapata (2022), where the performance of OREO was also worse compared to beam search on the CNN/DM dataset (See et al., 2017). Furthermore, OREO's performance continues to be worse on the extract-then-summarize task, with similar performance gaps observed in extractive summarization. This potentially suggests that for the extract-then-summarize task, the oracle ROUGE score is positively correlated with the ROUGE score at inference. We therefore adopt greedy selection with beam search for all our future experiments in this work.

1.4 Structure of Report

Chapter 1 is the introduction where we detail the motivation behind the project and present results of background studies in support of further investigation. Chapter 2 surveys previous work related to the project's themes, including long-document summarization, extractive summarization and retrieval enhanced generation. In chapter 3 and 4, we outline the methodology deployed in the proposed pipeline and provide details of the datasets used, as well as the experimental setups. Chapter 5 presents and discusses the results of our proposed pipeline, followed by an overview of the project and a discussion of future directions.

Chapter 2

Related Work

2.1 Extractive Summarization

Early work on extractive summarization, prior to the emergence of the transformer architecture (Vaswani et al., 2017), approached this task by formulating it as a sentence classification or ranking problem. This was achieved using graph-based approaches (Erkan and Radev, 2004; Mihalcea and Tarau, 2004) or recurrent neural networks (Nallapati et al., 2016; Cheng and Lapata, 2016) to label sequences in the input, determining whether they should be included in the summary. The introduction of the transformer model enabled new approaches that could better capture long-range dependencies and attend to the entire input context. Multiple architectures extending the BERT model (Liu and Lapata, 2019; Cui et al., 2020) have demonstrated strong performances on the CNN/DM dataset (See et al., 2017), which is widely used for summarization tasks. More recently, there have been attempts to use LLMs for the extractive summarization task through prompting approaches (Zhang et al., 2023), showcasing the potential of LLMs to generate factual and human-readable summaries.

As mentioned in Section 1.1, previous literature also investigated better oracle extractive summarization methods to obtain oracle labels that achieve good results in terms of automatic evaluation metrics and improve the training of extractive summarization models. Most existing approaches build oracle labels by greedily selecting sentences that together maximize the ROUGE scores (Nallapati et al., 2016; Gu et al., 2022). Nevertheless, previous literature suggested that greedy oracle methods have a bias toward selecting sentences at the start of the input (Kedzie et al., 2018; Grenander et al., 2019). Addressing this issue, some recent work proposes different labeling schemes that sacrifice oracle ROUGE scores but improve the final performance of extractive summarization models (Xu and Lapata, 2022).

2.2 Abstractive Summarization

Long documents pose challenges for traditional attention mechanisms due to their quadratic computational and memory complexities. Consequently, much existing work

has focused on linear-time attention mechanisms, which combine local self-attention (for neighboring tokens) with full self-attention over a set of global tokens to minimize information loss (Beltagy et al., 2020; Zaheer et al., 2021; Guo et al., 2022). Similarly, as described in Section 1.2, the literature also explores using content selection to compress long contexts with an RNN model (Manakul and Gales, 2021).

The advance in LLMs has aroused interest in extending the context window of base LLMs. Since modifying the base model architecture requires pre-training, which is computationally expensive, existing approaches mainly focus on fine-tuning methods that adapt LLMs to longer contexts with manageable computational costs. For example, the position interpolation method (Chen et al., 2023) modifies rotary position embedding (Su et al., 2023) to extend the context length of LLAMA (Touvron et al., 2023a) to over 30k. Similarly, LoRa fine-tuning (Hu et al., 2021) via approximate attention has been deployed to adapt the context length to over 100k for 7b models (Chen et al., 2024), showing improvements in language modeling in terms of perplexity.

2.3 Conditional Generation

Most NLP tasks can be solved using end-to-end models, where desired outputs are generated given only the input. However, one problem with end-to-end approaches is that they are not interpretable and are prone to hallucination, generating outputs that are not fully grounded in the input. Various attempts have addressed this issue by proposing multi-stage pipelines, breaking down the problem to smaller and more logical sub-tasks. For example, Puduppully et al. (2019) broke down the data-to-text problem by introducing intermediate steps of content selection and planning, presenting a high-level organization of the input for better interpretability. Narayan et al. (2021) proposed a more readable representation consisting of entity chains that symbolize the logic flow of input, using it as a pre-training objective or an intermediate step of generation at inference.

With the advance of LLMs, previous work also attempts to build more sophisticated pipelines and craft more informative prompts exploiting the abilities of LLMs to follow very complex instructions. For example, chain-of-thought prompting has demonstrated significant performance gains in very large models by instructing them to include a sequence of reasoning steps before the answer in their outputs (Wei et al., 2023), modifying only the output format and leaving the rest of the pipeline unchanged. Intermediate generations are also used as a compression step in open-domain question-answering tasks by generating a summary of the input knowledge using a lightweight model (Xu et al., 2023), effectively reducing the computation cost while guaranteeing minimal performance loss.

Our work draws significant inspiration from Narayan et al. (2023), which uses QA pairs as the format for intermediate presentation. The idea is based on the *"Question Under Discussion"* theory, where discourse structure can be clarified by raising questions and their respective answers based on spans of input text (Riester, 2019). Specifically, each QA pair can be viewed as a piece of factual information from the input text, together representing a series of important points that construct a blueprint of the

Model	Seq len.	Avg.	QM	QASP	NQA	QLTY	MSQ	HQA	MFQA
GPT-43B	4k	26.44	15.56	23.66	15.64	49.35	11.08	28.91	40.90
+ ret	4k	29.32	16.60	23.45	19.81	51.55	14.95	34.26	44.63
GPT-43B	16k	29.45	16.09	25.75	16.94	50.05	14.74	37.48	45.08
+ ret	16k	**29.65**	15.69	23.82	21.11	47.90	15.52	36.14	47.39
Llama2-70B	4k	31.61	16.34	27.70	19.07	63.55	15.40	34.64	44.55
+ ret	4k	36.02	17.41	28.74	23.41	70.15	21.39	42.06	48.96
Llama2-70B	16k	36.78	16.72	30.92	22.32	**76.10**	18.78	43.97	48.63
+ ret	16k	37.23	**18.70**	29.54	23.12	70.90	23.28	44.81	50.24
Llama2-70B	32k	37.36	15.37	**31.88**	23.59	73.80	19.07	49.49	48.35
+ ret	32k	**39.60**	18.34	31.27	**24.53**	69.55	**26.72**	**53.89**	**52.91**
Llama2-7B	4k	22.65	14.25	22.07	14.38	40.90	8.66	23.13	35.20
+ ret	4k	**26.04**	16.45	22.97	18.18	43.25	14.68	26.62	40.10
Llama2-7B	32k	**28.20**	16.09	23.66	19.07	44.50	15.74	31.63	46.71
+ ret	32k	27.63	17.11	23.25	19.12	43.70	15.67	29.55	45.03

Figure 2.1: Results of retrieval on LLMs copied from Xu et al. (2024). *+ ret* denotes the results obtained incorporating retrieval. "QM" is a query-focused summarization dataset, while the others are QA datasets with long evidences sourced from the SCROLLS benchmark (Shaham et al., 2022) and LongBench (Bai et al., 2023).

input. Blueprints are generated in natural language, which is more human-readable and expressive, including further details and complex relations. This approach is studied across a number of long-form question-answering and summarization datasets, demonstrating that blueprints improve factuality and allow tighter control over the output.

2.4 Retrieval Enhanced Generation

Information retrieval is often applied to NLP tasks that require locating relevant texts from potentially unlimited input data, such as open-domain question-answering and fact checking. Given an input query, often in the form of a question or short summary, and some knowledge source containing documents or passages as external knowledge, a retriever needs to find the relevant documents or passages that will be used as input contexts for solving the knowledge-intensive task. The difference between retrieval and extractive summarization is that retrieval methods focus on whether passages contain information needed to answer the query, while extractive summarization selects sentences considering their importance and salience in representing the key points of the input document.

The significant advance of long context modeling extended context windows to the scale of 100K tokens (Chen et al., 2024; Fu et al., 2024) for LLMs, enabling models to process more information but also increasing inference costs. Moreover, the *lost-in-the-middle* problem (Liu et al., 2023) indicates that models cannot utilize long contexts very well, suggesting solutions focus on either improving context utilization or investigating whether the input contexts can be compressed with minimal loss. Addressing the latter point, Xu et al. (2024) studied how retrieval methods can be applied to query-focused

summarization or question-answering tasks requiring very long contexts as information sources, utilizing the input query to select only the most relevant information for efficient and focused inference. Additionally, Bai et al. (2023) also evaluated the performance of retrieval methods as content compression for a series of long context tasks.

We now describe the work of Xu et al. (2024) in detail, as we use a similar idea in our proposed pipeline. The main approach is to retrieve relevant passages from the input context as the new input to LLMs for query-focused NLP tasks. Specifically, the document is first chunked into uniform-length passages. A retriever then retrieves the top-k passages against the input query. Xu et al. (2024) showed that using retrieval improves the performance on multiple query-focused summarization and question-answering tasks for LLMs such as LLAMA-2 (Touvron et al., 2023b). Importantly, using retrieval yields better performance for LLMs of varying maximum context length, even those that can fit entire input contexts. This suggests the presence of *lost-in-the-middle* effect of LLMs when handling long input contexts, showing that retrieval helps the model to focus on salient information.

Chapter 3

Methodology

An illustration of our proposed pipeline is shown in Figure 1.1. Given an input document, we first generate QA blueprints using a fine-tuned end-to-end LM (Section 3.1). Next, each generated blueprint question is used as a query for a retriever to retrieve the top-k relevant passages from the input (Section 3.2). After obtaining all the retrieved passages for all the blueprint questions, we concatenate them for the use of extractive and abstractive summarization (Section 3.3).

3.1 Blueprint Generation

The notion of "blueprint" was first proposed by Narayan et al. (2023). A blueprint for an input document consists of QA pairs containing factual information from the input and can be viewed as a gist of what should be included in the final output. By generating blueprints as an intermediate step before summarization, we can guide summary generation with them, making the whole process more interpretable. Formally, given a document d, the blueprint generation stage employs a fine-tuned language model to generate an ordered sequence of QA pairs, also called blueprint, denoted as $(a_1, q_1), \cdots, (a_n, q_n)$ where a refers to the answer and q to the question. The number of QA pairs varies depending on the specific output of the model. Next, we describe how we obtain data to train models for blueprint generation and provide details about the base language model, as well as the input-output format.

Summarization datasets typically do not contain blueprints derived from the input. As a result, we need to extend existing datasets to include a reference blueprint, which will enable us to fine-tune an LM to generate blueprints during inference. We follow a similar approach as described by Narayan et al. (2023). Firstly, a model is trained on the SQuAD dataset (Rajpurkar et al., 2016), which uses the answer and context to predict the question. For example, given an answer of *coke* and a context *Coke is a soft drink*, the question predicted would be *What is a soft drink*. The question generation model is obtained by fine-tuning T5-3b (Raffel et al., 2020) using the modified SQuAD dataset format for question generation. For each example in the dataset, we extract the noun chunks in the reference summary using spaCy (Honnibal et al., 2020). We then use these extracted noun chunks as answers for each QA pair in the blueprint. Finally,

All generated QA pairs		RP	QA	CR	CO
Q_1: What is the residue left from?	A_1: demonic vanquishes	✓	✓	✓	✗
Q_2: What do the sisters lose when demonic vanquishes build up in the manor?	A_2: sleep	✓	✓	✓	✓
Q_3: What does the demonic residue take over?	A_3: their lives	✓	✓	✓	✗
Q_4: Who calls upon the Witch Doctor?	A_4: they	✓	✓	✗	
Q_5: Who does Leo not trust?	A_5: whom	✓	✓	✗	
Q_6: Who does not trust the Witch Doctor?	A_6: Leo	✓	✓	✓	✓
Q_7: What does the Witch Doctor believe the sisters are?	A_7: the massive amount	✓	✗		
Q_8: Who does the Witch Doctor think is evil?	A_8: Witch	✗			
When residue left from demonic vanquishes builds up in the manor, the sisters lose sleep and it takes over their lives. They call upon the Witch Doctor, whom Leo does not trust. The Witch Doctor makes a house call to the Charmed Ones but because of the massive amount of demonic energy he believes they are evil.					

Table 3.1: A blueprint filter example for a set of QA pairs based on a summary. RP, QA, CR and CO are shorthands for Repetition, Question-Answering Consistency, Coreference and Coverage. QA pairs that pass/fail each filter are labeled with ✓/✗.

we utilize each blueprint answer and the reference summary as context to generate the blueprint question.

Empirical examination of the blueprint has revealed many errors. hence, we filter out undesired QA pairs and use the filtered blueprint as references when training the blueprint LM. Table 3.1 illustrates an example of how the generated blueprint is filtered. We describe the details of the filtering methods used below.

Repetition A common error observed in the question generation model is that the blueprint answer is sometimes included in the generated blueprint question. For example: *"Q: Where is Paris? A: Paris"*. Such QA pairs contain repeated information, which makes them less meaningful as reference blueprints. We filter these cases by checking if the blueprint answer is contained in the generated blueprint question.

Question-Answering Consistency Another error made by the question generation model is that the generated blueprint question might not be answered correctly by the blueprint answer. We filter these cases by checking the round-trip QA consistency (Alberti et al., 2019), where we utilize a QA model to generate an answer based on the generated question and the given context, checking whether it matches the blueprint answer.

Coreferences Some extracted noun chunks can be confusing. Consider the sentence *"They built a company in Edinburgh."* If we use the pronoun *"They"* as the answer, we would likely have the QA pair *"Q: Who built a company in Edinburgh? A: They."* In this case, we do not know what *"They"* stands for as its coreference lies somewhere else in the context, making the QA pair uninformative. We filter these cases by removing answers with occurrences of words that are highly related to this issue, such as *"they", "these", "those"*.

Coverage This step is identical to the coverage check first introduced by Narayan et al. (2023). We construct the bag of tokens to include only the noun chunks and repeatedly select the blueprint question with the highest overlap until the bag is empty.

In order to train models to be able to generate blueprints at inference, we use the reference blueprints collected and filtered previously, to fine-tune a LongT5-XL model (Guo et al., 2022) to directly generate blueprints given input documents. Following Narayan et al. (2023), we structure the model output as a sequence of QA pairs in the format of $a_1; q_1; \cdots; a_n; q_n$ with the prepended prefixes *Q:* and *A:*. The final blueprint is therefore the sequence of QA pairs $(q_1, a_1), \cdots, (q_n, a_n)$.

3.1.1 LongT5

LongT5 is a long context language model built upon T5, featuring an approximate attention mechanism that reduces the quadratic complexity when computing self-attention (Vaswani et al., 2017). Instead of attending to every token in the input context, LongT5 employs a combination of local and global attention. Each token attends to a fixed window of local context and several global tokens, as shown in Figure 3.1. The global tokens are obtained by dividing the input context into blocks, and for each block, an aggregate representation is computed by summing and then normalizing the embedding of all the tokens within it. We use LongT5 as our backbone model due to its ability in handling long inputs and its strong performance on long document summarization tasks, as evidenced by the SCROLLS benchmark (Shaham et al., 2022).

3.2 Retrieval

As blueprints can serve as a gist of the input document, we can utilize the blueprint questions as queries to retrieve salient passages back from the input document, performing high-quality content selection that ideally removes some less relevant information. Formally, an input document d is first split into uniform length passages $c_1, \cdots, c_m$ be-

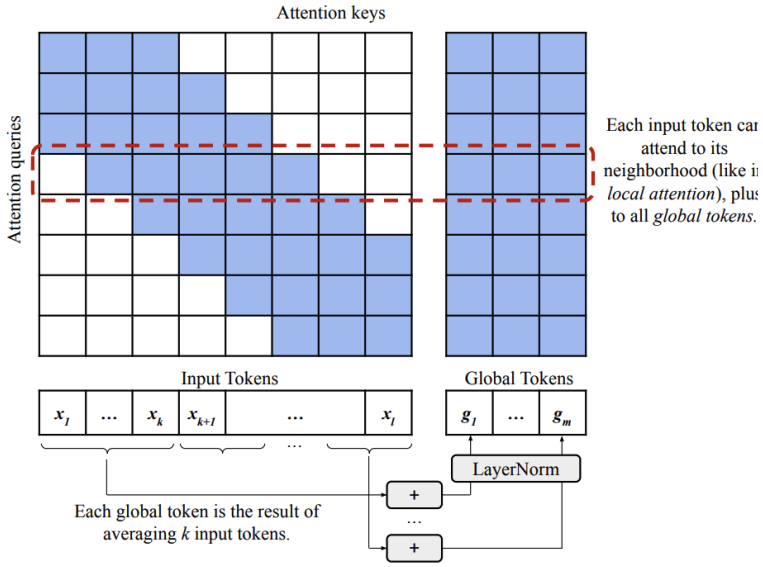

Figure 3.1: Illustration of the LongT5 attention mechanism by Guo et al. (2022).

fore a retriever R retrieves the top-k most relevant passages given the blueprint question q.

3.2.1 BM25

BM25 (Robertson and Zaragoza, 2009) is a lexical ranking function used in information retrieval systems, particularly for text documents. It operates as a bag-of-words retrieval function that ranks documents based on the relevance scores of matching terms against the query. Given a query q containing query terms $q_1, \cdots, q_n$, the BM25 score of a passage p for the query q is calculated as:

$$score(p,q) = \sum_{i=1}^{n} \left[IDF(q_i) \times \frac{(k_1+1) \times freq(q_i,p)}{k_1 \times (1-b+b \times \frac{length(p)}{avgdl}) + freq(q_i,p)} \right] \tag{3.1}$$

Where $IDF(q_i)$ represents the Inverse Document Frequency of the term q_i, $freq(q_i,p)$ denotes the term frequency of q_i in passage p, $length(p)$ represents the length (number of words) of passage p, $avgdl$ stands for the average document length in the corpus, and k_1 and b are free parameters.

3.2.2 Dense Retrieval

Lexical approaches like BM25 cannot capture semantic relationships between queries and documents. Dense retrieval attempts to tackle this limitation by encoding both queries and documents into dense vector representations, often leveraging transformer-based models (Karpukhin et al., 2020a). This approach allows for retrieval based on semantic similarity, where documents with similar meanings will have closer vectors

in the embedding space. Formally, given a query q and a passage p, dense retrieval uses a query encoder E_Q and passage encoder E_P to generate embedding for the query q and passage p respectively. The embedding similarity is used to rank passages, where a higher score indicates a higher likelihood of relevance to the query. For example, Karpukhin et al. (2020a) used dot product similarity calculated as:

$$sim(q,p) = E_Q(q)^T E_P(p) \tag{3.2}$$

3.2.3 Tradeoff

BM25 is very efficient to run due to its reliance on lexical features such as term frequency and inverse document frequency, making it computationally inexpensive to calculate and implement. On the contrary, dense embedding projects sequences into high-dimensional spaces, allowing more features to be considered, such as semantic similarity and factual relevance.

In this work, we utilize BM25 due to task-specific reasons. Our task needs to retrieve passages for every blueprint question, which often yields many passages even in the top-1 setting. In order to utilize these passages for extractive summarization (described in more detail in the next section), we need to keep the total number of tokens in these passages similar to the average reference summary length to allow meaningful evaluation. This makes dense retrieval less effective, as it is often applied to tasks like open-domain question answering, where long passages are retrieved. Lexical approaches like BM25 can perform quite well by capturing the keywords in these relatively short passages while being significantly more efficient to run compared to dense retrieval.

3.3 Composing the Summary

To obtain final summaries, we concatenate all retrieved passages for each blueprint question and use them 1) directly for extractive summarization and 2) as input to a summarizer LM to generate an abstractive summary. Formally, for the task of extractive summarization, after retrieving the top-k passages$p_1^{(i)}, \cdots, p_k^{(i)}$ ordered according to their positions in the input document d for blueprint questions $q^{(i)}$, we concatenate all the passages to form the extractive summary:

$$S_{ext} = concat(p_1^{(1)}, \cdots, p_k^{(1)}, p_1^{(2)}, \cdots, p_1^{(n)}, \cdots, p_k^{(n)}) \tag{3.3}$$

This extractive summary is used as the input to the abstractive summarizer. Similar to the blueprint generation stage, we also fine-tune a LongT5 model to perform the abstractive summarization step. The process can be described as:

$$S_{abs} = LongT5_{abs}(S_{ext}) \tag{3.4}$$

Chapter 4

Experimental Setting

4.1 Datasets

In this project, we focus on summarization datasets that do not require a query to specify the topic of interest, i.e. query-focused summarization; instead, we generate summaries solely based on the input document. We use the following two long document summarization datasets that have different domains:

GovReport GovReport (Huang et al., 2021) is a long-document summarization dataset containing reports published by the U.S. Government Accountability Office (GAO)[1] and the Congressional Research Service (CRS)[2]. The dataset is split into training, validation and test sets with approximate ratio of 90-5-5. All summaries are written in the form of abstractive summaries by expert annotators.

SummScreen SummScreen (Chen et al., 2022) is a long-document summarization dataset comprised of TV show transcripts. Unlike formally written articles, these transcripts mainly consist of dialogues between TV show characters and descriptions of environments or character interactions. The dialogues' contexts are further identified and separated by descriptions of scenes with the format [*Scene Description*]. All summaries are written by human annotators and can be viewed as recaps of the entire transcript in the form of abstractive summaries.

The original SummScreen dataset has two distinct splits: TV MegaSite, Inc. (TMS)[3], which are mostly soap operas, and ForeverDreaming (FD)[4], which has more genres than TMS. Following most existing work, we utilize the FD split as it contains transcripts from more TV shows than the TMS split, thereby making the dataset more diverse and potentially generalizing well to the summarization of other TV show transcripts. The FD split has training, validation and test sets with approximate ratio of 10-1-1.

[1] `www.gao.gov`
[2] `crsreports.congress.gov`
[3] `tvmegasite.net`
[4] `transcripts.foreverdreaming.org`

Dataset	Domain	# examples (train/val/test)	Avg # tokens Input	Avg # tokens Output
GovReport	Government	17,457 / 972 / 973	9,616	597
SummScreenfd	TV show	3,673 / 338 / 337	8,987	137

Table 4.1: Statistics of the two summarization datasets used in our experiments.

The statistics for both datasets are shown in Table 4.1. We also include an example input/output pair for each dataset in Appendix A.1. For each dataset, the data is collected from publicly available Hugging Face repositories. We keep all examples from the collected datasets and remove all non-ASCII characters in the inputs and summaries.

4.2 Experiment Setup

We fine-tune T5-3b on the question generation task described in Section 3.1 that generates blueprint questions given blueprint answers and a supporting context. Each blueprint answer and its supporting context are attached with a prefix and separated by a single space: "*Answer:* {*answer*} *Context:* {*context*}". The output is the plain question with no specific prefix. We fine-tune the model for 10 epochs and use the checkpoint with the lowest validation loss. Our batch size is 8, and we utilize the Adam optimizer (Kingma and Ba, 2017) with learning rate 1e-4, betas (0.9, 0.999) and epsilon 1e-8. Questions are decoded with greedy decoding and no repetition penalty.

For the blueprint generation stage, we fine-tune the LongT5-XL (3b parameters) model that generates a string of concatenated blueprint QA pairs given the input document. Following (Narayan et al., 2023), we truncate the input document to 4096 tokens for all datasets, constituting about half of original input length according to Table 4.1. The model is fine-tuned for 40 epochs, and we select the checkpoint with the best validation set average ROUGE score for future inference. We use a batch size of 8 and the Adafactor optimizer (Shazeer and Stern, 2018) with a learning rate of 1e-3, identical to the original LongT5 fine-tuning learning rate. The maximum output length is set to 1024 for GovReport and 512 for SummScreenfd to ensure optimal performance, considering the average output length shown in Table 4.1. The blueprint is decoded with greedy coding and repetition penalty set to 1.2, based on empirical observations that the fine-tuned LongT5 model sometimes generates repeated n-grams.

Retrieval against blueprint questions is performed using BM25, as discussed in Section 3.2. In practice, we observe that the blueprint generation stage yields on average 20-30 questions for GovReport and 10-20 questions for SummScreenfd. To enable meaningful evaluation for the extractive summarization task defined in Section 3.3, we chunk the input into 40-token segments for GovReport and 20-token segments for SummScreenfd. We then concatenate the top-1 retrieved passage for each blueprint question.

For the final stage of abstractive summarization over retrieved passages, we also fine-tune a LongT5-XL model that generates an abstractive summary given the concatenated

Generation Task	Model	# Tokens		Optimizer	LR	Epochs
		Input	Output			
Question	T5-3B	All	-	Adam	1e-4	10
Blueprint	LongT5-XL	4096	1,024 / 512	Adafactor	1e-3	40
Summary	LongT5-XL	All	1,024 / 512	Adafactor	1e-3	40

Table 4.2: List of important hyperparameters used for each stage of the pipeline excluding retrieval. For question generation, we do not specify maximum input and output length as both are within range of the T5 context length. The output lengths vary across the datasets.

retrieved passages. As the concatenated retrieved passages are significantly shorter in length compared to the original input document, we do not truncate them during fine-tuning. Following the extract-then-summarize approach described in Section 1.2, we replace inputs in training and validation data with oracle-extracted passages for fine-tuning. The hyperparameters used are identical to those used in the blueprint generation stage. An overview of all the important hyperparameter setups is provided in Table 4.2.

4.3 Baseline

In addition to the vanilla LongT5-XL baseline, we also consider the MemSum extractive summarization model (Gu et al., 2022), which has achieved state-of-the-art results on the GovReport dataset[5]. Specifically, we use the MemSum model as an extractive summarization baseline and employed extract-then-summarize with MemSum extracted summaries as an abstractive summarization baseline for comparison with our proposed pipeline.

4.3.1 MemSum

MemSum is an extractive summarization model that treats extractive summarization as a multi-step episodic Markov Decision Process (MDP). In each step, MemSum defines a sentence state composed of three components: 1) the local content of the sentence, encoded by a bidirectional LSTM (Hochreiter and Schmidhuber, 1997), 2) the global context within the document, encoded by another bidirectional LSTM over the sentence embedding, and 3) the extraction history, which captures information about previously extracted sentences using multi-head attention, as illustrated in Figure 4.1. MemSum employs a policy network to score sentences based on the sentence state, allowing it to select an action of either extracting the top-scoring sentence or stopping the extraction process. MemSum is trained with reinforcement learning to maximize the average ROUGE-1/2/L scores of the extracted summaries.

[5] `https://paperswithcode.com/sota/extractive-text-summarization-on-govreport`

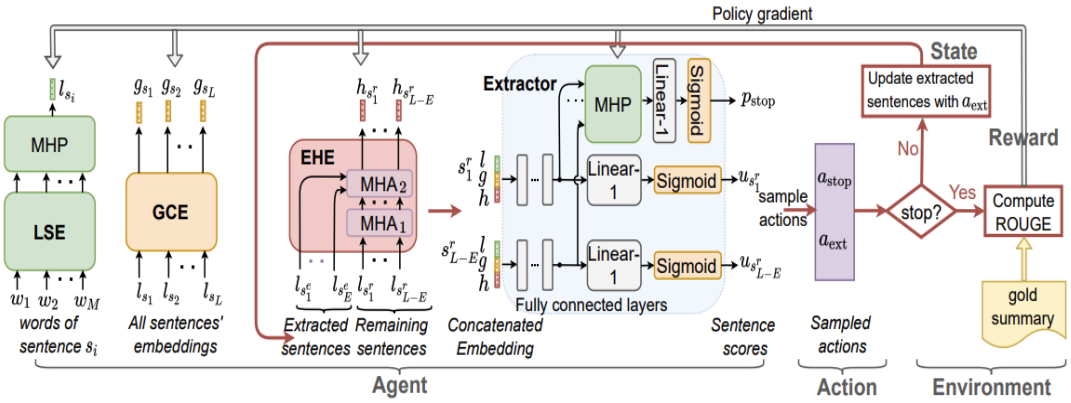

Figure 4.1: Architecture of the MemSum extractive summarization model by Gu et al. (2022)

In addition to the extractive summarization model described above, MemSum uses greedy selection with beam search over input sentences to create oracle-extracted summaries with high average ROUGE-1/2 scores. As shown previously in Table 1.1, the ROUGE-1/2 scores achieved by the oracle extractive summarization method significantly outperform those of LongT5-XL for abstractive summarization. Therefore, we employ the beam search approach in all our experiments that require obtaining oracle-extracted summaries.

4.4 Evaluation

4.4.1 ROUGE

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score (Lin, 2004) is a widely adopted metric for evaluating the quality of model-generated summaries against human-written reference summaries. It measures the overlap of n-grams between the system-generated summary and the reference summaries, providing various scoring variants such as ROUGE-N which calculates the n-gram recall, and ROUGE-L for longest common subsequence similarity. We use ROUGE-1/2/L score as a measure to reflect the quality of both extractive and abstractive summarization.

4.4.2 F1 Oracle

For extractive summarization only, we design a metric named "F1 oracle" that measures the agreement of selected sentences between extractive summarization models and oracle extractive summarization. The motivation is that extractive summarization systems are often trained on oracle extractions, while the task itself can be formulated as simple as a binary sentence classification task. Therefore, we can calculate the precision and recall between the model and oracle-extracted sentences to gain insights into how well the model is trained to match the oracle selection process as accurately as possible. Formally, for a model-generated extractive summary consisting of m input sentences and a reference summary containing n input sentences, if we have k matching sentences,

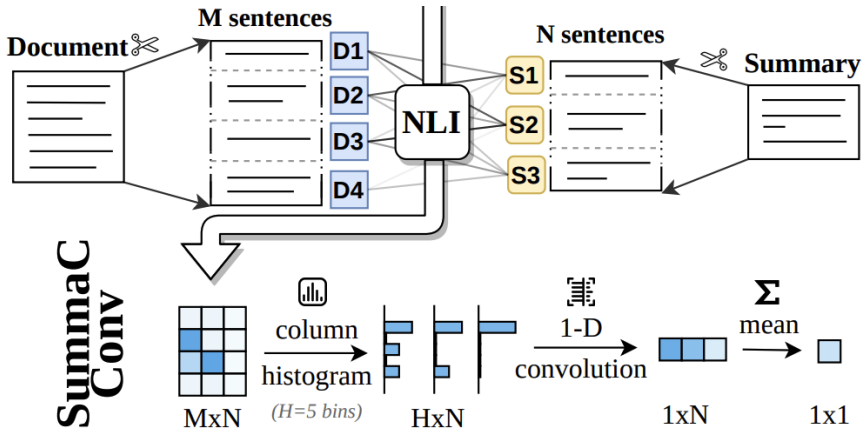

Figure 4.2: Overview of the SummaCConv entailment copied from (Laban et al., 2022). It aggregates all sentence-to-sentence entailment scores calculated using an NLI model.

then we can calculate the precision, recall, and F1 as follows:

$$Precision = \frac{k}{m} \qquad Recall = \frac{k}{n} \qquad F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.1}$$

4.4.3 Entailment

The two metrics discussed previously generally focused on content matching against the reference from a lexical perspective. However, for the task of abstractive summarization, it is also crucial to ensure that the generated summary is faithful to the input document. This means that the summaries should ideally not only capture the relevant content but also avoid hallucinating the meaning and implications of the original texts. Therefore, we employ several faithfulness metrics to evaluate factual consistency in the generated abstractive summaries.

The first faithfulness metric we use is entailment, which refers to the process of checking the logical relationship between two statements where one statement logically implies the other. In the context of summarization, the entailment metric evaluates whether the generated summary is logically entailed by the original document. A high entailment score generally indicates that the summary preserves the key facts and implications of the input text without introducing contradictory information. We employ entailment in evaluating abstractive summaries based on previous research suggesting its high correlation with human judgments (Maynez et al., 2020).

In practice, entailment is usually calculated at the sentence level, as internal reasoning when comparing whole passages and documents can be very complex. Sentence-level computation is also ideal for our primary task of long document summarization, as comparing documents poses high context length requirements for natural language inference (NLI) models. Formally, given a summary consisting of m sentences and an input document with n sentences, an entailment score is calculated for each summary sentence against each input sentence using an NLI model. The final entailment metric

Context	
The 2024 Summer Olympics is an upcoming sports event scheduled to take place from 26 July to 11 August 2024 in France, with Paris as the main host city.	
Summary	**Entailed**
Paris hosts Olympics 2024.	✓
The 2024 Olympics will feature new sports.	✗
France organizes the 2024 Summer Olympics.	✓

Table 4.3: Example of what summaries are entailed/not entailed by the given context.

is calculated by aggregating these scores, with the simplest approach being to take the maximum entailment score for each summary sentence across all input sentences, and then take the average of all the maximum entailment scores:

$$\text{Entailment} = \frac{1}{m}\sum_{i=1}^{m}\max_{j=1}^{n}\text{NLI}(s_i, d_j) \tag{4.2}$$

In this work, we consider another aggregation method named SummaCConv (Laban et al., 2022), which has superior performance compared to the simple approach outlined in Equation 4.2. An overview of the SummaCConv method is shown in Figure 4.2. The SummaCConv method utilizes a learned convolutional layer to aggregate the entailment scores between each pair of document and summary sentences. First, it bins the entailment scores for each summary sentence into a fixed-size histogram representing the distribution of scores. Next, a 1D convolutional layer with a kernel size equal to the number of bins is applied to each histogram, compiling it into a single score for that summary sentence. Finally, the scores for all summary sentences are averaged to obtain the overall inconsistency score for the summary. By considering the full distribution of entailment scores instead of just the maximum, SummaCConv is able to make more robust predictions compared to the maximum aggregation approach.

4.4.4 FactScore

FactScore (Min et al., 2023) is a recently introduced faithfulness metric that breaks down passages into phrase-level facts and measures whether each fact is supported by a reliable external knowledge source constructed from Wikipedia. An example of how a passage is broken down into atomic facts is shown in Figure 4.3, where each atomic fact can be viewed as a single unit of knowledge about the passage. Given these atomic facts, The FactScore metric then checks whether each atomic fact is supported by the external knowledge source and outputs the proportion of facts that pass the check. Formally, for a passage p with facts A_p, the FactScore of the passage is calculated as:

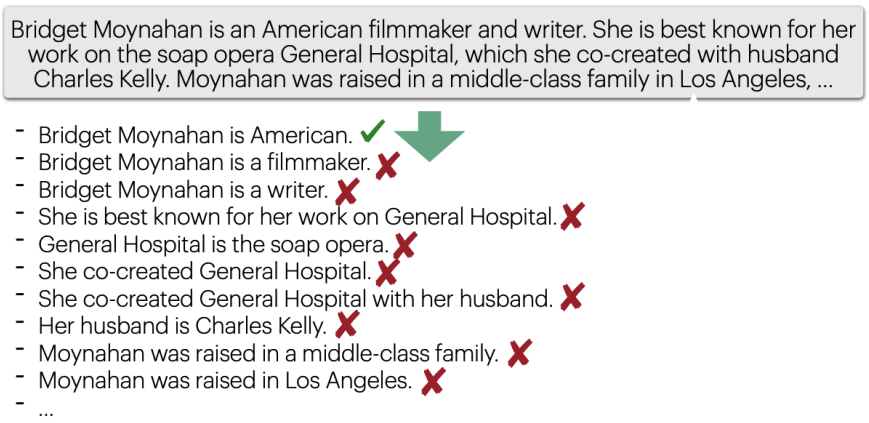

Figure 4.3: Example of how FactScore breaks down a biography to atomic facts copied from (Min et al., 2023). All atomic facts are then checked against a knowledge source to determine which facts generated are factually incorrect.

$$f(p) = \frac{1}{|A_p|} \sum_{a \in A_y} \mathbb{I}[\text{knowledge source supports a}] \tag{4.3}$$

Min et al. (2023) used LLMs to both generate and check atomic facts and found that ChatGPT and GPT-4 perform better at these tasks. One limitation of FactScore is that the original paper designed it to be a metric catered specifically to evaluating models' abilities in generating biographies with input format as *"Tell me a bio of <entity>"*. In this task, no ground truth generation is needed for evaluation as checking atomic facts only use relevant information in the knowledge source. Despite the task mismatch, the idea of the approach for faithfulness evaluation remains useful, as atomic facts offer finer granularity compared to sentences in entailment.

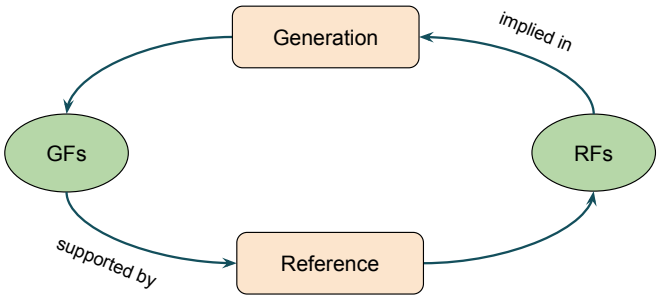

Figure 4.4: Example of the adapted FactScore metric for summarization. GF and RF are shorthands for Generated Fact and Reference Fact, respectively.

We now describe how we adapt the FactScore metric to the long document summarization task. An overview of the adaptation process is illustrated in Figure 4.4. Given a generated summary and its reference, we first break them down to generation facts (GFs) and reference facts (RFs) following the original FactScore approach. For each GF, we check whether it is supported by the concatenated RFs and output the proportion of GFs that pass the check as FactScore *precision*. Similarly, for each RF, we check

whether it is implied in the concatenated GFs and output the proportion of RFs that pass the check as FactScore *recall*. Together, the precision and recall reflect the accuracy and coverage of factual information in the generated summary with respect to the reference.

In practice, we run the adapted FactScore metric using GPT-3.5-turbo for both atomic fact generation and fact checking. We note that Min et al. (2023) originally used the text-davinci-003 model for atomic fact generation, which is now deprecated[6]. Therefore, we replace it with the GPT-3.5-turbo model. We format our prompts to be similar to the original FactScore prompts with minor changes for accurate adaptation. We use a temperature of 0.7 for all generations.

[6] `https://platform.openai.com/docs/deprecations/2023-07-06-gpt-and-embeddings`

Chapter 5

Results

5.1 Extractive Summarization

Table 5.1 shows the results of our proposed pipeline using the end-to-end LongT5-XL blueprint generation model and BM25 top-1 passages against baseline methods using MemSum and GPT-3.5-turbo extractive prompting introduced by Zhang et al. (2023). Compared to MemSum, our proposed pipeline performs worse on all ROUGE metrics except ROUGE-L on GovReport. The performance gap is most significant for ROUGE-2 on GovReport, where our proposed pipeline scores 3.4 points less than MemSum, while on other metrics, the performance gaps fluctuate around 1.0. Nevertheless, our proposed pipeline shows a significant increase over MemSum on the F1-oracle metric, suggesting a better ability at learning to follow oracle selections.

The reason for this performance gap can be attributed to our extractive summarization pipeline not being directly trained to maximize ROUGE scores over the summary. Our LongT5 model is fine-tuned end-to-end, considering only the blueprint but not the retrieval performance. Additionally, since the ground truth blueprints are also generated from the abstractive summary without considering extractive summarization, our blueprint generation model can be viewed as not being trained on objectives that maximize extractive summarization performance. Moreover, the pipeline setup using an end-to-end fine-tuned LongT5 may also cause sub-optimal blueprints to be generated. Narayan et al. (2023) showed in their results that the approach of first generating blueprints, then summaries conditioned on the blueprint only, has significantly inferior performance. The use of BM25 top-1 and short retrieval length might also negatively affect performance. Nevertheless, we demonstrated the potential of this simple pipeline by observing only a 1.0 lower ROUGE score on most metrics while even outperforming on ROUGE-L for GovReport. The higher F1-oracle score further highlights the proposed pipeline's capability to align with the oracle labels despite not being directly trained on these labels. Further discussion is provided in Section 5.3.

Compared to GPT-3.5-turbo with zero-shot prompting, we observe a significant gap between the proposed pipeline and GPT-3.5-turbo zero-shot on GovReport, with differences of 5.7, 6.6, and 2.4 in the ROUGE-1, ROUGE-2, and ROUGE-L metric, respectively. On the contrary, GPT-3.5-turbo performs best on SummScreenfd, achiev-

Methods	GovReport				SummScreenfd			
	R-1	**R-2**	**R-L**	**F1$_{oracle}$**	**R-1**	**R-2**	**R-L**	**F1$_{oracle}$**
MemSum	**58.3**	**28.5**	24.0	16.9	26.5	3.4	14.2	2.4
GPT-3.5-turbo (zero-shot)	51.4	18.5	22.3	-	**27.5**	**5.4**	**14.8**	-
LongT5 4096 + BM25 top-1	57.1	25.1	**24.7**	**24.6**	24.5	2.7	11.6	**7.5**

Table 5.1: Results for extractive summarization experiments. The F1-oracle metric is not reported for GPT-3.5-turbo, as the outputs have minor differences from the original document, making sentence/passage exact matching difficult to implement. We use zero-shot prompting for GPT-3.5-turbo and truncate the input to 4096 tokens to avoid outliers exceeding context length restrictions and to make a meaningful comparison with the proposed pipeline.

ing the highest scores on all metrics. This discrepancy in performance can potentially be explained by the nature of the two datasets. As described in Section 4.1, GovReport is formatted as articles written in formal language. This clean format allows fine-tuning approaches that are better at capturing keyword similarities to perform well on GovReport. In contrast, SummScreenfd consists of TV show transcripts containing mostly dialogues and scene descriptions, while the corresponding summaries focus more on capturing the abstract information conveyed in the transcripts. Because of this, GPT-3.5-turbo's stronger natural language understanding abilities can help to extract sentences that accurately capture abstract information.

The performance differences on the two datasets can be further analyzed to reveal key properties of extractive summarization systems. In Table 1.1, we see that the LongT5-XL abstractive summarization model achieves scores of 58.1, 27.3, and 29.7 on ROUGE-1, 2, L for GovReport, and 31.4, 6.7, and 18.8 for SummScreenfd, respectively. Comparing these results to the MemSum baseline and the proposed pipeline, we can observe that the ROUGE score difference on GovReport is relatively small, with roughly similar ROUGE-1 and ROUGE-2 but higher ROUGE-L. Nevertheless, the performance gap on SummScreenfd becomes much more significant, where the abstractive model has ROUGE scores that are 6.9, 4.0, and 7.2 points higher compared to the proposed pipeline. Since summarization on SummScreenfd is often more abstract, requiring the capture of complex discourse information, the observation that extractive summarization methods perform less well on SummScreenfd suggests that current models are still struggling to demonstrate a high-level understanding of the input text. Current methods emphasize maximizing ROUGE scores, which focus on keyword recall, while future approaches should also aim to improve natural language understanding in extractive summarization models to generate summaries that are more comprehensive and faithful.

Method	GovReport			SummScreenfd		
	R-L	**SummaC (input/ext)**	**FactScore**	**R-L**	**SummaC (input/ext)**	**FactScore**
LongT5-XL 4096	**29.7**	68.8	78.1	**18.8**	36.9	**38.8**
MemSum ext + LongT5-XL beam fine-tune	28.9	63.5 / 41.4	**78.3**	16.6	36.5 / 22.2	30.1
Blueprint ext + LongT5-XL beam fine-tune	27.2	**68.9** / 51.3	74.9	16.3	**37.4** / 23.8	30.4
Zero-shot LLM						
GPT-3.5-turbo	21.8	-	-	13.7	-	-
GPT-3.5-turbo (blueprint ext)	21.7	-	-	15.2	-	-

Table 5.2: Results of abstractive summarization. *LongT5-XL 4096* is the end-to-end abstractive summarization model. For both MemSum and our proposed extractive summarization pipeline, we fine-tune LongT5-XL on oracle labels and perform extract-then-summarize. We also include zero-shot LLM results using GPT-3.5-turbo in two settings 1) utilizing the first 4096 tokens and 2) utilizing the blueprint-extracted passages. For brevity, we only report the ROUGE-L score. The SummaC scores compare the results to the input document truncated to the first 4096 tokens for all outcomes (**input**/ext), and for the extract-then-summarize approaches, we also calculate entailment against the extracted passages (input/**ext**).

5.2 Abstractive Summarization

Table 5.2 shows the results of our proposed method using blueprint-extracted passages, compared against the LongT5-XL abstractive summarization and extract-then-summarize using MemSum extracted passages baselines. Similar to the results of extractive summarization, we observe that our pipeline produces worse ROUGE-L scores compared to both LongT5-XL and MemSum extract-then-summarize. The performance gap between our pipeline and MemSum extract-then-summarize suggests that the quality of retrieved passages at the extractive summarization stage also influences the abstractive summarization stage. We also note that our extract-then-summarize results presented in Table 1.2 have ROUGE-L scores of 32.8 and 22.0 for GovReport and SummScreenfd, respectively, which are much higher than the results in Table 5.2 where the best performing LongT5-XL abstractive baseline only has ROUGE-L scores of 29.7 and 18.8 for GovReport and SummScreenfd, respectively. This highlights the importance of obtaining salient extracted passages, as they have a strong influence on abstractive summarization results.

In addition, we note that both extract-then-summarize methods produce lower ROUGE-L scores compared to the end-to-end LongT5 abstractive summarization model, with the biggest gap of 2 ROUGE-L scores on SummScreenfd. For our proposed pipeline, the reason for performance drop can be attributed to the fact that we use two end-to-end LongT5 models in our pipeline, which potentially accumulate errors in the output. This is consistent with the results of Narayan et al. (2023), where the 2-stage pipeline performs significantly worse than using a single LongT5 model to generate both the blueprint and summary. Nevertheless, the abstractive model still works reasonably well compared to the MemSum extract-then-summarize baseline, suggesting that current extractive summarization approaches cannot effectively find salient information for abstractive models to focus on.

In terms of the GPT-3.5-turbo zero-shot baseline, we observe that the ROUGE-L scores are significantly lower than the previous baselines and the proposed pipeline. This result is consistent with previous abstractive summarization evaluations on LLMs (Shaham et al., 2023), where even state-of-the-art LLMs like GPT-4 cannot surpass the performance of state-of-the-art fine-tuned results. Additionally, we also compare the zero-shot performance between using the first 4096 tokens and extracted passages with blueprint questions. The resulting ROUGE-L scores are similar for GovReport, while blueprint extraction has 1.5 higher ROUGE-L on SummScreenfd.

Despite worse results on the ROUGE-L metric for our proposed pipeline, the results on faithfulness metrics are competitive against both baseline models, with our proposed pipeline achieving the best scores on SummaC for both datasets and on FactScore for SummScreenfd. Regarding the SummaC metric, we first calculate the entailment of generated summaries against the first 4096 tokens of the input document, as our proposed pipeline uses this truncated input for the blueprint generation stage. For extract-then-summarize approaches, we also calculate entailment against the extracted passages to examine the influence of extractive summarization on abstractive summary. We observe that compared to the first 4096 tokens, the SummaC score of our proposed pipeline outperforms all baselines, with the most significant improvements seen over the MemSum extract-then-summarize approach, with an increase of 5.4 and 0.9 on GovReport and SummScreenfd, respectively. The performance gap is even bigger for SummaC calculated against extracted passages, where the gains are now 9.9 and 1.6. This shows that our proposed pipeline not only generates abstractive summaries that are better entailed by the input document but also helps the abstractive summaries to be more grounded on the extracted passages. The reasons for this performance increase can be attributed to the use of blueprints, which helps retrieval in locating factual information in the input, which provides stronger signals to the abstractive summarizer on what contents it should include.

In terms of the FactScore metric, our proposed pipeline slightly outperforms the MemSum extract-then-summarize on SummScreenfd but performs significantly worse on GovReport. The LongT5-XL abstractive baseline achieves the strongest results on FactScore, recording 38.8 on SummScreenfd compared to 30.4 for the blueprint extract-then-summarize and 78.1 on GovReport, which is close to the 78.3 of MemSum extract-then-summarize. These results highlight significant performance gaps: between the blueprint extract-then-summarize method and baselines for GovReport, and be-

tween the LongT5-XL abstractive baseline and the extract-then-summarize methods for SummScreenfd.

There are several takeaways from the FactScore results. First, the strong and consistent LongT5-XL abstractive baseline results suggest that extract-then-summarize methods have shortcomings in achieving fine-grained factual consistencies. Considering better results on entailment for the blueprint extract-then-summarize model, the reason for the worse results on extract-then-summarize methods can potentially be attributed to retrieved passages missing some important factual information (i.e. lower recall), as calculating entailment is similar to calculating precision where we check whether each summary sentence is entailed by the input document. This makes it possible to have high entailment even if the summary does not cover all important information. Second, looking specifically at the FactScore results for extract-then-summarize methods, we see that the blueprint approach performs slightly better than the MemSum on SummScreenfd, while being significantly worse on GovReport. The performance difference between the two datasets can potentially be explained by the differing writing styles in the input/output. SummScreenfd summaries are less straightforwardly inferred from the input, making blueprints potentially helpful for the abstractive summarizer to focus on specific factual information in TV show transcripts. In contrast, GovReport's formal language may make information aggregation and reasoning easier, diminishing the advantage of using blueprints. Hence, the MemSum extract-then-summarize approach, with better extractive summarization performance, may benefit more from GovReport's formal writing style, making factual content more extractable.

5.3 Oracle Experiments

Despite the slightly worse performance of the proposed pipeline on extractive summarization, we do note that the whole process can potentially be improved to achieve better performance. Similar to what we did in the Introduction section, where we established upper bounds of the extractive summarization task on long document summarization, we extend the proposed pipeline with simple oracle approaches to explore possible performance improvements.

5.3.1 Blueprint Generation

For the blueprint generation stage, one potential performance bottleneck is that the LongT5 model is only fine-tuned with a maximum of 4096 input tokens. Better performance may be achieved by considering the entire input, which could enable more important aspects to be considered. Drawing inspiration from the hierarchical merging method used for book-length summarization (Chang et al., 2024), we use the blueprint generation model to generate a blueprint for each 4096-token chunk in the input. This yields much more blueprint QA pairs than the first 4096-token approach, but the concatenation of the retrieved passages would also result in more tokens for extractive summaries, making evaluation unfair. We thus perform greedy selection with beam search on the retrieved passages, essentially calculating a recall of oracle extractions to see whether the *generate for each* approach can capture more salient

Method	GovReport			SummScreenfd		
	R-1	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
Generate for each 4096 chunk	**68.9**	**35.8**	**30.5**	**35.9**	5.5	15.9
Oracle blueprint + BM25 top-1	62.3	34.1	30.2	33.6	**6.0**	**16.5**

Table 5.3: Results of the *generate for each* and the ground truth extractive summarization methods.

information than only using the first 4096 tokens. The retrieval method remains to be BM25 top-1.

Our results are shown in Table 5.3. We compare this approach to the extractive summarization performance using BM25 top-1 retrieval over the ground truth blueprints. Interestingly, we observe that the *generation for each* approach outperforms the ground truth on all ROUGE metrics for GovReport, with significant gaps on ROUGE-1, while the performance on SummScreenfd is also competitive. Relating to the background study described in Section 2.1, we argue the importance of including more context length beyond the 4096 tokens used by Narayan et al. (2023). In addition, the *generation for each* method could be viewed as performing an intermediate filtering that focuses more on salient passages, which potentially suggests that extractive summarization approaches can be applied to this filtered set of passages to reduce the noise of less relevant information.

5.3.2 Retrieval

Method	**Recall against oracle**	
	GovReport	SummScreenfd
top-1	30.0	10.4
top-10	80.3	47.6

Table 5.4: Recall against oracle for BM25 top-1 and top-10 on GovReport and SummScreenfd.

For the retrieval stage, we utilize the top-1 passage for each blueprint question to ensure a sensible concatenated passage length for evaluation. Nevertheless, previous work has demonstrated the inaccuracy of retrieval when using a small number of top-k passages for downstream tasks (Karpukhin et al., 2020b). Therefore, we conduct a simple experiment by considering the top-10 passages retrieved using BM25 and calculate their recall against the oracle, using a similar calculation as described in Section 4.4.2.

Our results are shown in Table 5.4. Compared to BM25 top-1, top-10 passages have a drastically higher oracle recall with 50.3 and 37.2 gaps for GovReport and SummScreenfd, respectively. This suggests potential improvements to the proposed pipeline

by using more generous retrieval methods aligned with the extractive summarization objective, with carefully selected procedures to locate the best passage related to a single blueprint question. In addition to retrieval methods, the retrieval unit length could also be an influential factor as Xu et al. (2024) used 300 word length passages as retrieval targets (compared to 40 and 20 words for GovReport and SummScreenfd respectively in this work). Even though longer passage lengths make evaluation on extractive summarization not meaningful, the concatenated passages could instead offer improvements to the extract-then-summarize stage by providing the abstractive summarizer with more information.

5.4 Empirical Analysis

In Table 5.5, we present examples of summaries generated by the MemSum and Blueprint extract-then-summarize methods for a TV show transcript from the SummScreenfd dataset, along with the generated blueprint QA pairs and the ground truth summary. Inconsistencies in the generated summaries and QA pairs are highlighted in red. Upon comparison of the two generated summaries, it becomes evident that the MemSum extract-then-summarize method produces a summary that is almost entirely unfaithful, with only one sentence consistent with the information in the input document. Upon closer examination, we note that the summary heavily focuses on the character *Professor Spafford*, who plays a minor role largely irrelevant to the main plot of the transcript which revolves around Joey's party. Conversely, the blueprint extract-then-summarize summary exhibits fewer factual inconsistencies, capturing the gist of a crucial scene where Rachel learnt about Joey's party and Joey's initial reluctance to have his friends attend due to fear of them looking silly at the party.

The factual inconsistencies in the blueprint extract-then-summarize summary may be attributed to errors in the blueprint QA pairs, as they contain incorrect information about characters Ross and Charlie. Despite these errors, the QA pairs successfully captured salient information regarding Rachel discovering Joey's secret party, which likely contributed to the summary accurately describing this scene. Additionally, it is worth noting that the blueprint extract-then-summarize summary includes information about characters Chandler and Phoebe, which are not mentioned in the blueprint QA pairs but are present in the retrieved passages. This discrepancy suggests that retrieved passages may contain noise that deviates the focus of the QA pairs.

Reference Summary	While Rachel is in the same position as Joey was a year ago, Joey holds a Days of our Lives cast party, but does not want any of his friends to attend. Ross gets a crush on another professor, Charlie (Aisha Tyler), who has only ever dated ”Nobel Prize” winners. He and Rachel catch her and Joey kissing at the party.
MemSum ext Summary	Joey is unable to get Monica to attend his Days of Our Lives party on the roof of the building, but he promises to get her there if Monica will only agree to pose as a fake ”British accent” for a guy who is in a cult. Monica is soon in over her head and it costs her the friendship of one of the guys in the cult. Ross and Charlie find a very sweet and kind Professor Spafford (whom they later see in a restaurant) and his wife, who is very knowledgeable about science and won the Nobel Prize for his work. When they go on a cruise to the Galapagos, Ross and Charlie meet a very different Professor Spafford, who is very different in his thinking and who makes absolutely no sense at all. The two disagree with his teaching methods and refuse to accept his wife's findings. Ross and Charlie go to a restaurant to get away from all the women, and find a very masculine Professor Spafford. He refuses to believe anything that Rachel has to say, and storms out in anger. Rachel, realizing that he is not the only one in a cult, breaks up with the very handsome Professor Spafford.
Blueprint ext Summary	Chandler dates a woman who intimidates him by being a soap opera fanatic, despite his protests that he doesn't like them. Rachel goes to a rooftop party with Joey, who invites her to meet some friends of his from work. Joey is uncomfortable around his friends who are also his co-workers. Phoebe has trouble connecting with her new boss, who is also her ex-boyfriend. Ross meets a guy who flirts with him, but intimidates him by being a Nobel Prize laureate.
Blueprint QA pairs	**Q**: Who is shocked to see that Charlie Wealer is the new professor in his department **A**: Ross **Q**: What does Ross have to deal with by visiting a restaurant containing nuts **A**: his allergy **Q**: Who secretly hosts a party for the Days of Our Lives crew **A**: Joey **Q**: Where does Joey host his party **A**: the roof **Q**: When did Rachel discover that Joey was hosting a party for the Days of Our Lives crew **A**: the last minute **Q**: Who discovers that Joey was hosting a party for the Days of Our Lives crew **A**: Rachel **Q**: What does Rachel accidentally discover that Joey was hosting **A**: the party **Q**: Where does Rachel go to apologize to Joey **A**: the house

Table 5.5: Examples of MemSum and Blueprint extract-then-summarize abstractive summaries from the *Friends* TV show S9E20, along with generated blueprint QA pairs for the Blueprint extraction approach. Due to space constraints, we omit the retrieved passages. Summary information and blueprint QA pairs inconsistent with the input are highlighted in red.

Chapter 6

Conclusions

In this work, we developed a pipeline for both extractive and abstractive summarization. Initially, the pipeline generates blueprints from the input document. It then retrieves relevant input passages based on the blueprint questions to create an extractive summary. Finally, the selected passages are used to compose an abstractive summary using an oracle fine-tuned model specifically designed for summarizing extracted passages. We conducted our experiments on the GovReport and SummScreenfd datasets, evaluating the results against state-of-the-art extractive and abstractive baselines, as well as zero-shot LLM approaches. Performance was assessed using various automatic metrics reflecting generation quality and faithfulness.

During our background study, we validated the idea of using extractive summarization for long documents by establishing performance upper bounds using oracle extractive summarization methods. We further explored whether such oracle methods could be applied to model fine-tuning to enable better task performance in summarization over extracted inputs. These observations supported our hypotheses that prioritizing the processing of salient information in the input could potentially lead to performance gains in the long document summarization task.

Our results demonstrate that employing a simple setup utilizing end-to-end models for both blueprint and summary generation, along with BM25 retrieval, can achieve competitive faithfulness performance in terms of entailment compared to LongT5-XL and MemSum extract-then-summarize. However, our approach exhibits inferior performance in terms of generation quality as measured by ROUGE scores, while also falling behind on FactScore, where significant performance gaps exist compared to the LongT5-XL baseline. We attribute the strong showing on entailment to the intermediate generation of blueprints, which capture factual information from the input, thereby enabling a more grounded abstractive summarization process. Additionally, we discuss reasons for the lower generation quality, which can be explained by the drawbacks of using two separate end-to-end models and the simplicity of our setup in fine-tuning and retrieval methods, and lower FactScore due to incompleteness in the retrieved passages used for abstractive summarization. Nevertheless, our oracle experiments show that our proposed approach can generate summaries that capture more aspects of the ground truth under oracle settings, showing promise for future improvements

to the pipeline. Furthermore, we note that our proposed pipeline, with intermediate blueprints, is more interpretable than end-to-end abstractive and extract-then-summarize approaches, providing a list of factual information that can be viewed as a gist of the input to make summarization less opaque.

In the future, we aim to explore more sophisticated pipeline configurations to gain a better understanding of the extent to which blueprints and retrieval can enhance summary quality. Specifically, we plan to test alternate setups by increasing the retrieval unit length to assess whether extracting more information can improve abstractive summarization. Additionally, we intend to modify the blueprint generation and retrieval stages to consider extractive summarization objectives, which would enable us to obtain higher-quality retrieved passages. By conducting more in-depth studies of the process, we aim to develop more optimal pipelines capable of achieving competitive generation quality while maintaining fidelity compared to state-of-the-art baselines.

Bibliography

Alberti, C., Andor, D., Pitler, E., Devlin, J., and Collins, M. (2019). Synthetic QA corpora generation with roundtrip consistency. In Korhonen, A., Traum, D., and Màrquez, L., editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., and Li, J. (2023). Longbench: A bilingual, multitask benchmark for long context understanding.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer.

Chang, Y., Lo, K., Goyal, T., and Iyyer, M. (2024). Booookscore: A systematic exploration of book-length summarization in the era of llms.

Chen, M., Chu, Z., Wiseman, S., and Gimpel, K. (2022). SummScreen: A dataset for abstractive screenplay summarization. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8602–8615, Dublin, Ireland. Association for Computational Linguistics.

Chen, S., Wong, S., Chen, L., and Tian, Y. (2023). Extending context window of large language models via positional interpolation.

Chen, Y., Qian, S., Tang, H., Lai, X., Liu, Z., Han, S., and Jia, J. (2024). Longlora: Efficient fine-tuning of long-context large language models.

Cheng, J. and Lapata, M. (2016). Neural summarization by extracting sentences and words. In Erk, K. and Smith, N. A., editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Cui, P., Hu, L., and Liu, Y. (2020). Enhancing extractive text summarization with topic-aware graph neural networks. In Scott, D., Bel, N., and Zong, C., editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5360–5371, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J.,

Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Erkan, G. and Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Fu, Y., Panda, R., Niu, X., Yue, X., Hajishirzi, H., Kim, Y., and Peng, H. (2024). Data engineering for scaling language models to 128k context.

Grenander, M., Dong, Y., Cheung, J. C. K., and Louis, A. (2019). Countering the effects of lead bias in news summarization via multi-stage training and auxiliary losses. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6019–6024, Hong Kong, China. Association for Computational Linguistics.

Gu, N., Ash, E., and Hahnloser, R. (2022). MemSum: Extractive summarization of long documents using multi-step episodic Markov decision processes. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6507–6522, Dublin, Ireland. Association for Computational Linguistics.

Guo, M., Ainslie, J., Uthus, D., Ontanon, S., Ni, J., Sung, Y.-H., and Yang, Y. (2022). LongT5: Efficient text-to-text transformer for long sequences. In Carpuat, M., de Marneffe, M.-C., and Meza Ruiz, I. V., editors, *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736, Seattle, United States. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A. (2020). spacy: Industrial-strength natural language processing in python. *Zenodo*.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models.

Huang, L., Cao, S., Parulian, N., Ji, H., and Wang, L. (2021). Efficient attentions for long document summarization. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y., editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020a). Dense passage retrieval for open-domain question answering. In Webber, B., Cohn, T., He, Y., and Liu, Y., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and tau Yih, W. (2020b). Dense passage retrieval for open-domain question answering.

Kedzie, C., McKeown, K., and Daumé III, H. (2018). Content selection in deep learning models of summarization. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J., editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1818–1828, Brussels, Belgium. Association for Computational Linguistics.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

Laban, P., Schnabel, T., Bennett, P. N., and Hearst, M. A. (2022). SummaC: Re-visiting NLI-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, 10:163–177.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

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

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. (2023). Lost in the middle: How language models use long contexts.

Liu, Y. and Lapata, M. (2019). Text summarization with pretrained encoders. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Manakul, P. and Gales, M. (2021). Long-span summarization via local attention and content selection. In Zong, C., Xia, F., Li, W., and Navigli, R., editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6026–6041, Online. Association for Computational Linguistics.

Maynez, J., Narayan, S., Bohnet, B., and McDonald, R. (2020). On faithfulness and factuality in abstractive summarization. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Mihalcea, R. and Tarau, P. (2004). TextRank: Bringing order into text. In Lin, D. and Wu, D., editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Min, S., Krishna, K., Lyu, X., Lewis, M., Yih, W.-t., Koh, P., Iyyer, M., Zettlemoyer, L., and Hajishirzi, H. (2023). FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Nallapati, R., Zhai, F., and Zhou, B. (2016). Summarunner: A recurrent neural network based sequence model for extractive summarization of documents.

Narayan, S., Maynez, J., Amplayo, R. K., Ganchev, K., Louis, A., Huot, F., Sandholm, A., Das, D., and Lapata, M. (2023). Conditional generation with a question-answering blueprint. *Transactions of the Association for Computational Linguistics*, 11:974–996.

Narayan, S., Zhao, Y., Maynez, J., Simoes, G., Nikolaev, V., and McDonald, R. (2021). Planning with learned entity prompts for abstractive summarization.

Puduppully, R., Dong, L., and Lapata, M. (2019). Data-to-text generation with content selection and planning.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv e-prints*, page arXiv:1606.05250.

Riester, A. (2019). Constructing qud trees. *Questions in Discourse*.

Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Shaham, U., Ivgi, M., Efrat, A., Berant, J., and Levy, O. (2023). Zeroscrolls: A zero-shot benchmark for long text understanding.

Shaham, U., Segal, E., Ivgi, M., Efrat, A., Yoran, O., Haviv, A., Gupta, A., Xiong, W., Geva, M., Berant, J., and Levy, O. (2022). SCROLLS: Standardized CompaRison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 12007–12021, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shazeer, N. M. and Stern, M. (2018). Adafactor: Adaptive learning rates with sublinear memory cost. *ArXiv*, abs/1804.04235.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. (2023). Roformer: Enhanced transformer with rotary position embedding.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023a). Llama: Open and efficient foundation language models.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023b). Llama 2: Open foundation and fine-tuned chat models.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2023). Chain-of-thought prompting elicits reasoning in large language models.

Xu, F., Shi, W., and Choi, E. (2023). Recomp: Improving retrieval-augmented lms with compression and selective augmentation.

Xu, P., Ping, W., Wu, X., McAfee, L., Zhu, C., Liu, Z., Subramanian, S., Bakhturina, E., Shoeybi, M., and Catanzaro, B. (2024). Retrieval meets long context large language models.

Xu, Y. and Lapata, M. (2022). Text summarization with oracle expectation.

Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. (2021). Big bird: Transformers for longer sequences.

Zhang, H., Liu, X., and Zhang, J. (2023). Extractive summarization via chatgpt for faithful summary generation.

Appendix A

Appendix

A.1 Dataset Examples

Table A.1 and A.2 provide one example input/output pair from GovReport and SummScreenfd datasets, respectively. The contexts and summaries are truncated to fit into one page where necessary.

Congress annually considers 12 regular appropriations bills for the fiscal year that begins on October 1. These bills together with other legislative measures providing appropriations known as supplemental and continuing appropriations (also referred to as continuing resolutions or CRs) provide annual appropriations for the agencies, projects, and activities funded therein. The annual appropriations cycle is often initiated after the President's budget submission. The House and Senate Appropriations Committees then hold hearings at which agencies provide further information and details about the President's budget. These hearings may be followed by congressional consideration of a budget resolution establishing a ceiling on overall spending within appropriations bills for the upcoming fiscal year. Committee and floor consideration of the annual appropriations bills occurs during the spring and summer months and may continue through the fall and winter until annual appropriations actions are completed. This report discusses FY2019 congressional appropriations actions and the impacts of the statutory budget enforcement framework established in the Budget Control Act of 2011 (BCA; P.L. 112-25) and the Bipartisan Budget Act of 2018 (BBA 2018; P.L. 115-123). It includes a chronological discussion and timeline (Figure 1) of these actions. FY2019 Appropriations and the Bipartisan Budget Act of 2018 FY2019 appropriations actions were impacted by the BCA, which placed statutory limits on spending for FY2012-FY2021, divided between defense and nondefense. In addition, the law created procedures that would automatically lower those caps if specified deficit-reducing legislation were not enacted. Congress has adjusted these statutory caps, including through the Bipartisan Budget Acts (BBAs) of 2013 (for FY2014 and FY2015), 2015 (for FY2016 and FY2017), 2018 (for FY2018 and FY2019), and 2019 (for FY2020 and FY2021), which provided for spending cap increases in both defense and nondefense categories. BBA 2018 capped FY2019 discretionary spending for defense at $647 billion and for nondefense at $597 billion. · · · · · · MORE CONTEXT · · · · · ·
Congress annually considers 12 regular appropriations measures to provide discretionary funding for federal government activities and operations. For FY2019, appropriations actions spanned two Congresses, between which there was a change in the majority party in the House. The process of drafting, considering, and enacting FY2019 appropriations began in early 2018 and included the House and Senate Appropriations Committees each marking up and reporting all 12 annual appropriations bills by the end of July. Five appropriations bills in the 115th Congress were enacted into law by the start of the fiscal year. An additional seven appropriations bills remained in various stages of consideration. Continuing resolutions (CRs) were enacted in order to extend funding of government operations covered in these seven bills. · · · · · · MORE SUMMARY · · · · · ·

Table A.1: Example of one truncated input/output pair in the GovReport dataset.

[Scene: Central Perk, Joey is talking to Ross] Joey: There's this woman, that I like. A lot. Well, it's complicated. She's with this other guy. For a long time. And I could never do that to the guy, y'know? 'Ccause we're really good friends. Ross: So, uh, this guy, she used to go out with, is, uh... is he a good guy? Joey: Yeah, he's the best. Ross: Then talk to him! He might be fine with it. Joey: Oh, I don't know. Ross: Joey, it's worth finding out. I mean, if you really like her. Joey: I do! So much! I can't stop thinking about her! I can't sleep, I- Ross: Okay, Joey, you know what? You have to go for it. How often does this happen to you, huh? You owe it to yourself. (Walks towards the door until...) Joey: It's Rachel. [Fade to Black, then fade in again with Ross stopped at the doorway.] Ross: (closes the door) Did you um-I'm sorry, did you just say it's Rachel? Joey: Yes. Ross: Um, you...you like Rachel? Joey: Yes. I like Rachel. Ross: Rachel?! Joey: (startled) Yeah, okay but look, buy uh-Hey-hey, y'know, y'know who else I like? You! And it-it doesn't get said enough. I like you Ross. Ross: But R-R-Rachel-Rachel?! Joey: Yeah, but it's not a big deal. Ross: It's not a big deal? Oh, I'm sorry I just...um, I...what about all the stuff you-you just said? I mean how about, I like-you-you can't stop thinking about her. Like how you can't sleep? Joey: I'm an actor, y'know? As-as a group, we tend to be over dramatic. Ross: Rachel who's carrying my baby? Rachel? Joey: Look no, I-I know it's bad, and I know it's wrong. Okay? But-but it's not like anything's ever gonna happen. Y'know? These-these are just feelings, they're gonna go away. Ross: Y'know what? I-I gotta go. (Starts to leave.) Joey: Oh come on Ross! Hey Ross-Ross don't... Ross: (stops) I just-y'know-I-I just have one-Rachel?! (He exits and starts to walk away, passes a window, stops, and says "Rachel?!" again. Joey sighs and turns around to face Gunther.) · · · · · · MORE CONTEXT · · · · · ·
Ross, shocked at Joey's declaration, avoids him, but eventually convinces him to tell Rachel. Joey confesses his love for Rachel, but Rachel politely and lovingly turns him down. Phoebe is convinced that a British man called Don is Monica's soulmate.

Table A.2: Example of one truncated input/output pair in the SummScreenfd dataset.