THE UNIVERSITY *of* EDINBURGH

Abstractive Summarization of Long Narratives through Content Selection and Model Scaling

Rohit Saxena

Doctor of Philosophy

Institute for Language, Cognition and Computation
CDT Natural Language Processing
School of Informatics
The University of Edinburgh

2025

Abstract

Abstractive summarization of long narrative texts, such as novels and movie screenplays, presents significant challenges due to their extensive length, complex structure, and the necessity of capturing essential narrative elements accurately. While large language models (LLMs) have demonstrated remarkable progress in text summarization, their ability to process long narratives remains limited due to computational constraints and the difficulty of extracting salient elements. This thesis addresses these challenges, particularly in movie screenplays, by focusing on two key issues: identifying salient scenes crucial to the overall story and handling the computational constraints inherent in processing lengthy texts.

First, we introduce MovieSum, a large-scale dataset specifically designed for movie screenplay summarization. MovieSum consists of 2,200 movie screenplays paired with their corresponding Wikipedia plot summaries and is manually formatted to represent structural screenplay elements. This dataset is significantly larger than existing resources and includes metadata such as IMDb IDs, enabling access to additional knowledge. We use MovieSum to benchmark recent LLMs and provide a baseline for future research in narrative summarization. Additionally, we introduce the Movie Scene Saliency Dataset (MENSA), a subset of MovieSum containing human-annotated salient scenes from 100 diverse movies.

Second, we investigate the role of scene saliency and saliency-based content selection in screenplay summarization. A movie screenplay consists of numerous scenes, but only a fraction of them contribute meaningfully to the overall story. We propose a two-stage summarization framework utilizing the MENSA dataset. The first stage identifies key scenes based on their relevance to the movie's narrative, while the second stage generates an abstractive summary using only these salient scenes. Our findings show that this approach outperforms existing state-of-the-art summarization methods, producing summaries that more accurately reflect the content of the narrative.

Finally, we address the computational limitations of transformer-based models in processing long documents, including movie screenplays. Existing models rely on truncation, which leads to information loss and inconsistencies between training and inference. To mitigate this, we propose CachED, a gradient caching technique that enables end-to-end training of encoder-decoder models on full-length documents without truncation. We apply CachED to extend BART, creating CachED-BART, which is capable of backpropagation on nearly one million tokens without additional parameter over-

head. Experimental results demonstrate that CachED-BART achieves superior performance on long-form summarization tasks, including movie screenplays and books, while maintaining efficiency and scalability.

This thesis advances the field of long-form narrative summarization by introducing structured datasets, scene-aware summarization techniques, and novel training methodologies. Our results highlight the importance of selecting salient narrative elements and leveraging efficient model architectures to generate accurate and coherent summaries for complex, lengthy texts such as movie screenplays. Through these contributions, this thesis aims to enhance the efficiency and accuracy of movie script summarization, while also providing valuable insights into overcoming the computational challenges associated with long-form narrative summarization.

Lay Summary

Abstractive summarization of long narratives is about taking a lengthy piece of text, such as a movie screenplay or novel, and turning it into a concise summary that captures the most important parts of the story. Traditionally, summarization systems attempt to compress the entire text into a shorter version, but they often struggle with very long narratives. These texts are rich in detail, filled with complex storylines and many characters, making it difficult for systems to determine what is truly important.

In this thesis, we focus on two key ideas to improve the summarization process: selecting the most meaningful parts of a narrative and improving how summarization models handle very long texts. First, we introduce a new dataset called MovieSum, which includes over 2,000 movie screenplays, each paired with its corresponding summary. This dataset supports the training and evaluation of models designed for long narrative summarization. In addition to the full dataset, we construct a specialized subset with human-annotated "salient" scenes, which are scenes considered essential for understanding the story, from 100 different movies. This annotation helps models learn to identify the most relevant parts of a narrative.

Next, we propose a two-step summarization approach. In the first step, the model uses the saliency annotations to select the most critical scenes, effectively compressing the narrative down to its core content. In the second step, the model generates a summary using only these selected scenes. This process ensures that the resulting summary reflects the key elements of the story while ignoring less important details.

We also address the technical challenge of processing very long documents with current models, which typically face memory and computation limits and often require the truncation of input text. To overcome this, we introduce a technique called CachED (Gradient Caching for Encoder-Decoder models). CachED allows smaller models to handle extremely long documents by breaking them into manageable chunks, processing each chunk separately, and then combining the results. This makes it possible to train models on narratives that are much longer than what traditional approaches can handle, without sacrificing performance.

By using content selection and scalable model training, this thesis shows that it is possible to generate fluent, accurate summaries from very long narratives. Our experiments demonstrate that these methods improve the quality of generated summaries, helping models to better capture the essential ideas of complex narratives.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my primary supervisor, Frank Keller, for his invaluable guidance, support, and encouragement throughout my PhD journey. Frank's insightful feedback, patience, and high standards have profoundly shaped the way I think about research and writing. I am truly grateful for the freedom he gave me to explore my ideas while always providing thoughtful direction whenever I needed it. Working with Frank has been an immense privilege, and I have learned from him not only how to do good research but also how to be a better scholar and mentor.

I would also like to sincerely thank my co-supervisors, Hao Tang and Pasquale Minervini, for their guidance, generosity, and inspiration. Hao has an incredible ability to simplify complex problems and make the most intricate technical details feel approachable and fun. Pasquale's endless curiosity, energy, and passion for research have been truly contagious. He is one of the most hardworking and humble researchers I have ever met, always approachable, full of ideas, and ready to engage hands-on in discussions and experiments. His enthusiasm has significantly shaped my research direction and continues to inspire me to push boundaries with curiosity and persistence.

I am also grateful to Edoardo M. Ponti and Mohit Iyyer for examining my thesis. Their thoughtful feedback and insightful discussions have greatly improved the quality of this work. I would also like to thank Mirella Lapata, Ivan Titov, and Adam Lopez for their guidance and support throughout my PhD journey.

I would like to thank Srikant for giving me the opportunity to intern at Amazon, and Rishi, Tejas, and Ananya for making me feel at home in Seattle. Those months remain some of my most memorable and formative experiences.

To my PhD friends, thank you for making this journey memorable and fulfilling. Rohan, for the many coffees, lunches, and your unfailingly positive attitude. Aryo, for being an amazing friend and co-author whose dedication and hard work have inspired me deeply. Henry, for bringing thoughtful linguistic and representational insights to our discussions. Georgia, for the fun office banter, from ECoG to cats. And Agostina, for being a wonderful conference buddy during many academic trips.

I was fortunate to share my office with Ronald, Jason, Irene, and Aida. Thank you all for making the workdays so enjoyable. I am also thankful to Nikita and Parag for our many dinners and PhD rants, which kept me sane through stressful times.

Thanks also to everyone at the CDT, forum, and to all members of the research

groups I was part of, in particular Amr, Mathias, Zheng, Tom H, Giwon, Yu, Arushi, Ruchika, Gautier, and Danyang. Special thanks to Sally for her countless help with administrative matters and for always greeting our queries with a smile. I also want to acknowledge the CSTR lunch group for the countless engaging conversations and for introducing me to so many wonderful people.

My sincere thanks also go to Niranjan, my former mentor, whose support and encouragement were instrumental in helping me pursue a PhD. I also thank my former office friends Savita, Arun, Punit, Komal, and Prashant for their friendship and encouragement during the early days of my applications.

Finally, I would like to express my heartfelt gratitude to my family. To my parents, my late mother Meena, whose love and sacrifices continue to guide me every day, and my father Ashok. To my family, Mohit, Akanksha, and little Meir, thank you for your constant love and support. And to my childhood friend Asit, thank you for always being there.

And most of all, to my wife Radhika, my pillar of strength, love, and patience. You have been with me through every challenge and triumph, offering unconditional support and grounding me when things seemed impossible. None of this would have been possible without you. Thank you for everything.

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.

(*Rohit Saxena)*

Contents

Chapter 1

Introduction

Narratives have been one of the fundamental aspects of human communication since ancient times. They serve as the primary means by which humans make sense of complex events, share experiences, and transmit cultural knowledge (Fisher, 1989; Bruner, 1991). From oral storytelling traditions and classical literature to contemporary media such as movies, TV shows, and video games, narratives help humans interpret the world and build communal identities.

In recent years, the volume of digital narrative content has vastly increased. Digital platforms and streaming services offer catalogs comprising thousands of movies and TV episodes, while online repositories host extensive archives of novels. For example, streaming services for movies/TV shows like Netflix provide access to tens of thousands of titles and continue to grow exponentially. This surge has led to information overload, leaving users inundated with data and often struggling to decide which movie or show to watch, and which book or story to read.

Narrative summarization can help alleviate this overload by condensing long narratives into digestible forms. A well-crafted summary enables individuals to grasp the core events and themes of a narrative without the need to process every detail. Whether it is a plot synopsis for a movie, a brief recap of a TV series, or an overview of a lengthy book, summarization facilitates faster decision-making and more efficient consumption of narratives. For instance, with movies, users might read a textual summary (as found on platforms like Wikipedia or IMDb) to get an overview of the story, refresh their memory of content they have already seen, or catch up on earlier installments before watching new releases.

Beyond these applications, narrative summarization also poses unique challenges compared to other text genres, requiring models to capture temporal progression, causal

dependencies, character dynamics, and discourse structure. These aspects are central to human narrative comprehension, as emphasized in cognitive science and narratology (Kintsch and van Dijk, 1978; Graesser et al., 1994; Zwaan and Radvansky, 1998). Studying narrative summarization therefore not only addresses the problem of information overload but also provides an opportunity to test computational models against the cognitive and linguistic abilities humans rely on to understand stories.

The rapid growth of digital narrative content has made it increasingly challenging to produce high-quality human-generated summaries. As the volume of movies, TV shows, books, and other narrative formats continues to soar, manually creating concise and informative summaries for each item is not only time-consuming but also impractical. This is particularly difficult for narratives, where the essential story elements are often interwoven with less relevant details. Thus, automated narrative summarization especially for these long narratives, would greatly help in making them more accessible and can assist users in navigating vast narrative repositories. Ultimately, effective summarization serves as an interface between overwhelming content and the limited time and attention of users (Saxena et al., 2017).

Automating the process of summarization has garnered substantial interest in the field of Natural Language Processing (NLP). It has evolved over several decades, reflecting the changing capabilities of computational models and the shifting nature of available data. Jones (1998) defines summarization as a process of transforming a source text into a condensed version while retaining its essential meaning. A summary achieves this through content reduction by selection and/or generalization of key information from the source. The earliest approaches to summarization were predominantly extractive. Extractive methods retrieve text spans from the input and concatenate them to generate a summary. Early approaches used frequency-based methods or heuristic based methods for extractive summarization like word frequency and position (Luhn, 1958; Edmundson, 1969; Brandow et al., 1995). The emergence of advanced statistical and machine learning techniques summarization research began to incorporate more structured representations of text. Techniques such as Latent Semantic Analysis (LSA; Landauer et al., 1998) and graph-based methods emerged, aiming to capture the underlying semantic relationships within a document. Graph-based methods like LexRank (Erkan and Radev, 2004), TextRank (Mihalcea and Tarau, 2004), and Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) modeled documents as networks of sentences, using centrality measures to select the most salient content. In parallel, supervised learning methods were developed, with research-

ers framing summarization as a classification problem—identifying which sentences should be included in the summary based on features such as term frequency, sentence length, and position (Kupiec et al., 1995; Lin, 1999; Osborne, 2002). These approaches provided significant improvements over purely heuristic methods by leveraging statistical learning.

In contrast to extractive summarization, which selects and concatenates spans from the source, abstractive summarization generates novel text that may paraphrase, compress, and reorganize information while preserving meaning. Lately, the emergence of neural networks and sequence-to-sequence (seq2seq) models revolutionized summarization by enabling abstractive methods. Early neural network approaches utilized Recurrent Neural Networks (RNN; Mikolov et al., 2010; Hochreiter and Schmidhuber, 1997) and later, Transformer-based models (Vaswani et al., 2017) were introduced. Rush et al. (2015) introduced attention-based models for abstractive summarization, and subsequent developments such as pointer-generator networks (See et al., 2017) further improved these models by introducing a mechanism that explicitly allows the model to copy words directly from the source text. Neural models (Cheng and Lapata, 2016; Dong et al., 2019) demonstrated the ability to rephrase and compress information, but they were initially limited to short texts due to the computational constraints of processing long inputs. In particular, Transformer-based architectures (Vaswani et al., 2017) incur quadratic memory complexity with respect to input length, which makes scaling to long narratives prohibitively expensive and often forces truncation during training. More recenlty, the introduction of large-scale pre-trained models such as BERT (Devlin et al., 2019), BART (Lewis et al., 2020), and T5 (Raffel et al., 2020) has had a profound impact on summarization. These models, pre-trained on massive corpora and fine-tuned for summarization tasks, have achieved breakthrough results on several benchmarks. In the last few years, the field of NLP has seen the emergence of large language models (LLMs), which are powerful text-generation systems trained on the language modeling task (Touvron et al., 2023; Grattafiori et al., 2024; OpenAI et al., 2024). These models undergo further fine-tuning to follow instructions and align with human preferences, making them highly versatile and capable of performing various natural language tasks, including summarization.

Consequently, automating the process of narrative summarization has garnered substantial interest in the field of Natural Language Processing (Gorinski and Lapata, 2015; Chen et al., 2022). Narrative summarization not only addresses the practical need to manage information overload but also poses an intriguing research challenge. Narrat-

ives are inherently complex: they involve long-range dependencies between events, intricate interactions among characters, and often non-linear storylines that defy simple chronological order. To effectively summarize such content, computational models must be capable of discerning which elements of the narrative are truly pivotal, a property we refer to as narrative saliency. This concept goes beyond simple word frequency or sentence position; it requires models to capture causal relationships, identify important plot points, and understand the overall structure of a story (Papalampidi et al., 2020).

Despite the progress in summarizing short texts (e.g., news articles, product reviews) using neural and large language models, narrative summarization poses additional challenges. Long-form narratives often exceed tens or even hundreds of thousands of tokens, scattering crucial plot developments throughout. Moreover, their rich structure includes characters, events, flashbacks, and subplots, all of which demand careful attention. Traditional Transformer architectures (Vaswani et al., 2017) exhibit quadratic time and memory complexity in the input length, limiting their context windows. This practical constraint leads to truncation of input, causing potential loss of essential information and a mismatch between training and inference conditions (Liu et al., 2023). Overcoming these scalability hurdles while maintaining narrative coherence and factual consistency remains a core challenge.

1.1 Challenges in Summarizing Long Narratives

Narrative texts, such as movie scripts and entire books, bring into focus three key challenges:

High quality Datasets: A significant challenge in long narrative summarization is the variability and noise inherent in the available data. Narrative sources, such as film scripts, TV show transcripts, and literary texts, often exhibit inconsistent formatting, missing context, and irregular annotation quality, which complicates the extraction of meaningful information. Unlike curated news articles, narratives are produced by diverse creators and may contain idiosyncratic conventions, informal language, and errors introduced during digitization or transcription. Furthermore, reference summaries used for training are frequently generated through crowd-sourcing or semi-automatic methods, resulting in inconsistencies in style, length, and content coverage. These data quality issues can hinder a model's ability to learn robust representations of narrative structure and to reliably identify salient content.

Content Selection and Saliency: With extensive input texts, it becomes crucial to determine which parts of the narrative are most important. Saliency in narratives is not solely a matter of frequency or position; it involves understanding the structure (scenes), causal relationships, and character dynamics. Effective summarization requires models to identify and focus on these salient spans of text, a task that is challenging when relevant content is interspersed throughout the text.

Scale and Model Constraints: Standard Transformer architectures (Vaswani et al., 2017) have quadratic time and memory complexity with respect to input length. As a result, models are often forced to truncate inputs (typically to 512 or 1024 tokens), leading to the loss of critical context. Long-form narrative exceeds these context length windows substantially, leading to model truncation and a discrepancy between training and inference. Scalable architectures that preserve comprehensive context end-to-end are thus critical. Models such as Longformer (Beltagy et al., 2020), BigBird (Zaheer et al., 2020a), and Linformer (Wang et al., 2020) adapt the sparse-attention mechanism to handle larger input windows. However, they often require specialized pretraining and still face upper bounds on length. Empirical results also show performance drops as input contexts grow extremely large (Ivgi et al., 2023a).

1.1.1 Qualities of Effective Summarization

A high-quality summary must satisfy several key qualities. First, it should be informative, covering the most salient content of the source text while omitting redundant or irrelevant details. Second, it must maintain factual consistency, ensuring that statements in the summary are entailed by or faithful to the source. Finally, it should exhibit coherence and fluency, presenting the information in a logically ordered and linguistically correct way.

1.2 Thesis Contributions

This thesis proposes new datasets, an efficient model, and content selection strategies to address the challenges of long-form narrative summarization. Our core contributions are summarized below:

1. **Movie Screenplay Datasets.** We introduce MovieSum and MENSA, two datasets designed for understanding and summarization movie narratives. MovieSum pairs manually formatted movie screenplays with Wikipedia plot

summaries, while MENSA provides human-labeled scene-to-summary alignments across 100 diverse movies. This enables the study of scene-level saliency, bridging the gap between raw screenplay elements and abstractive summaries.

2. **Scene Saliency and Abstractive Summarization.** Building on MENSA, we develop a supervised classifier to identify salient scenes in a given movie script. Incorporating these predicted salient scenes into an abstractive transformer-based model leads to significant performance gains on benchmark datasets such as ScriptBase (Gorinski and Lapata, 2015). The improvements in automatic metrics confirm that content selection substantially benefits long-form summarization.

3. **End-to-End Long Document Summarization.** We propose **CachED** (Caching for Encoder-Decoder models), a memory-efficient approach that allows full-length training without truncation. By splitting the input into chunks and caching gradients at the decoder level, we reduce the peak memory footprint. This permits scaling to extremely long inputs (over 100K tokens) and specifically removes the need to truncate sequences, closing the train-test gap. Our experiments on BookSum, MENSA, and additional long-document benchmarks illustrate that fully utilizing the entire text significantly improves summary coverage, coherence, and factual alignment.

1.3 Thesis Outline.

- **Chapter 2 (Background).** We present the theoretical and technical foundations that underlie this thesis. This chapter reviews the Transformer architectures and its self-attention mechanism (including its quadratic complexity), details encoder-decoder models (with a focus on cross-attention and context length limits in the models), surveys efficient attention methods, and describes fusion-in-decoder and chunk-based strategies. We also review previous work on the salience in summarization, including narrative salience, and discuss key evaluation metrics.

- **Chapter 3 (Movie and Scene Saliency Dataset: Creation and Benchmarking).** We present the motivations behind MovieSum and MENSA, describe the annotation protocols, and analyze their complexity. We benchmark a diverse range of summarization models available at the time of the experiments on MovieSum, highlighting the difficulties inherent to movie script summarization.

- **Chapter 4 (Content Selection Using Scene Saliency for Summarization).** We propose a supervised saliency classification model trained on MENSA, and integrate it into a two-stage framework for abstractive summarization. We demonstrate state-of-the-art performance on large-scale screenplay corpora, validating scene-level saliency as a strong content selection mechanism.
- **Chapter 5 (Efficient Long-Document Summarization).** We describe our gradient caching approach that facilitates end-to-end training with very long inputs. We evaluate CachED-BART on multiple datasets, including book-level summarization, showing that directly encoding entire documents yields more coherent and faithful summaries.
- **Chapter 6 (Conclusions and Future Work).** We summarize the contributions of this thesis, discussing how explicit scene saliency and memory-efficient full-text training can advance long-form narrative summarization. We also outline potential extensions to multi-modal tasks and personalised summarization.

The publications included in this thesis are:

- Rohit Saxena and Frank Keller. 2024. Select and Summarize: Scene Saliency for Movie Script Summarization. In Findings of the Association for Computational Linguistics: NAACL 2024, pages 3439–3455, Mexico City, Mexico. Association for Computational Linguistics.
- Rohit Saxena and Frank Keller. 2024. MovieSum: An Abstractive Summarization Dataset for Movie Screenplays. In Findings of the Association for Computational Linguistics: ACL 2024, pages 4043–4050, Bangkok, Thailand. Association for Computational Linguistics.
- Rohit Saxena, Hao Tang, and Frank Keller. 2025. End-to-End Long Document Summarization using Gradient Caching. In Transactions of the Association for Computational Linguistics (TACL).

The papers, in addition to those included in the thesis, are as follows.

- Rohit Saxena, Pasquale Minervini, and Frank Keller. 2025. PosterSum: A Multimodal Benchmark for Scientific Poster Summarization. NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling.

- Rohit Saxena, Aryo Pradipta Gema, Pasquale Minervini. 2025. Lost in Time: Clock and Calendar Understanding Challenges in Multimodal LLMs. Reasoning and Planning for LLMs Workshop at The Thirteenth International Conference on Learning Representations.

Chapter 2

Background

In this chapter, we provide a brief overview of the background necessary to understand the methods developed in this thesis. We begin with an overview of summarization models starting with Transformer architectures, focusing on the self-attention mechanism and its inherent quadratic complexity. We then discuss encoder-decoder models and their application to text summarization. Subsequently, we discuss various approaches to making attention more efficient, including sparse attention and its variants, followed by a discussion of fusion-in-decoder and chunk-based strategies. We also describe prior work on salience in summarization, narrative saliency in particular, and provide a review of evaluation metrics used in summarization research.

2.1 Transformer Architectures and Self-Attention

To understand the models and methods explored in this thesis, we first examine the Transformer architecture. Transformers (Vaswani et al., 2017), a neural network architecture, have dominated the field of modern Natural Language Processing due to their ability to model long-range dependencies through self-attention. In contrast to previous recurrent networks, Transformers process all the tokens in parallel, enabling efficient training on GPUs. While Transformers dominate, recent State Space Models (SSMs; Gu et al., 2022) offer parallelizable sequence processing during training. We focus on Transformer-based summarization in this thesis, while noting that State Space Models represent an alternative line of work for long-context modeling.

2.1.1 Self-Attention Mechanism

Given an input sequence of n tokens with corresponding embeddings $X \in \mathbb{R}^{n \times d}$, the self-attention mechanism computes three matrices: queries Q, keys K, and values V, obtained via learned projections:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V, \tag{2.1}$$

where $W^Q, W^K, W^V \in \mathbb{R}^{d \times d_k}$ are projection matrices that map the input embeddings of dimension d into a space of size d_k. The attention output is given by:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \tag{2.2}$$

Here, each element α_{ij} of the attention matrix represents the weight assigned to token j when processing token i. The softmax normalization ensures the weights sum to one for each token.

A major drawback of standard self-attention is its quadratic complexity with respect to the sequence length n. Computing $QK^\top$ requires $O(n^2 d_k)$ operations and memory, which becomes a bottleneck for long inputs. This constraint caps the maximum sequence length that can be handled efficiently, both during training and during generation, motivating a wide range of architectural variants and approximations in long-document processing.

While a single attention map captures one similarity pattern, many tasks require attending to multiple relationships simultaneously. This motivates the use of multi-head attention, described in the next subsection.

2.1.2 Multi-Head Attention and Transformer Layers

Transformers use multi-head attention to allow the model to focus on different parts of the input simultaneously. Instead of a single attention computation, the model computes h attention heads:

$$\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V), \quad i = 1, \ldots, h, \tag{2.3}$$

and then concatenates their outputs:

$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O. \tag{2.4}$$

Each Transformer layer consists of a multi-head self-attention (sub-layer), followed by a position-wise feed-forward network (FFN) with a non-linear activation function (e.g.,

ReLU (Glorot et al., 2011), GeLU (Hendrycks and Gimpel, 2023)):

$$\text{FeedForward}(X) = \sigma(XW_1 + \mathbf{b}_1)W_2 + \mathbf{b}_2, \tag{2.5}$$

where σ denotes the activation function, $W^O \in \mathbb{R}^{(hd_k)\times d}$, $W_1 \in \mathbb{R}^{d\times d_f}$, $W_2 \in \mathbb{R}^{d_f\times d}$, $\mathbf{b}_1 \in \mathbb{R}^{d_f}$, and $\mathbf{b}_2 \in \mathbb{R}^d$. Residual connections and layer normalization are applied around each multi-head attention and feed-forward sub-layer.

Having established the core Transformer components, we next consider how these architectures are used in encoder–decoder models for text generation, particularly summarization.

2.2 Encoder-Decoder Models for Summarization

The encoder-decoder framework is central to many natural language understanding and summarization systems. It is composed of two Transformer stacks: an encoder that processes the source text to produce a sequence of contextualized representations and a decoder that generates the target summary conditioned on these representations, as shown in Figure 2.1.

2.2.1 Architecture Overview

Let $X = [x_1, x_2, \ldots, x_n]$ denote the source text and $Y = [y_1, y_2, \ldots, y_m]$ the target summary. The encoder computes contextual representations of the input:

$$Z = \text{Enc}(X) \in \mathbb{R}^{n\times d}, \tag{2.6}$$

and the decoder generates the summary autoregressively:

$$P(y_t \mid y_{<t}, Z) = \text{Dec}(y_{<t}, Z)\ . \tag{2.7}$$

During training, the model is typically optimized via teacher forcing, minimizing cross-entropy loss between the predicted and ground-truth tokens.

$$\mathcal{L}_{\text{CE}} = -\sum_{t=1}^{m} \log P(y_t \mid y_{<t}, Z). \tag{2.8}$$

The term autoregressive refers to the sequential process in which the decoder generates each token one at a time, with each token being conditioned on all the tokens generated previously. A key component of the decoder is **cross-attention**, which enables the

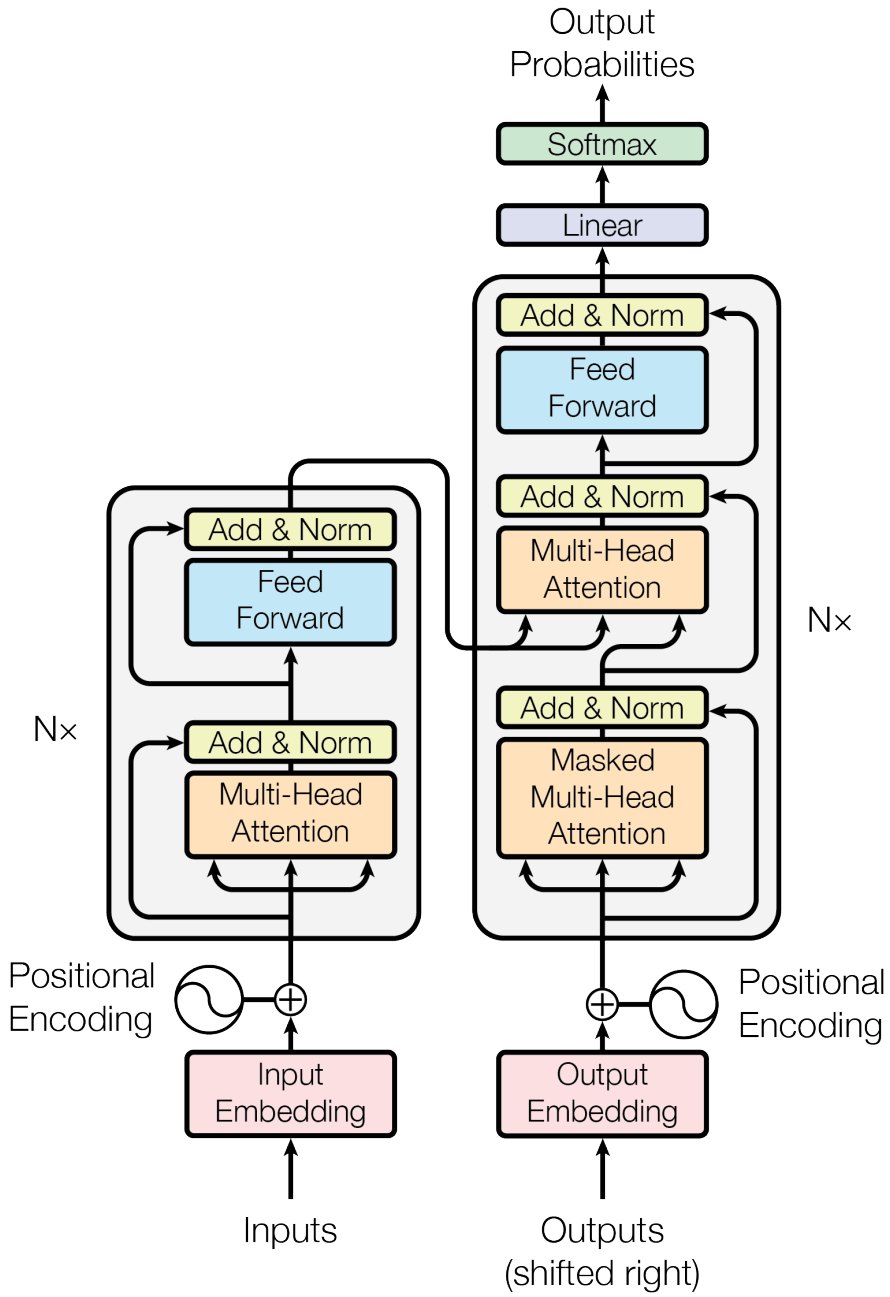
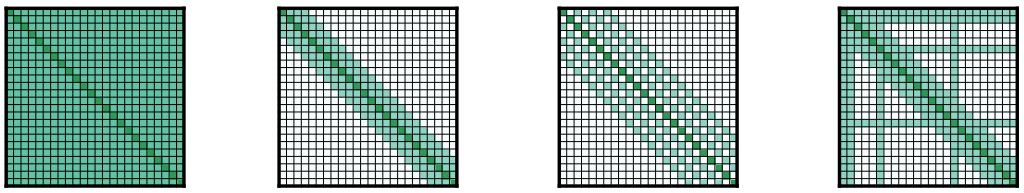


Figure 2.1: Illustration of the standard encoder-decoder Transformer architecture from Vaswani et al. (2017). The model follows an encoder-decoder structure, where both components are composed of stacked layers containing multi-head attention and position-wise feed-forward sub-layers, each with residual connections and layer normalization. In the encoder, self-attention allows each token to attend to all others in the input. The decoder includes masked self-attention to preserve autoregressive generation and a cross-attention mechanism that attends over the encoder outputs. Positional encodings are added to the input embeddings to retain sequence order.



(a) Full n^2 attention (b) Sliding window (c) Dilated window (d) Global + local

Figure 2.2: Self-attention patterns in Longformer including full, local, and hybrid forms.

decoder to attend to the encoder outputs. It operates similarly to self-attention, with the main difference being that the keys and values come from the encoder's final hidden states, while the queries are derived from the decoder's hidden states. At each decoder layer, cross-attention is computed as follows:

$$Q = H^D W_Q, \quad K = H^E W_K, \quad V = H^E W_V, \tag{2.9}$$

$$\text{CrossAttn}(Q,K,V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \tag{2.10}$$

where H^D denotes the decoder hidden states, H^E represents the encoder outputs, and W_Q, W_K, W_V are learnable projection matrices.

2.3 Efficient Attention Mechanisms for Long Inputs

Although encoder–decoder Transformers are highly effective, their quadratic attention cost limits the length of inputs they can handle. To address this challenge, this section surveys attention variants developed to improve scalability.

2.3.1 Local Attention and the Longformer

Local window attention restricts each token's attention to a fixed window of neighboring tokens, reducing the complexity from $O(n^2)$ to $O(n \cdot w)$, where w is the window size. Beltagy et al. (2020) extended this idea by combining local attention with a dilated window and a small set of global tokens that have full attention, enabling the model to capture both local and long-range dependencies as shown in Figure 2.2. This architecture forms the basis of the Longformer-Encoder-Decoder (LED), which has been successfully applied to long-document summarization tasks.

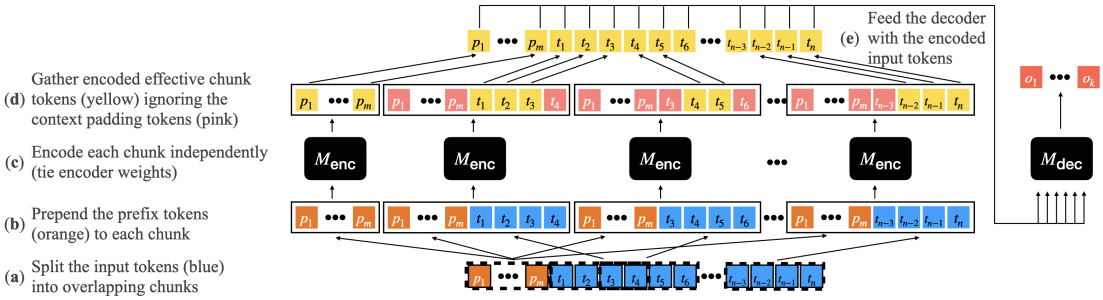


Figure 2.3: Overview of the SLiding-Encoder and Decoder (SLED) model (Ivgi et al., 2023b).

2.4 Fusion-in-Decoder and Chunk-Based Modeling

An alternative approach to managing long inputs is to divide the input into smaller chunks and fuse their representations in the decoder. This is commonly known as the Fusion-in-Decoder (FiD) approach.

2.4.1 Fusion-in-Decoder Principle

In the FiD framework (Izacard and Grave, 2021), each chunk of the input document is independently encoded by a shared encoder. The decoder then attends over the concatenated encoder outputs from all chunks. This approach is conceptually related to efficient attention strategies discussed earlier, in that it reduces encoder complexity by restricting attention within chunks, while deferring cross-chunk interactions to the decoder. Mathematically, let the document be divided into C chunks $X^{(1)}, X^{(2)}, \ldots, X^{(C)}$, and let

$$Z^{(c)} = \text{Enc}(X^{(c)}). \tag{2.11}$$

The overall encoder output is:

$$Z = \text{Concat}(Z^{(1)}, Z^{(2)}, \ldots, Z^{(C)}). \tag{2.12}$$

The decoder then generates the output conditioned on $\mathbf{Z}$:

$$P(y_t \mid y_{<t}, Z) = \text{Dec}(y_{<t}, Z). \tag{2.13}$$

This approach allows the model to effectively process very long documents without changing the underlying encoder architecture.

An example of the FiD approach is the SLiding-Encoder and Decoder(SLED) model (Ivgi et al., 2023b). In SLED, the input document is divided into overlapping

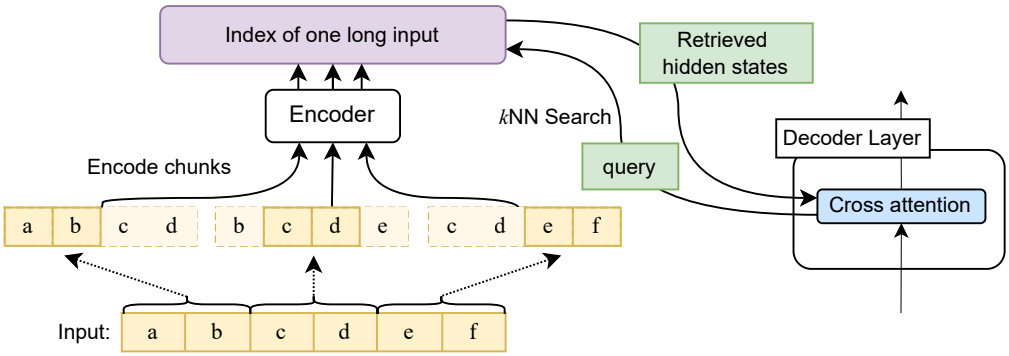


Figure 2.4: An overview of Unlimiformer (Bertsch et al., 2023b) architecture.

chunks to mitigate boundary effects. Each chunk is encoded independently, and only the central portion (the non-overlapping part) of each chunk is retained. The decoder attends to the concatenated outputs to produce a summary, as shown in Figure 2.3. This method preserves local context within each chunk while enabling global information fusion through the decoder.

Another instance of the FiD approach is Unlimiformer (Bertsch et al., 2023b), which combines retrieval mechanisms at the decoding. Instead of concatenating all encoder outputs, Unlimiformer constructs a k-nearest neighbors (kNN) index over the encoder representations and allows the decoder to retrieve only the most relevant tokens at each generation step, as shown in Figure 2.4. This reduces the effective size of the context seen by the decoder, enabling processing of extremely long inputs.

2.4.2 Large Language Models

During the course of this thesis, a new paradigm of powerful models based on pre-training has emerged in the form of Large Language Models (LLMs). These models, such as GPT-3 (Brown et al., 2020a), GPT-4 (OpenAI et al., 2024), LLaMA (Touvron et al., 2023), and various other LLMs, have demonstrated remarkable abilities in natural language understanding and generation. They are typically trained on enormous corpora with billions of parameters, enabling them to capture rich linguistic patterns and extensive world knowledge that were previously unattainable with smaller models.

LLMs are primarily built on the Transformer architecture (see Section 2.1) and can be categorized into two main types: encoder-decoder models (see Section 2.2) and decoder-only models. Decoder-only models, such as the GPT family and LLaMA, generate text in an autoregressive manner by predicting the next token given all previous tokens.

Despite their impressive capabilities, LLMs come with a critical limitation: a fixed maximum context window. For example, GPT-3 is typically limited to 2048 tokens, and many encoder-decoder models are restricted to 512 or 1024 tokens. More recent models have pushed this boundary much further: for example, LLaMA-3.1 supports up to 128K tokens. This maximum context is defined during training by the model's positional encoding scheme or architectural design. As a result, processing inputs longer than the pre-defined limit is challenging because the model has never seen such long sequences and may not represent positions beyond the trained range effectively.

To overcome this limitation, several methods have been developed to extend the context length of LLMs. Infini-Attention (Munkhdalai et al., 2024) incorporates a compressive memory into the attention mechanism. Instead of attending to an ever-growing number of tokens, the model compresses older content into a fixed-size memory representation. This allows the model to process long context lengths by continuously summarizing distant information. The model is trained to learn how to maintain this compressive memory.

Another approach involves augmenting the model with an external memory, as seen in Memorizing Transformers (Wu et al., 2022). It performs an inference time modification and augments the model with an external memory bank. The model processes text and stores key hidden states in an external database.

A further method is to adjust positional encodings so that a model can generalize to longer inputs at inference time, a technique known as positional interpolation (Chen et al., 2023). The method processes a long sequence by interpolating the token positions. For example, if a model was trained with a maximum of 1024 tokens, input sequences of 2048 tokens can be mapped such that token 2048 corresponds to position 1024 in the original scheme, with intermediate positions scaled accordingly.

Another inference-time strategy is Rotary Position Embeddings (RoPE; Su et al., 2024) scaling (or theta scaling). RoPE, a relative positional encoding method incorporated during training, uses a base frequency parameter θ that determines the rate at which positional information rotates in the embedding space. By choosing a larger θ during inference, the rate of change is slowed, effectively extending the context window. This method allows models such as LLaMA-3 (Grattafiori et al., 2024) to extend their effective context from 4k to 16k or even 100k tokens with minimal degradation in performance.

2.4.3 Sparse and Dynamic Attention for Long Contexts

A complementary line of work reduces the quadratic cost of attention by sparsifying query–key interactions. Early approaches learn sparsity during training, encouraging heads to focus on a small subset of tokens (Correia et al., 2019). More recent methods focus on training-free, post-hoc sparsification that can be applied to existing LLMs. A key observation is that long-context inference comprises distinct phases with different bottlenecks: (i) training (full backprop), (ii) prefilling (processing the long input context once), and (iii) generation (token-by-token decoding).

Recent methods therefore target the prefill phase by selectively pruning attention to retain only the most informative query–key interactions. For example, MInference (Jiang et al., 2024) dynamically sparsifies attention at prefill time to accelerate long-context LLMs without retraining. Other techniques select salient key–value (KV) entries before generation, effectively predicting what the model will need and discarding the rest (SnapKV; Li et al., 2024). Related retrofitting methods compress or cache activations and KV states to reduce memory and compute in a model-agnostic fashion (Nawrot et al., 2024).

Beyond fixed local windows, newer sparsity patterns are often input-dependent, allowing the model to allocate attention where it is most beneficial for the current input. At the same time, theoretical perspectives that relate Transformers to multi-state recurrent computation further motivate state-aware sparsification when contexts are very long (Oren et al., 2024). While the underlying principle of restricting attention is closely aligned with the efficiency theme of this thesis, we highlight a key difference in focus: much of this literature targets decoder-only LLMs and the prefill phase of inference, whereas our experiments primarily use encoder–decoder summarization models.

2.5 Benchmarks for Long-Context Summarization

Several datasets have been developed to evaluate models on long-context summarization, complementing earlier work on short news summarization. GovReport (Huang et al., 2021) introduced lengthy government reports with abstractive summaries. SummScreenFD (Chen et al., 2022) provided transcripts of TV show episodes paired with community-written recaps, and was later incorporated into the Scrolls benchmark (Shaham et al., 2022). QMSum (Zhong et al., 2021) focuses on meeting tran-

scripts, while NarrativeQA (Kočiský et al., 2018), NarraSum (Zhao et al., 2022), and NarrativeXL (Moskvichev and Mai, 2023) provide long narrative texts such as books, TV series, or movies. Most relevantly, BookSum (Kryscinski et al., 2022a) contains long books paired with chapter- and book-level summaries, pushing models to process inputs exceeding 100K tokens.

These resources highlight the increasing difficulty of abstractive summarization as input length grows, and demonstrate that even state-of-the-art models struggle with coherence and factuality at extreme scales. In the following chapter, we introduce two new resources, MovieSum and MENSA, which specifically target the challenges of movie screenplay summarization and scene-level saliency.

2.6 Salience in Summarization and Narrative Salience

A core requirement of summarization is to identify which parts of the input are most important—this is known as salience (Jones, 1998). In general summarization, salience is typically estimated using features such as word frequency, position, or centrality measures. In narrative summarization, however, salience is more nuanced and must capture not only surface-level importance but also narrative importance.

2.6.1 General Salience in Summarization

Early approaches to automatic summarization, especially extractive summarization, defined salience in statistical terms. Techniques such as term frequency (TF), inverse document frequency (IDF), sentence position, and cue words were used to heuristically assign importance scores to sentences or passages (Luhn, 1958; Edmundson, 1969). Later, graph-based ranking algorithms such as LexRank (Erkan and Radev, 2004) and TextRank (Mihalcea and Tarau, 2004) represented documents as sentence graphs where nodes were linked by similarity and centrality measures were used to select the most salient sentences. Supervised models of salience trained classifiers or regression models to predict the importance of content units based on handcrafted features (Kupiec et al., 1995; Lin, 1999). More recently, neural models have shifted toward learning contextualized salience, where encoder-based models estimate importance scores based on token or sentence embeddings (Cheng and Lapata, 2016; Zhou et al., 2018). Abstractive summarization models perform training via next-token prediction without explicitly identifying salient spans (Lewis et al., 2020; Raffel et al., 2020).

2.6.2 Narrative Salience

While traditional summarization research has defined salience in statistical or contextual terms, narrative salience extends the concept to structured, temporally evolving content such as stories, screenplays, or books. In this context, salience is not solely determined by frequency or prominence but also by the narrative function of events and their roles in plot development. Narrative salience focuses on the structural and causal elements that drive a story. Classical structuralist theories (Todorov, 1971; Barthes and Duisit, 1975) suggest that stories consist of a progression of ordered elements, where transitional moments are inherently more salient in shaping the narrative arc. Similarly, cognitive theories (Lehnert, 1981) emphasize that salient events are those deeply interconnected with other events in the narrative; removing these events would disrupt the causal and thematic flow of the story. In a narrative, not all details are equally important; the salient parts typically include key plot points, major events, and critical character interactions. For instance, in a movie script, a scene where the protagonist confronts a major challenge or engages in a pivotal confrontation is generally more salient than a transitional dialogue between minor characters. Despite extensive exploration, the concept of salience still lacks a universally agreed-upon definition or operationalization.

2.7 Evaluation Metrics for Summarization

2.7.1 ROUGE Metric

The ROUGE metric (Lin, 2004) remains the standard for automatic summarization evaluation. ROUGE measures n-gram overlap between a model generated summary and one or more reference summaries. ROUGE includes ROUGE-1 (unigrams), ROUGE-2 (bigrams), and ROUGE-L (longest common subsequence), which serve as proxies for content coverage. However, ROUGE is largely recall-oriented and may not fully capture semantic equivalence.

2.7.2 BERTScore

To address ROUGE's limitations, **BERTScore** (Zhang et al., 2019) computes semantic similarity between tokens in the generated and reference summaries using contextual embeddings. BERTScore typically correlates better with human judgments than

ROUGE because it can recognize paraphrases and semantically similar phrases.

2.7.3 Factual Consistency and QA-Based Metrics

Factual consistency is particularly critical for abstractive summarization. Metrics such as **QAEval** (Deutsch et al., 2021), **AlignScore** (Zha et al., 2023), and **FactScore** (Min et al., 2023) have been introduced to assess whether the generated summary is supported by the source document. These metrics often rely on entailment models or question-answering (QA) systems to verify that the claims made in the summary are consistent with the source. For instance, QA-based evaluation generates questions based on the summary and checks whether the source can provide correct answers, serving as an indirect measure of factual correctness. Note that these metrics cannot be directly applied to the entire source document when it is very long; therefore, they are typically used to compare the generated summary against the reference summary.

2.8 Conclusion

This chapter provided essential background on summarization techniques and Transformer-based models, highlighting fundamental mechanisms such as self-attention, encoder-decoder structures, and efficient attention variants necessary for processing long inputs. We introduced key strategies, including Fusion-in-Decoder and chunk-based methods, and discussed approaches for extending context length in Large Language Models. Finally, we reviewed prior work on general salience in summarization, contrasted it with narrative salience, and outlined common evaluation metrics employed in summarization research.

Chapter 3

Movie and Scene Saliency Dataset: Creation and Benchmarking

In the previous chapter, we introduced the core principles and technical details of summarization models. We explored the Transformer architecture, examined efficient attention mechanisms to manage computational complexity, and discussed narrative salience in the context of effective summarization.

In this chapter, we present two new datasets for movie script summarization. The first dataset, MovieSum, is a collection of movie screenplays paired with their corresponding summaries. The second dataset, MENSA (Movie Scene Saliency), focuses on movie scene saliency and comprises human-annotated salient scenes from the subset of movies from MovieSum. We detail the creation process for both datasets in Section 3.2 and 3.7, provide descriptive statistics in Section 3.3 and 3.7, and evaluate them using a diverse range of models available at the time of our experiments in Section 3.4 and 3.5. The contents of this chapter have been published in Saxena and Keller (2024a) and Saxena and Keller (2024b)

3.1 Introduction

Large language models (LLMs) have demonstrated significant improvements in abstractive summarization as discussed in Chapter 2. However, these models often struggle when processing long input contexts, particularly when relevant information is distributed throughout the document (Liu et al., 2023). To better understand this phenomenon and to advance research, datasets are needed that not only contain long-form documents but also have the property that important information is dispersed through-

out the document. Movie screenplays have these characteristics: to generate a faithful summary, an understanding of scenes, characters, and events across the entire length of the screenplay is required.

More recently, research in narrative summarization has focused on TV shows and books (Kryscinski et al., 2022a; Moskvichev and Mai, 2023), with comparatively less attention given to movie screenplays (Gorinski and Lapata, 2015; Papalampidi et al., 2020). Notably, Chen et al. (2022) introduced a dataset of TV show transcripts, which has gained considerable interest and was incorporated into a long-document summarization benchmark (Shaham et al., 2022). However, TV episode transcripts differ from movie screenplays in several ways: they tend to be shorter, primarily consist of spoken dialogue with minimal scene or character descriptions, and are often not self-contained, as they reference events and characters from previous episodes. In contrast, movie screenplays are structured documents that include various screenplay elements such as scene headings, locations, character names, dialogue, and detailed scene descriptions. These elements are explicitly formatted by screenwriters to denote different aspects of the narrative.

The largest existing movie screenplay dataset (Gorinski and Lapata, 2015, 2018), ScriptBase-j, comprises 917 automatically formatted screenplays, with the most recent movie dating back to 2013. In this dataset, the formatting was produced automatically, making it difficult to reliably preserve dialogue, character names, and detailed scene descriptions, and the collection stops at 2013, limiting coverage of more recent narrative styles. To address the limitations of existing datasets, we introduce MovieSum, a large-scale dataset designed for abstractive summarization of movie screenplays. MovieSum consists of 2,200 screenplays paired with their corresponding Wikipedia plot summaries, making it more than twice the size of existing screenplay datasets. The dataset is formatted using a professional scriptwriting tool to accurately represent screenplay structure and includes IMDb IDs for each movie, allowing researchers to link external knowledge sources. The movies in MovieSum span a diverse range of genres and release years (1930–2023).

In addition to MovieSum, we introduce the Movie Scene Saliency Dataset (MENSA), a subset of MovieSum containing human-annotated salient scenes from 100 diverse movies. We define scene saliency based on whether a scene is mentioned in the summary—i.e., if a scene is referenced in the Wikipedia summary, it is considered salient. This gold-standard labeling highlights exactly which scenes contribute meaningfully to the final summary and enable training of models to recognize saliency

explicitly.

We first provide a detailed description of MovieSum, including the steps for collecting, manually formatting, and filtering screenplays. We conduct extensive experiments to evaluate the performance of state-of-the-art summarization models on MovieSum, demonstrating its utility as a benchmark dataset for narrative summarization research. Experimental results suggest that recent models struggle with long-form abstractive summarization, emphasizing the need for improved techniques. For MENSA, in addition to providing dataset statistics, we also introduce the annotation scheme we developed for scene saliency annotation and describe the human annotation process, including inter-annotator agreement analysis. Both, MovieSum and MENSA, provide comprehensive resources for investigating the challenges of long-form narrative summarization. MovieSum offers a large-scale collection of screenplays paired with plot summaries, while MENSA provides fine-grained annotations of scene saliency.

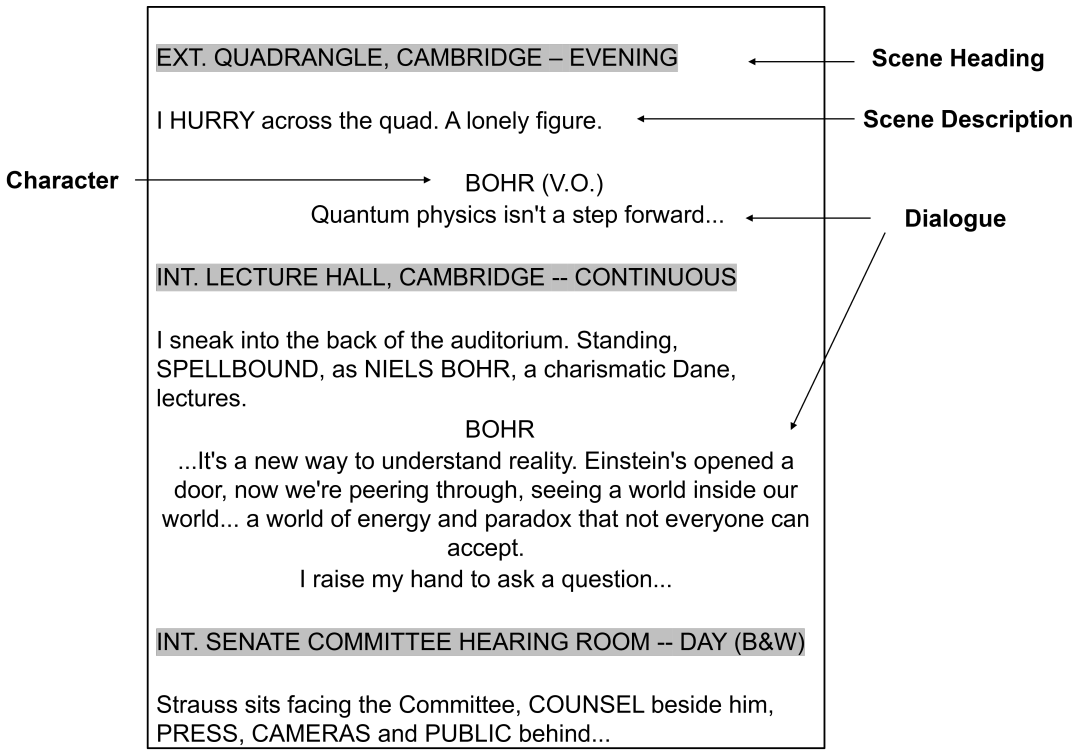


Figure 3.1: Example of cleanly formatted scenes from a movie screenplay.

3.2 The MovieSum Dataset

We present MovieSum, a movie screenplay abstractive summarization dataset that consists of 2200 movie screenplay-summary pairs. All movie screenplays in the dataset are in English.

3.2.1 Collection of Movie Screenplays

We collected movie screenplays from a range of movie screenplay websites.[1] In total, we assembled 5,639 movie screenplays documents in various text format along with metadata of movie name, IMDB identifier, and release year. If the IMDB identifier was missing, we extracted it using the IMDB database. We then manually removed movies based on two criteria. Firstly, we removed any duplicate movie screenplays by using the movie names and release years to identify the duplicates. Secondly, we filtered out screenplays which did not have text in them or were incomplete.

3.2.2 Screenplay Formatting

Movie screenplays are structured documents with various script elements such as scene headings (also known as slug lines), characters' names, dialogues, and scene descriptions (or actions). These elements have specific markers based on spacing. Most of this formatting is lost when extracting text from these movie screenplay documents, making it challenging to retrieve the elements using regular expressions. To ensure the quality of the dataset, after filtering, we manually corrected the movie screenplay and formatted each movie screenplay using Celtx,[2] a professional screenplay writing tool. Figure 3.1 shows an example of formatted scenes from a movie screenplay. This process preserved the format of movie screenplay elements. We further filtered out screenplays with encoding or optical character recognition errors.

3.2.3 Collection of Wikipedia Plot Summaries

To build a robust summarization dataset, it is necessary to collect high-quality human-written summaries. Similar to previous work (Kočiský et al., 2018), we collected Wikipedia plot summaries, which we found to be of high quality, helped by the fact that

[1] `https://www.scriptslug.com/`, `https://imsdb.com/`, `https://www.dailyscript.com/`
[2] `https://www.celtx.com/`

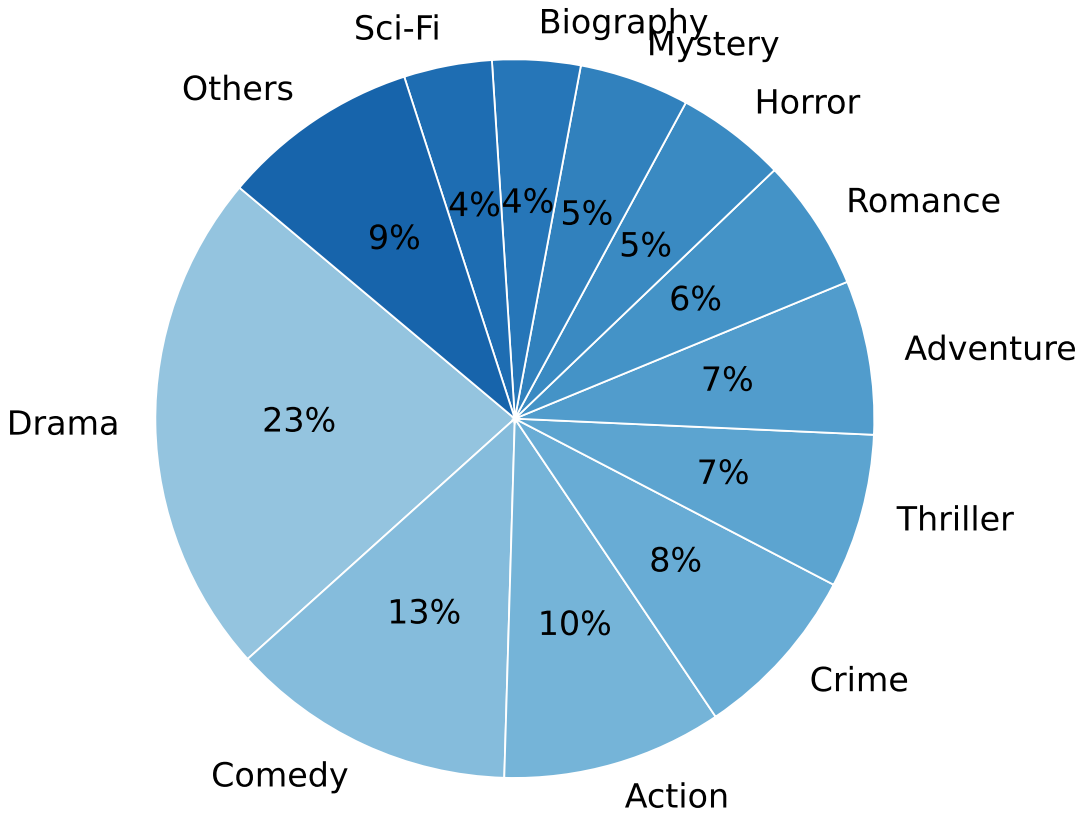


Figure 3.2: Genres Distribution of movies in the MovieSum dataset.

Wikipedia summaries follow a consistent set of guidelines for movie plot summaries.[3] It should be noted that Wikipedia plots are typically written from the visual experience of the released film rather than from the screenplay itself. This can shift the resulting summary because of differences in medium and possible deviations during production (e.g., improvisation or editing). We adopt this pairing as a pragmatic and widely used proxy (Kočiský et al., 2018), but we explicitly acknowledge this limitation.

To collect the Wikipedia plot summary, we first extracted the Wikipedia page of the movie using the movie name and year, then collected text under the Plot section. We filtered out movies where the Wikipedia page or the plot section was unavailable.

This process resulted in 2200 manually formatted movie screenplays with corresponding Wikipedia summaries.

[3] `https://en.wikipedia.org/wiki/Wikipedia:How_to_write_a_plot_summary`

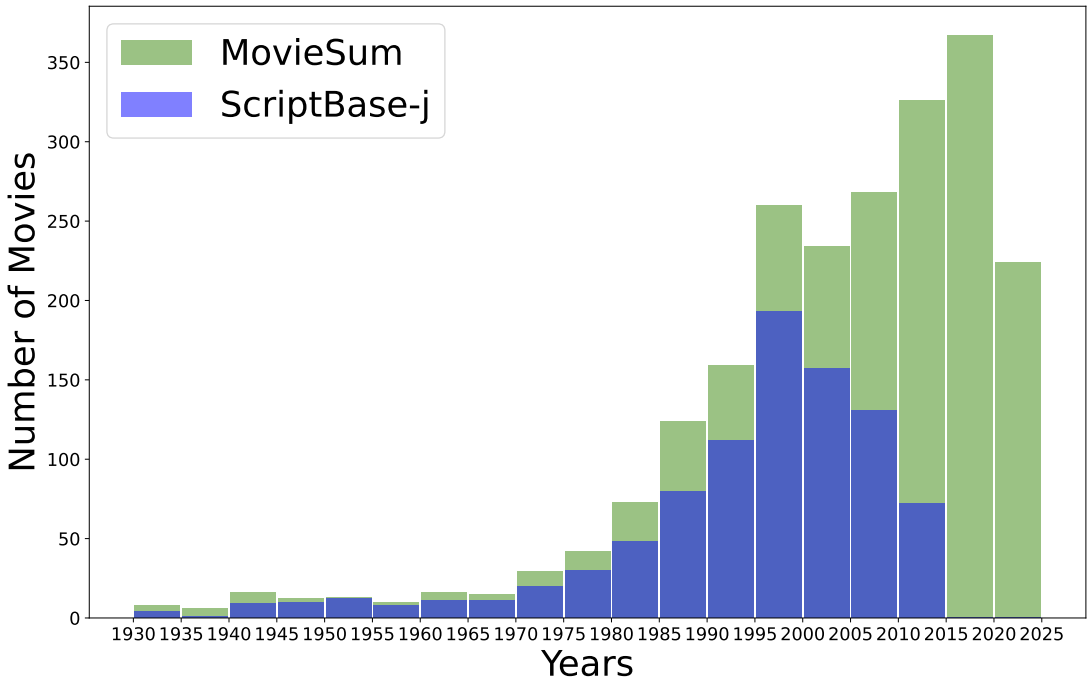


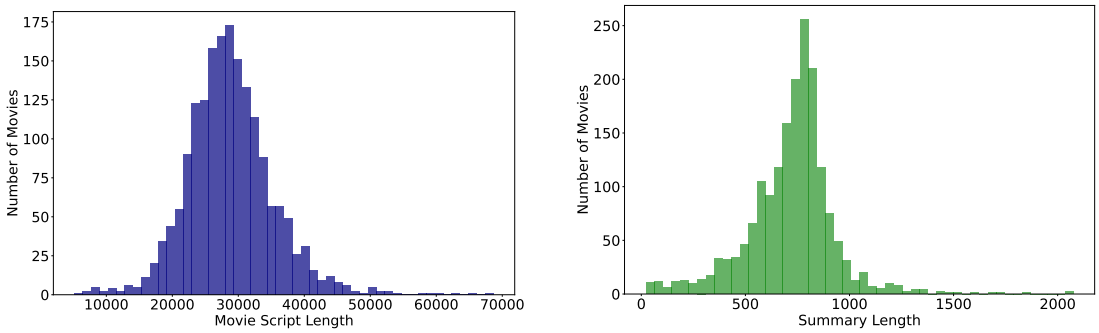
Figure 3.3: Release years of movies in MovieSum compared to ScripBase-j

3.3 Dataset Analysis

% Novel n-grams in Summary			
1-grams	2-grams	3-grams	4-grams
31.69	68.88	93.12	98.6

Table 3.1: Statistics for percentage of novel n-grams in the MovieSum summaries.

The MovieSum dataset consists of 2200 manually formatted movie screenplays along with their corresponding summaries. The average length of the screenplays is 29k words, with an average summary length of 717 words. Importantly, this dataset is twice the size of the previously available movie screenplays dataset with formatted movie screenplays (Gorinski and Lapata, 2015). Figure 3.2. illustrates the genre distribution of the movies within the dataset and showcases the broad range of genres. In Figure 3.3, the distribution of release years is depicted, revealing that the movies span a wide range of years, with a substantial number of them originating in recent years. This distribution also reflects a natural recency bias in the dataset, as more films have been released in recent decades and newer screenplays are more readily available online. Figure 3.4a and 3.4b show the length distribution for movie scripts and their summaries across the



(a) Distribution of movie script length. (b) Distribution of summary length.

Figure 3.4: Comparison of script and summary length distributions from the training set.

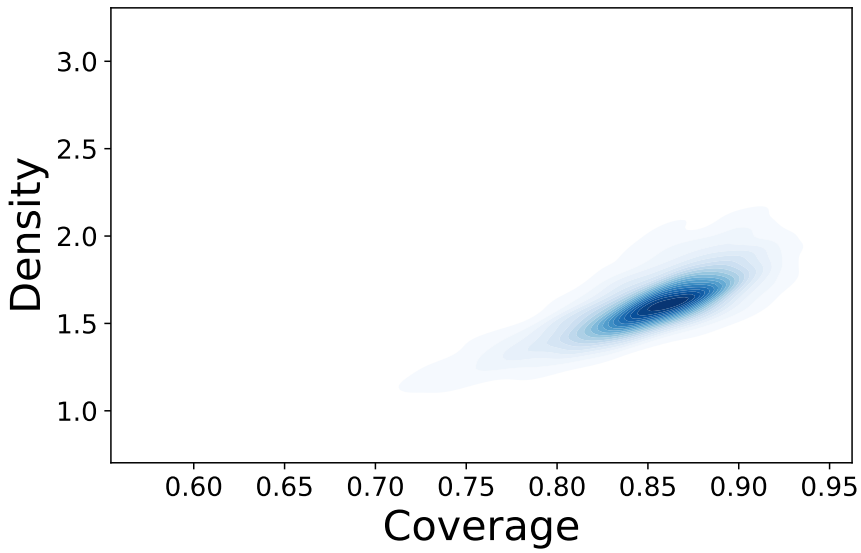


Figure 3.5: Coverage-Density plot of the summaries.

training set.

To study the abstractiveness of the summary, we report the percentage of novel n-gram in Table 3.1 as reported by Fabbri et al. (2021); Zhao et al. (2022). It shows that a high number of 3-gram and 4-gram are novel in summary and not present in the movie screenplays implying high abstractiveness of the summary. To further understand the abstractiveness of the summaries we computed the coverage and density of the summaries as discussed by Fabbri et al. (2021). The low density in Figure 3.5, indicates low overlap between the summary and the screenplays.

Datasets	Domain	Type	Task	Size	Doc. Len.	Sum. Len
ScriptBase–j	Movies	Formatted Screenplays	Summarization	917	29K	753
ScriptBase–alpha	Movies	Formatted/Raw Screenplays	Summarization	917 / 359	29K	738
SummScreenFD	TV Episodes	Raw Transcripts	Summarization	4348	7605	113
NarraSum	Movies/TV	Plot Summary	Summarization	122K	786	147
NarrativeXL	Books	Raw Text	QA	1500	87K	–
NarrativeQA	Books/Movies	Raw Text/Screenplays	QA/Summarization	783 / 789	61K	650
BookSum	Books	Raw Text	Summarization	405	112K	1167
MovieSum	Movies	Formatted Screenplays	Summarization	2200	29K	717

Table 3.2: Comparison of MovieSum with different narrative datasets for movies and TV shows.

3.3.1 Comparison with Existing Datasets

We briefly recap the most relevant benchmarks in the narrative domain from Chapter 2 and compare them to our dataset. The corresponding statistics are reported in Table 3.2. These datasets include ScriptBase-j (Gorinski and Lapata, 2015), ScriptBase-alpha (Gorinski and Lapata, 2015), SummScreenFD (Chen et al., 2022), NarraSum (Zhao et al., 2022), NarrativeXL (Moskvichev and Mai, 2023), NarrativeQA (Kočiský et al., 2018), and BookSum (Kryscinski et al., 2022a). Notably, BookSum and NarrativeXL have a longer average document length but comprise books, not screenplays. NarraSum contains plot summaries as documents rather than actual screenplays, and therefore has the lowest average document length among the datasets we compare. NarrativeQA includes both books and movie screenplays, with books being notably lengthy, making the average document length comparable to book datasets. However, it consists of only 789 unformatted movie screenplays.

MovieSum differs substantially from TV show transcript datasets such as SummScreenFD (Chen et al., 2022), the ForeverDreaming subset of the SummScreen dataset as used in the Scrolls benchmark (Shaham et al., 2022). SummScreenFD comprises transcripts of TV show episodes, which are crowd-sourced and paired with community-written recaps. In contrast, the movie scripts in our dataset were written by screenwriters and the summaries were curated by Wikipedia. It is important to note that movies and TV shows have different storytelling structures, number of acts, and length. SummScreenFD has shorter input texts and summaries compared to movie scripts as shown in Table 3.2. TV show episodes are also not self-contained, as events or characters from previous episodes can be referenced.

Both ScriptBase-j and ScriptBase-alpha datasets are close to our screenplay dataset. ScriptBase-j contains formatted screenplays, whereas ScriptBase-alpha comprises the unformatted raw text of screenplays. It is important to note that ScriptBase-j is a subset of ScriptBase-alpha, which consists of 917 formatted screenplays. On the other hand, ScriptBase-alpha includes an additional 359 movies. Our work can be considered as the extension of ScriptBase-j as it also consists of formatted screenplays. At the same time, we overcome two critical limitations of SciptBase-j:

(1) The formatting of the movie screenplay was performed automatically. Although it is easy to detect the scene heading based on rules and string matching, it is challenging to distinguish dialogues, character names, and scene descriptions. The work does not provide any details regarding the automatic formatting strategy. MovieSum, on

the other hand, includes all the movies from ScriptBase-j formatted using professional screenplay tools.

(2) Both subsets of ScriptBase consist of movies released until 2013. In contrast, MovieSum also includes recently released movies. This is crucial for ensuring that summarization models remain robust to new movie narrative conventions. The movie release years are graphed in Figure 3.3.

3.4 Experiments

We evaluate the MovieSum dataset using several baselines and representative neural abstractive summarization models available at the time of our experiments. We first report the **Lead-N** baseline, which simply outputs the first *N* tokens of the movie script as the movie summary. We varied the value *N* to understand the impact of summary length on performance and report results for **Lead-512**, **Lead-768**, and **Lead-1024**. For the extractive baseline, we used **TextRank** (Mihalcea and Tarau, 2004), a graph-based unsupervised extractive summarization method (see Chapter 4, Section 4.4.2 for more details). For instruction-tuned large language models, we used **Vicuna 1.5 13B** 16K (Zheng et al., 2023), built on Llama-2 (Touvron et al., 2023), and **FLAN-UL2** (Tay et al., 2023; Wei et al., 2022) in a zero-shot setting. For fine-tuned models with long inputs, we utilized **LongT5** (Guo et al., 2022), **PEGASUS-X** (Phang et al., 2023), and the Longformer Encoder-Decoder (**LED**) model (Beltagy et al., 2020). We selected the best-performing instruction-tuned and fine-tuned models available at the time of experimentation. These models were fully fine-tuned, and we report results on the test set. The implementation details of the models are mentioned in Appendix A.1. We report ROUGE-1 (R-1), ROUGE-2 (R-2), ROUGE-L (R-L), as well as BERTScore precision (BS_p), recall (BS_r), and F1 (BS_{f1}).

3.4.1 Results

Table 3.3 shows the summarization evaluation results using ROUGE F1 (1/2/L) scores (Lin, 2004) and BERTScore (Zhang et al., 2019) on MovieSum. The Lead baseline performs better with a higher number of words, achieving the best result with 1024 words. This is not surprising, as the ROUGE metric is known to give higher scores for longer summaries (Schluter, 2017). In the case of zero-shot, FLAN-UL2 8K performed substantially better than Vicuna 13B 16K. This confirms that a longer context

	R-1	R-2	R-L	BS_p	BS_r	BS_{f1}
Lead-512	10.35	1.27	9.84	49.25	43.59	46.23
Lead-768	14.43	1.79	13.76	49.29	45.7	47.41
Lead-1024	17.93	2.24	17.15	49.12	46.91	47.98
TextRank	33.32	5.27	32.10	51.46	52.47	51.85
FLAN-UL2 8K (ZS)	23.62	4.29	22.01	52.9	49.57	50.87
Vicuna 13B 16K (ZS)	16.35	3.55	15.44	48.89	48.49	47.07
TextRank with Vicuna 13B 16K (ZS)	17.14	3.68	15.47	59.24	49.05	53.57
Moving Window Vicuna 13B 16K (ZS)	19.56	3.32	18.57	54.95	48.7	51.53
Pegasus-X 16K	42.42	8.16	40.63	58.81	50.56	54.36
LongT5 16K	41.49	8.54	39.78	56.09	55.36	55.68
Longformer (LED) 16K	44.85	9.83	43.12	59.11	58.43	58.73

Table 3.3: Results of summarization models on the MovieSum dataset. We report ROUGE and BERTScores for the baselines and other summarization models.

does not necessarily lead to attention to the full context length. We also tested reducing the input context using TextRank, which only marginally improves the zero-shot performance for Vicuna 13 model. To further utilize the full context, we also report results for moving window chunk-based zero-shot summarization and concatenating the generated summaries. This performs better compared to using only the 16K context length. The best performance was achieved with fine-tuned models with longer context lengths. Pegasus-X, LongT5, and LED perform similarly well, with LED demonstrating superior performance.

3.5 Additional Analysis: The Role of Memorization vs. Context in LLM Performance

In addition to the experiments and results described above, we further investigate the performance of large language models (LLMs) on the MovieSum dataset. Given the widespread availability of movie-related data online, we also evaluate the extent to which LLMs have memorized information about the movies in the dataset. Specifically, we examine two prompt types: one where the model receives only the movie title and release year, and another where the full movie screenplay text is provided as con-

	R-1	R-2	R-L	BS_p	BS_r	BS_{f1}
Title and Year Prompt						
GPT-3.5-Turbo (16K)	27.37	6.64	25.7	55.99	47.87	51.55
Llama 3.1-8B (128K)	20.73	2.84	19.03	44.52	37.74	40.69
GPT-4o (128K)	43.21	11.05	40.83	56.59	55.54	56.02
Movie Screenplay Text						
GPT-3.5-Turbo (16K)	20.3	5.1	19.04	58.88	46.7	52.02
Llama 3.1-8B (128K)	11.53	1.85	10.52	53.49	37.82	43.99
GPT-4o (128K)	41.46	10.02	39.09	57.86	54.7	56.18

Table 3.4: Performance of LLMs under two prompt conditions: using only the movie title and year (testing memorization) vs. using the full movie screenplay text. We report ROUGE and BERTScores.

text for summarization (truncated to fit within a 16K token limit where necessary). The hypothesis is that providing minimal input (i.e., title and year) tests the model's ability to retrieve memorized summaries, whereas the screenplay text requires the model to process and summarize contextual details from a longer narrative. We report results for GPT-3.5-Turbo (16K context length), Llama 3.1-8B (128K context length), and GPT-4o (128K context length).

Table 3.4 presents the evaluation of the three models under both conditions. For the Title and Year prompt, models are expected to rely on memorized information from training data. In this setting, GPT-4o achieves the highest scores across all metrics (R-1: 43.21, R-L: 40.83), suggesting it has effectively memorized summaries for a substantial portion of the movies in the dataset. GPT-3.5-Turbo and Llama 3.1-8B exhibit lower performance, which may indicate either less memorized content or a reduced ability to retrieve it accurately.

When summarizing using the full movie screenplay text, the performance of all models decreases. This suggests that models are sensitive to longer inputs and may struggle to fully integrate the extended context. Despite the performance drop, GPT-4o still outperforms the other models, indicating a stronger capacity for both memorization and long-context processing. Comparing the performance of large language models requires careful consideration, as distinguishing between memorization and genuine contextual understanding is inherently complex. A more in-depth exploration of the

	R-1	**R-2**	**R-L**	**BS_p**	**BS_r**	**BS_{f1}**
Dialogues Only (LED)	44.68	10.02	42.94	59.30	58.29	58.74
Description Only (LED)	44.72	9.72	42.92	59.47	58.45	58.92
Heuristic Only (LED)	44.45	9.78	42.71	58.93	58.15	58.54

Table 3.5: Analysis of screenplay structure on summarization model (LED). We report ROUGE and BERTScores for the baselines and other summarization models.

interplay between these factors is an important direction for future research but falls beyond the scope of this thesis.

3.6 Analysis of Screenplay Structure

Figure 3.1 shows an example of a cleanly formatted screenplay with distinct elements such as scene heading, characters, and dialogues. We analyze the importance of screenplay elements, dialogue and scene description, and their impact on summarization performance. We selected the best model (LED) from the full fine-tuned experiment and studied the effects of fine-tuning solely on dialogues and descriptions. Additionally, we investigated the effects of selectively using both dialogue and scene description based on heuristics similar to Pu et al. (2022). Results in Table 3.5 show a marginal impact on summarization when dialogues or scene descriptions are removed. This effect may largely be attributed to the reduced input length, which allows the model to process a greater proportion of the movie context. We also found the heuristic based removal of input context method results is comparable to the full-text results. This suggests that the current model does not fully utilize the inherent structure of the document, and new methods for content selection and summarization should consider both screenplay elements for movie script summarization.

3.7 MENSA: Movie Scene Saliency Dataset

We now present MENSA, a **M**ovie Sc**EN**e **SA**liency dataset that is a subset of MovieSum and includes human annotation of salient scenes in movie scripts. Movie scripts are structured in terms of scenes, where each scene describes a distinct plot element and hapening at a fixed place and time, and involving a fixed set of characters. It therefore makes sense to formalize movie summarization as the identification

of the most salient scenes from a movie, followed by the generation of an abstractive summary of those scenes (Gorinski and Lapata, 2015).

We define the saliency of a movie scene based on the mention of the scene in a user-written summary of the movie. If the scene appears in the summary, then it is considered salient for understanding the narrative of the movie. By aligning summary sentences to movie scenes, we identify salient scenes and later use them for movie summarization.

The MENSA dataset consists of the scripts of 100 movies and respective Wikipedia plot summaries annotated with gold-standard sentence-to-scene alignment. We selected these movies randomly from the MovieSum test set.

Total number of movies	100
Total number of scenes	16,208
Total number of summary sentences	3,295
Sentence-Scene alignment pairs	6,063
Total number of salient scenes	5,365
Mean number of tokens in a movie	35,926
Mean number of tokens in a summary	860
Annotation Agreement	
Exact Match Agreement (*EMA*)	52.80%
Partial Agreement (*PA*)	81.63%
Mean Annotation Distance (*D*)	1.21

Table 3.6: Statistics of the MENSA dataset.

3.7.1 Annotation Scheme

Formally, let M denote a movie script consisting of a sequence of scenes $M = \{S_1, S_2, ..., S_N\}$ and let D denote the Wikipedia plot summary consisting of a sequence of sentences $D = \{s_1, s_2, ..., s_T\}$. The aim is to annotate and select a subset of salient scenes M' such that $M' \subset M$ and $|M'| \ll |M|$, where for every scene in M' there exist one or more aligned sentences in D.

To manually align the summary sentences for 100 movies, we recruited five in-house annotators. They received detailed annotation instructions and were trained by the authors until they were able to perform the alignment task reliably. To analyze inter-annotator agreement, 15 movies were selected randomly and triple-annotated by

the annotators. The remaining 85 movies were single annotated, similar to the annotation process used by Papalampidi et al. (2019), to reduce the cost of annotation. As annotating and aligning a full-length movie script with its summary is a difficult task, we provided a default alignment to annotators generated by the alignment model of Mirza et al. (2021). For every summary sentence, annotators first verified the default alignment with movie script scenes. If the alignment was only partially correct or missing, they corrected the alignment by adding or removing scenes for a given sentence using a web-based tool. We assume that each sentence can be aligned to one or more scenes and vice versa. In Table 3.6, we present statistics of the scripts and summaries in the MENSA dataset.

3.7.2 Inter-Annotator Agreement

To evaluate the quality of the annotations collected, we computed inter-annotator agreement on the triple annotated movies using three metrics: (a) Exact Match Agreement (*EMA*), (b) Partial Agreement (*PA*), and (c) Mean Annotation Distance (*D*). These measures were used for a similar annotation task by Papalampidi et al. (2019).[4] EMA is the ratio of the intersection of the scenes that the three annotators exactly agree upon for a given summary sentence, which is averaged over all sentences in the summary (Jaccard Similarity) and computed as follows:

$$EMA = \frac{1}{T_M} \sum_{s=1}^{T_M} \frac{|A_s \cap B_s \cap C_s|}{|A_s \cup B_s \cup C_s|} \tag{3.1}$$

where T_M is the total number of sentences in all the summaries, and A_s, B_s, and C_s are the indices of the scenes selected for sentence s by the three annotators.

Partial agreement (*PA*) is the ratio where there is an overlap of at least one scene among the annotators and is given as follows:

$$PA = \frac{1}{T_M} \sum_{s=1}^{T_M} [A_s \cap B_s \cap C_s \neq \emptyset] \tag{3.2}$$

Annotation distance (d) for a summary sentence s between two annotators is defined as the minimum overlap distance and is computed as follows:

$$d_s[A,B] = \min_{\forall i \in A_s, \forall j \in B_s} |i-j| \tag{3.3}$$

where A_s and B_s are the indices of the scenes selected for a sentence s by the two annotators. The mean annotation distance (D) between the three annotators is defined

[4]We renamed total agreement in Papalampidi et al. (2019) to EMA for clarity.

as the maximum pairwise overlapping annotation distance averaged for three annotators across all sentences:

$$D = \frac{1}{T_M} \sum_{s=1}^{T_M} \max(d_s[A,B,C]) \tag{3.4}$$

where $d_s[A,B,C]$ is the pairwise annotation distance between three annotators and T_M is the total number of sentences in all the summaries.

EMA and PA between our annotators was 52.80% and 81.63%, respectively. The PA indicates that for every sentence in the summaries, there is a high overlap of at least one scene. This is consistent with the low mean annotation distance of 1.21, which indicates that on average the distance between the annotations is around one scene. The EMA shows that for more than half of the sentences, there is an exact match in scene-to-sentence alignment among the annotators.

Comprehensive evaluation of a range of models on MENSA is presented in later chapters. In Chapter 4, MENSA is used to (i) assess automatic alignment techniques and (ii) train a scene-saliency classifier for explicit content selection, where the abstractive summarization performance of several methods is reported. In Chapter 5, MENSA serves as an extremely long-document benchmark.

3.8 Discussion

The MovieSum and MENSA datasets address key gaps in existing resources for long-form narrative summarization. MovieSum provides high-quality, manually formatted movie screenplays paired with human-written Wikipedia summaries, offering a realistic challenge for abstractive summarization. Compared to previously available datasets, it covers a broader range of genres and release years while preserving screenplay structure (e.g., scene headings, dialogue, and descriptions). Empirical evaluations show that contemporary models (as of the time of experimentation) struggle with summarizing long scripts, particularly in zero-shot settings. Notably, closed-source large language models (LLMs) exhibit partial memorization of movie plots when prompted with only a title and release year.

Further examination of screenplay structure suggests that current modeling approaches do not fully utilize inherent document cues. Moreover, the introduction of MENSA, a human-annotated dataset of salient scenes, allows for more fine-grained analysis of how each scene contributes to the overall plot summary. This will enable more detailed studies on scene selection and saliency prediction. These aspects are

crucial for improving narrative comprehension in long-form text.

3.9 Summary

This chapter introduced two key resources for movie script summarization. First, the MovieSum dataset expands the available corpus of fully formatted screenplays. Second, the MENSA dataset provides human-generated scene-level saliency annotations that support research on scene selection and summarization. Together, these datasets also enable broader investigations of narrative understanding and scene-level reasoning in long-form text.

Chapter 4

Content Selection using Scene Saliency for Summarization

In the previous chapter, we introduced the MovieSum and MENSA datasets—two resources for movie summarization and scene saliency analysis. We demonstrated that summarization of long-form movie scripts is a challenging task, even for large models, due to the length and dispersed nature of salient content in scripts. MENSA further provided gold-standard scene-to-summary alignments that enabled fine-grained analysis of narrative structure and scene saliency.

In this chapter, we explore how movie summarization can benefit from content selection guided by scene saliency. We present a two-stage summarization framework: first, we train a supervised scene saliency classification model using silver-standard labels generated from the MENSA dataset; second, we use only the predicted salient scenes as input to a summarization model. We show that this approach improves summarization performance across multiple metrics while reducing input length.

4.1 Introduction

Abstractive summarization (as introduced in Chapter 2) is the process of reducing an information source to its most important content by generating a coherent summary. Long-form narrative summarization poses challenges to large language models (Beltagy et al., 2020; Zhang et al., 2020a; Huang et al., 2021) both in terms of memory complexity and in terms of attending to salient information in the text. Large language models perform poorly for long sequence lengths in zero-shot settings compared to fine-tuned models (Shaham et al., 2023). Recently, Liu et al. (2023) showed that the

performance of these models degrades when the relevant information is present in the middle of a long document. With an average length of 110 pages, movie scripts are therefore challenging to summarize.

Several methods have previously relied on content selection for summarization to reduce the input size by either performing content selection implicitly using neural network attention (Chen and Bansal, 2018; You et al., 2019; Zhong et al., 2021) or explicitly (Ladhak et al., 2020; Manakul and Gales, 2021; Zhang et al., 2022) by aligning the source document with the summary using metrics such as ROUGE (Lin, 2004). Unlike for news articles, the implicit attention-based method is problematic for movie scripts, as current methods cannot reliably process text of such length. On the other hand, current explicit methods are neither optimized nor evaluated for content selection using gold-standard labels. In addition, considering the large number of sentences in movies that contain repeated mentions of characters and locations, a method based on a lexical overlap metric such as ROUGE creates many false positives. Crucially, all these methods use source–summary alignment as an auxiliary task without actually optimizing or evaluating this task.

For news summarization, Ernst et al. (2021) created crowd-sourced development and test sets for the evaluation of proposition-level alignment. However, news texts differ from movie scripts both in length and in terms of the rigid inverted pyramid structure that is typical for news articles. For movie scripts, Mirza et al. (2021) proposed a specialized alignment method which they evaluated on a set of 10 movies. However, they do not perform movie script summarization.

Movie scripts are structured in terms of scenes, where each scene describes a distinct plot element and hapening at a fixed place and time, and involving a fixed set of characters. It therefore makes sense to formalize movie summarization as the identification of the most salient scenes from a movie, followed by the generation of an abstractive summary of those scenes (Gorinski and Lapata, 2015). Hence we define movie scene saliency based on whether the scene is mentioned in the summary i.e., if the scene is mentioned in the summary, it is considered salient. Using scene saliency for summarization is therefore a method of explicit content selection.

In this chapter, we first use the MENSA dataset, described in the previous chapter, to evaluate existing explicit alignment methods to align summary sentences with their corresponding scenes. We then propose a supervised scene saliency classification model to identify salient scenes given a movie script. Specifically, we use the alignment method that performs best on the gold-standard data to generate silver-standard labels

on a larger dataset, on which we then train a sequence classification model using scene embeddings to identify salient scenes. Using the predicted salient scenes, we fine-tuned a pre-trained language model to generate movie summaries. This model outperforms existing methods on movie summarization as measured by ROUGE and BERTScore (Zhang et al., 2020b). In addition to that, we evaluate the generated summaries using a question-answer-based metric (Deutsch et al., 2021) and show that summaries generated using only the salient scenes outperform those generated using the entire movie script or baseline models.

4.2 Related Work

4.2.1 Long-form Summarization

Summarization of long-form documents has been studied across various domains, such as news articles (Zhu et al., 2021), books (Kryscinski et al., 2022a), dialogues (Zhong et al., 2022), meetings (Zhong et al., 2021), and scientific publications (Cohan et al., 2018). As discussed in Chapter 2, many efficient Transformer variants have been proposed to handle long inputs (Zaheer et al., 2020b; Zhang et al., 2020a; Huang et al., 2021; Beltagy et al., 2020), yet even these models must truncate very long scripts due to memory and time complexity (see Table 3.6).

Over the past decade, numerous approaches movie summarization have been proposed. Gorinski and Lapata (2018, 2015) generate movie overviews using a graph-based model and create movie script summaries based on progression, diversity, and importance. In contrast, the aim of our work is to find salient scenes and use these for summarization. Papalampidi et al. (2019, 2021) summarize movie scripts by identifying turning points, important narrative events. In contrast, our approach is based on salient scenes and does not assume a rigid narrative structure. Recently, Agarwal et al. (2022) proposed a shared task for script summarization; the best model (Pu et al., 2022) used a heuristic approach which first selects the essential sentences based on the main character of the movie to truncate the script.

4.2.1.1 Summarization based on Content Selection

Several methods (Ladhak et al., 2020; Manakul and Gales, 2021; Liu et al., 2022a) have leveraged content selection for summarization. Chen and Bansal (2018) and Zhang et al. (2022) generate silver standard labels through greedy alignment of the source doc-

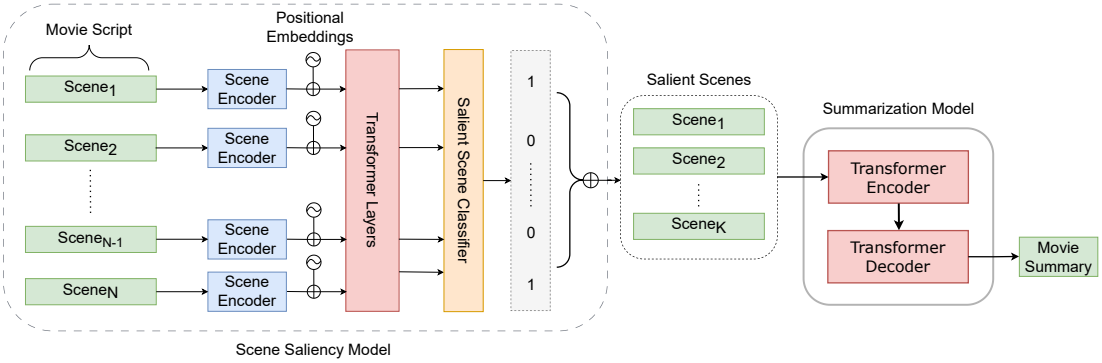


Figure 4.1: The architecture of the scene saliency detection and summarization models. The models are trained in a pipeline where salient scene detection is trained separately.

ument sentences with summary sentences. However, these methods do not explicitly evaluate alignments. Moreover, movie scripts consist of a large number of sentences with the same characters and location names, which can generate many false positives in greedy alignment. We collect gold-standard saliency labels to compare and evaluate alignment methods. Mirza et al. (2021) proposed a movie script alignment method for summaries but do not actually propose a summarization model. Recent work (Dou et al., 2021; Wang et al., 2022) has employed neural network attention for the summarization of short documents. However, movie scripts are challenging for attention-based methods, given their length.

Alignment Method	**P**	**R**	**F1**
Chen and Bansal (2018)	52.59	51.38	51.67
Zhang et al. (2022)	50.35	53.42	50.15
Mirza et al. (2021)	**84.42**	**68.53**	**73.55**

Table 4.1: Comparing alignment performance for different alignment methods on the gold-standard set.

4.3 Evaluation of Automatic Alignment Methods

Since it is too expensive and time-consuming to collect gold-standard scene saliency labels for the whole of Scriptbase (Gorinski and Lapata, 2015), we generate silver-standard labels to train a model for scene saliency classification. Based on our definition of scene saliency above, silver-standard labels for scene saliency can be generated

by aligning movie scenes with summary sentences.

Alignment between the source document segments and the summary sentences has been previously proposed for news summarization (Chen and Bansal, 2018; Zhang et al., 2022) and narrative text (Mirza et al., 2021). Using our gold-standard labels, we investigate which of these unsupervised approaches yields better alignment between movie scripts and summaries and therefore should be used to generate silver-standard labels for scene saliency.

Chen and Bansal (2018) used ROUGE-L to align a summary sentence to the most similar source document sentence. In our case, we transformed these source document (movie script) sentence-level alignments to scene-level alignments such that if the scene contains the aligned sentence, the scene will be aligned to the summary sentence. Zhang et al. (2022) used a greedy algorithm for aligning the document segment and the summary sentences. For each segment, the sentences are aligned based on the gain in ROUGE-1 score. In our case, movie scenes are considered as source document segments. Mirza et al. (2021) proposed an alignment method specifically for movie scripts using semantic similarity combined with Integer Linear Programming (ILP) to align movie script scenes to summary sentences.

We present the results of applying these three approaches on our gold-standard MENSA dataset in Table 4.1. We report macro-averaged precision (*P*), recall (*R*), and *F*1 score. The Mirza et al. (2021) method performs significantly better than the ROUGE-based methods, possibly as it was specifically proposed to align movie scenes and summary sentences.[1] We therefore used this alignment method to generate silver-standard scene saliency labels for the complete Scriptbase corpus.

Our dataset can be used in the future to evaluate content selection strategies in long documents. The gold-standard salient scenes can also be used to evaluate extractive summarization methods.

We now introduce our Select and Summarize (Select & Summ) model, which first uses a classification model (Section 4.4) to predict the salient scenes and then utilizes only the salient scenes to generate a movie summary using a pre-trained abstractive summarization model (Section 4.5). These models are trained in a two-stage pipeline.

[1]It was also used to generate the default alignment that our human annotators had to correct, which biases our evaluation towards the method of Mirza et al. (2021). However, our results are still a good measure of how many errors human annotators find in the alignment generated by this method.

4.4 Scene Saliency Classification Model

Using the set of generated silver-standard labels for scene saliency, we train a neural network-based classification model to predict scene saliency. We formulate this task as a sequence labeling task where the model takes a sequence of scenes $M = \{S_1, S_2, ..., S_N\}$ as input and predicts a sequence of binary labels $Y = \{y_1, y_2, ..., y_N\}$ denoting whether a scene is salient.

The model consists of two components, as shown in Figure 4.1. The first component is a scene encoder which computes scene representations by concatenating the sentences in the scene and encodes them using a pre-trained language model. Next, to learn contextual scene representation across the whole movie, we further encode the scene embeddings generated by the scene encoder using a transformer (Vaswani et al., 2017) block (L layers stacked), with unmasked self-attention initialized with random weights (Liu and Lapata, 2019). To preserve the sequence of the scenes, we add positional encodings to scene representations obtained from the first component. The final contextualized representation of the scenes is then used to classify whether scenes are salient or not. The model is trained for binary sequence labeling using the binary cross-entropy loss.

4.4.1 Dataset

To train the saliency model, we used the ScriptBase corpus (Gorinski and Lapata, 2015) that contains preprocessed scripts of movies with Wikipedia summaries. We removed the movies used in our gold-standard MENSA dataset from Scriptbase from the training set (note that the MovieSum dataset was released after this work, and this chapter only uses MENSA). This resulted in a training set containing 824 movie scripts, for which we generated silver-standard scene saliency labels using the model of Mirza et al. (2021), as previously discussed. We randomly split our gold-standard scene saliency dataset of 100 movies, using half of it for validation and the other half for testing.

4.4.2 Baselines

Majority Class: We used predicting the majority class as a simple baseline for classification. The dataset is highly imbalanced, with non-salient being the majority class.
Unsupervised TextRank: We used an extension of TextRank (Mihalcea and Tarau, 2004; Zheng and Lapata, 2019), a graph-based algorithm which is used for unsuper-

vised extractive summarization. Similar to Papalampidi et al. (2020), instead of a sentence-based graph we constructed a movie script graph such that nodes in the graph correspond to the scenes in the movie M. The edge e_{ij} between any two scene nodes S_i and S_j represents their similarity, with the edge weight being the similarity score. The centrality of a node S_i measures the importance of that node (in our case, the node represents the scene) and is computed as follows:

$$centrality(S_i) = \lambda_1 \sum_{j<i} e_{ij} + \lambda_2 \sum_{j>i} e_{ij}$$

where λ_1 and λ_2 are weights for forward-looking (edges to following scene nodes) and backward-looking (edges to preceding scene nodes) and sum to one. In our experiments, we represent the scene by computing a scene representation using a pre-trained language model (see below). We compute the weight of the edge between two nodes using the cosine similarity between the scene representations and select top-K nodes as the salient scenes based on their centrality score.

Supervised Bi-LSTM: For a supervised baseline, we used a bi-directional LSTM (Bi-LSTM) to learn contextual representation for the classification of scene saliency. Again, we computed scene representations by concatenating the sentences and encoding them using a pre-trained language model.

Note that the alignment model of Mirza et al. (2021) cannot be used as a baseline for saliency classification: it requires summaries to align to movie scripts at test time. In a summarization scenario, no summaries are available at test time.

4.4.3 Implementation Details

Our Scene Saliency Model and baseline models employed RoBERTa-large as the pre-trained scene encoder. Representation of a scene is computed using the first token's last hidden state of the model. The movie encoder transformer block has 10 layers with 16 heads and a feedforward hidden size of 2048. As the binary scene labels in the dataset are highly imbalanced, we used weighted binary cross entropy. We employed AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$ as our optimizer. The learning rate was fixed at 5e-5. For the baseline TextRank model, we performed a grid search for hyperparameters and used $\lambda_1 = 0.7$, $\lambda_2 = 0.3$, and $K = 15\%$ of move length. For Bi-LSTM, we used hidden dimension of size 512 followed by a fully connected layer. All hyperparameters were either adopted from prior work on summarization or selected via grid search on the validation set. We report only the best-performing settings.

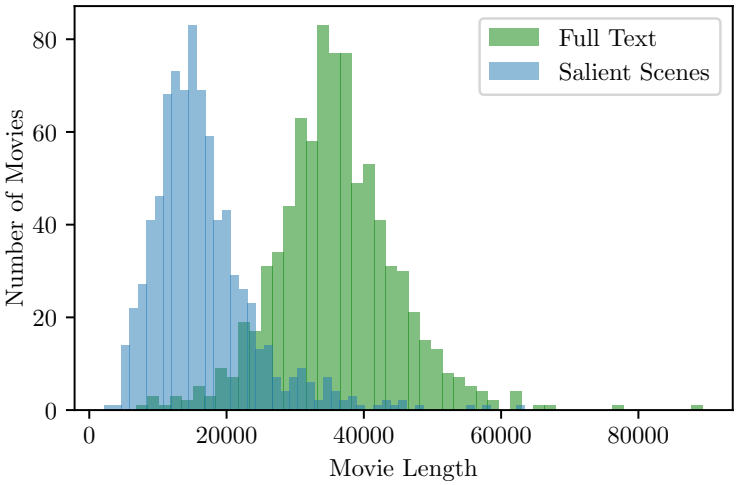


Figure 4.2: Distribution of movie length from the training set for full text and only the salient scenes.

Method	**P**	**R**	**F1**
Majority Class	31.99	50.00	38.70
TextRank	56.15	53.57	50.55
Bi-LSTM	64.39	61.85	61.17
Scene Saliency Model			
Raw Text	67.98	67.43	67.41
Character Normalized	**68.38**	**68.13**	**68.01**

Table 4.2: Comparing saliency classification performance for different classification models and baseline; macro-averaged precision (*P*), recall (*R*), and F1.

4.4.4 Results

The results of our saliency classification model and the baselines are summarized in Table 4.2. We report macro-averaged precision (P), recall (R), and F1 score for each model, as the labels are highly imbalanced given that only a limited number of scenes in each movie are salient. Our model outperforms the baselines and achieves 68.38, 68.13, and 68.01 on precision, recall, and F1. The results show that the majority baseline performance is equivalent to random guessing (macro-average). The unsupervised TextRank model has higher precision, recall, and F1 than the majority baseline, which indicates that it is able to correctly predict some scenes as salient based on the centrality score. Also, the high value of λ_1 (see Section 4.4.3) signifies that the backward-looking context is more important than forward-looking context for computing scene importance. The transformer-based scene saliency model achieves better performance than other baselines, indicating the effectiveness of transformer layers in learning the context across scene representations, which is helpful in classifying scene saliency. We also found that a higher number of layers worked better for the transformer, which indicates that more layers help in capturing complex relationships in the input.

4.4.5 Effect of the Main Character's Presence on Scene Saliency

Prior work on narrative and screenplay analysis (Papalampidi et al., 2019) suggests that the actions of the main character are a primary driver of the plot. Motivated by this, we examine whether the presence of the main character in a scene serves as a useful indicator of scene saliency.

To investigate this, we first identify the main character in each movie script by computing the total number of dialogue lines spoken by each character, selecting the one with the highest count. We then normalize the script by replacing all occurrences of this character's name with a generic placeholder token (Main_Character), since character names vary across movies. This normalization makes the main character explicitly identifiable for the model and allows it to learn association between the presence of the main character and scene importance, without relying on specific names.

As shown in Table 4.2, providing the model with this placeholder leads to a slight improvement in F1 score compared to the version trained on raw text. This suggests that tracking the main character's presence can help disambiguate which scenes are salient. However, the overall gain is relatively small, indicating that the presence of the main character is not the sole determinant of a scene's narrative relevance. Scenes

	R-1	**R-2**	**R-L**	**BS_p**	**BS_r**	**BS_{f1}**
Lead-512	10.30	1.22	9.73	49.89	43.88	46.68
Lead-1024	17.69	2.10	16.78	49.43	46.86	48.10
CreativeSumm (Agarwal et al., 2022)*	14.92	1.46	13.73	42.98	42.38	42.58
FLAN-T5-XXL (Zero-Shot)	10.82	1.13	10.54	46.22	35.81	39.48
Vicuna-13b-1.5 (Zero-Shot)	16.26	3.67	15.39	53.22	48.60	50.80
GPT-3.5 Turbo (Zero-Shot)	22.53	3.69	20.49	49.71	46.49	48.01
FLAN-UL2 (Zero-Shot)	25.86	5.42	24.71	51.40	48.79	50.71
Random Selection (LED)	41.28	8.99	39.81	54.13	53.96	54.04
$Summ^N$ Multi Stage (Zhang et al., 2022)	22.65	3.01	22.42	40.87	38.60	39.68
Unlimiformer (Bertsch et al., 2023a)	31.12	4.15	30.25	41.74	51.55	46.08
Two-Stage Heuristic (Pu et al., 2022)	46.89	11.11	44.65	56.39	56.30	56.34
Full Text (Pegasus-X)	46.20	10.58	45.35	57.10	57.01	57.05
Select & Summ (Pegasus-X)	48.29	11.40	46.62	57.39	57.20	57.31
Full Text (LED)	46.15	10.62	44.46	56.32	55.77	56.04
Select & Summ (LED)	**49.98**	**12.11**	**47.95**	**57.64**	**57.29**	**57.46**

Table 4.3: Results of our model Select and Summarize (Select & Summ) compared with other summarization models. *Denotes model results from the paper of the shared task.

involving secondary characters or key plot developments may also be marked as salient, even when the protagonist is not present.

4.5 Summarization Using Salient Scenes

We now investigate the benefit of using only salient scenes for the abstractive summarization of movie scripts. We formulate this task as a sequence-to-sequence generation problem. Formally, given a movie with a set of salient scenes $M = \{S_1, S_2, ..., S_K\}$, the goal is to generate a target summary $S = \{s_1, s_2, ..., s_m\}$. As the input length of the salient scenes is still quite large as shown in Figure 4.2, we use a Longformer Encoder-Decoder (LED) architecture (Beltagy et al., 2020). To handle long input sequences, LED uses efficient local attention with global attention for the encoder. The decoder then uses the full self-attention to the encoded tokens and to previously decoded locations to generate the summary.

4.5.1 Dataset

We used the same dataset and split as in Section 4.4.1, now with Wikipedia plot summaries as output for movie script summarization. However, instead of using the whole movie script, we utilize the output of our scene saliency model and input only the salient scenes when we generate movie summaries.

4.5.2 Baselines

We compare the proposed model with various baselines. **Lead-N** simply outputs the first *N* tokens of the movie script as the summary of the movie. We varied *N* to understand the impact of summary length on performance and report results on **Lead-512** and **Lead-1024**. **FLAN-T5-XXL** (Chung et al., 2022), **FLAN-UL2** (Wei et al., 2022), **Vicuna-13b-1.5** (Zheng et al., 2023) which is fine-tuned on Llama-2 (Touvron et al., 2023), and **GPT-3.5-Turbo**[2] Brown et al. (2020b) are instruction-tuned large language models (LLMs) which were used in zero-shot setting. **SUMM$^{\mathbf{N}}$** (Zhang et al., 2022) is a multi-stage summarization framework for long input dialogues and documents. **Unlimiformer** Bertsch et al. (2023b) uses retrieval-based attention mechanism for long document summarization. **Two-Stage Heuristic** (Pu et al., 2022) is a two-stage movie script summarization model which first selects the essential sentences based on heuristics and then summarizes the text using LED with efficient fine-tuning. **Random Selection** randomly selects salient scenes for summarization. **Full Text** takes the full movie script as input (no content selection) and truncates the text based on model input length.

4.5.3 Implementation Details

We experimented with two pre-trained models LED and Pegasus-X as base models for summarization which were fined-tuned on the Scriptbase corpus (see Section 4.4.1). Each input sequence for the movie is truncated to 16,384 tokens (including special tokens) to fit into the maximum input length of the model. We experimented with both the base and large variants of these models and found that the large models performed better and used them in our experiments. We used AdamW as an optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.99$) with a learning rate of 5e-5. We used a linear warmup strategy with 512 warmup steps. We trained the models to 60 epochs and used the checkpoint with the

[2]We used model gpt-3.5-turbo-1106 which has context length of 16K tokens.

best validation score. We used a beam size of five for decoding and generating the summary. We also created a random selection baseline by selecting a random k% of scenes and using those to generate a summary. We report the best result for random selection, which was obtained for $k = 25$ and LED. All the baseline models are fully trained on our dataset using the best configuration from the papers.

4.5.4 Results

In this section, we first present the main summarization results on the Scriptbase corpus, comparing our approach with a range of baseline and prior models. We then present additional analyses, including factuality evaluation using QA-based metrics (Section 4.6), cross-dataset zero-shot performance on SummScreen-FD (Section 4.7), and ablation and robustness studies (Section 4.8.1–4.8.3). Together, these results provide a comprehensive assessment of the proposed approach.

Table 4.3 shows our evaluation results using ROUGE (F1) scores and BERTScore on the Scriptbase corpus. Compared with the baseline models and previous work, our model achieves substantial improvements across all metrics. Specifically, our Select and Summarize model, which selects salient scenes, achieves 49.98, 12.11, and 47.95 on ROUGE-1/2/L scores and also shows improvements on BERTScore. Compared to a model which uses the full text of the movies, our model improves the performance by 3.83, 1.49, and 3.49 ROUGE-1/2/L points, respectively. The Lead-N baseline achieves better results than Agarwal et al. (2022) with a ROUGE-1 of 17.69 for Lead-1024. Our model outperforms SUMM$^{\text{N}}$ Zhang et al. (2022), which can be attributed to better content selection using salient scenes compared to greedy content selection based on ROUGE. As named entities and places are repeated across the movie script, the greedy alignment used in SUMM$^{\text{N}}$ can result in false positives. Unlimiformer performance is low compared to our model and the two-stage model, possibly because it does not include explicit content selection. The Pu et al. (2022) model performs slightly better than using Full Text, as removing sentences based on heuristics allows it to include movie script text which would otherwise be truncated. FLAN-UL2 performs better than GPT-3.5-Turbo and FLAN-T5-XXL in a zero-shot setting but our fine-tuned model outperforms all three models. Interestingly, the random selection baseline also achieves reasonably strong results. This suggests that summaries can to some extent be reconstructed even from peripheral scenes, since important characters and recurring locations appear throughout the script.

We also experimented with Pegasus-X (Phang et al., 2022) instead of LED as the base summarization model for Select & Summ. We found both models perform better when using our approach of selecting salient scenes compared to the full text, with LED demonstrating superior performance.

Figure 4.2. also shows that our model yields improvements even though it uses only half the length (only salient scenes) of the original script. This demonstrates the effectiveness of salient scene selection in movie script summarization. Appendix A.2 shows generated summaries for two movies.

Model	**F1**	**EM**
Full Text	22.70	13.81
Two-Stage (Pu et al., 2022)	25.21	14.35
Select & Summ	**29.42**	**20.05**

Table 4.4: Results of QAEval on summaries generated by Select and Summarize and baseline models.

4.6 Automatic QA-based Evaluation

Metrics like ROUGE (lexically based) and BERTScore (embedding based) are good for comparing the topic similarity between the reference and generated summaries, but fail to compare content-based factual consistency. To further evaluate the performance of our model, we used QAEval (Deutsch et al., 2021), a question-answering-based evaluation that generates question-answer pairs using the reference summaries. It then uses the model-generated summaries (candidates) to answer these questions, thereby measuring information overlap. It reports two standard answer verification methods used by SQuAD, F1 and exact match (EM) (Rajpurkar et al., 2016), averaged over all questions for all model-generated summaries.

Before the final evaluation, we filtered the generated questions using a question filtering method similar to Fabbri et al. (2022), which is useful for removing spurious questions/answers (for example answers consisting of personal pronouns and wh-pronouns). Table 4.4 shows results for QAEval on summaries generated by models using full text input, the two-stage heuristic approach (Pu et al., 2022), and Select and Summarize (our model), which uses the salient scenes predicted by our scene saliency classification model. We find that Select and Summarize performs better in answer-

ing factual questions, with a mean F1 of 29.42 and a mean exact match of 20.05%. Our model shows a clear improvement over using full movie scripts or a two-stage heuristic approach. This shows that summaries generated by our model capture more factual content compared to other baselines. Although the overall scores are fairly low, presumably because QAEval is not tailored to the evaluation of long-form narrative summaries, our model shows a clear improvement over using full movie scripts as summarization input. Appendix A.2 shows a sample of questions and answers for two movie summaries.

Model	**R-1**	**R-2**	**R-L**	**#P**
Summ^N	32.48	5.85	27.55	400
DialogLM	**35.75**	8.27	30.76	340
SLED	35.20	**8.70**	19.40	406
Select & Summ	35.61	8.58	**31.13**	161

Table 4.5: Zero-Shot performance of scene classifier on SummScreenFD compared with other baselines models. #P is the number of fine-tuned parameters in millions.

4.7 Zero-Shot on SummScreen-FD

We further investigate the performance of the scene saliency classifier on SummScreenFD (Chen et al., 2022) as used in Scrolls benchmark (Shaham et al., 2022). SummScreenFD consists of transcripts of TV show episodes with human-written recaps. We performed a zero-shot classification of the salient scenes on the SummScreenFD and used only salient scenes to fine-tune LED for summarization. We compare the results with strong existing methods on the dataset and report ROUGE scores. We observe that our model achieves comparable results to strong baselines on the SummScreenFD dataset as shown in Table 4.5, but with fewer parameters (Ivgi et al., 2023b; Zhong et al., 2022).

Model	P	R	F1
BART	66.13	66.48	66.06
LED (Encoder)	67.18	63.62	64.11
RoBERTa	**68.38**	**68.13**	**68.01**

Table 4.6: Performance of Scene Saliency Model for different base models as scene encoder.

4.8 Ablation and Analysis

4.8.1 Scene Encoder Experiment

We compared the performance of RoBERTa with that of BART (Lewis et al., 2020) and LED (Encoder only) as the base models for computing scene embeddings in the classification of salient scenes. For each model, we employed the large variant and extracted the encoder's last hidden state as scene embeddings. We report the results of scene saliency classification with different base models in Table 4.6. Among these models, RoBERTa's embeddings performed marginally better and also had fewer parameters.

4.8.2 Classifier Robustness

To study the robustness of the scene saliency classifier we performed k-fold cross-validation with $k = 5$. We report mean results with standard deviation across all folds in Table 4.7. The low standard deviation shows that the performance of the scene classifier is robust across different folds.

Mean Precision	Mean Recall	Mean F1
68.09 ± 0.006	68.03 ± 0.005	67.70 ± 0.003

Table 4.7: Cross validation result for scene saliency classifier.

4.8.3 Statistics for Summarization Result

All the ROUGE scores reported in this paper are mean F1 scores obtained with bootstrap resampling using 1000 samples. Table 4.8 presents these scores along with their 95% confidence intervals for both the Two-Stage Heuristic baseline and our proposed

Metric	Two-Stage Heuristic	SELECT & SUMM (LED)
R-1	46.89 (44.17-47.20)	49.98 (48.75-50.84)
R-2	11.11 (9.91-11.68)	12.11 (11.71-12.87)
R-L	44.65 (42.35-46.20)	47.95 (47.19-49.29)

Table 4.8: Statistics for Summarization Result

Select & Summ model. As shown in Table 4.8, our model significantly outperforms the closest baseline. Specifically, the our model achieves an R-1 score of 49.98 (95% CI: 48.75–50.84), compared to 46.89 (95% CI: 44.17–47.20) for the Two-Stage Heuristic. Similarly, improvements are observed in R-2 (12.11 vs. 11.11) and R-L (47.95 vs. 44.65). The confidence intervals for our model and the baseline have no overlap, indicating that the improvements are statistically significant. These results demonstrate that our approach, which leverages explicit scene saliency in a two-stage framework, produces better summaries than the baseline heuristic method.

4.8.4 Limitations

The findings of this chapter should be viewed in light of several limitations. We defined the saliency of a scene solely in terms of its recall in user-written summaries. However, saliency may also depend on other factors such as the presence of important characters, key narrative events, or the visual impact of a scene. In addition, our two-stage pipeline of salient scene classification followed by summarization has the drawback that errors in the first stage may propagate into the second without recovery.

4.9 Summary

In this chapter, we proposed a scene saliency classification model for the automatic identification of salient scenes in a movie script and introduced an abstractive summarization model that only uses the salient scenes to generate the movie summary. Our experiments showed that the proposed model achieves a significant improvement over the previous work on the Scriptbase corpus for movie script summarization and performs comparable to the best model available at the time of experimentation on the SummScreenFD dataset using zero-shot salient scene detection.

Our work demonstrates that the output of a summarization model can improve when

content selection is performed (by using only the salient scenes). A good content selection strategy can in principle reduce the input size without compromising the quality of the generated output. As a result of the smaller input size, the computational and memory requirements of the underlying large language model can be significantly reduced.

Chapter 5

Efficient Long-Document Summarization

In the previous chapter, we proposed a content selection approach for long-form narrative summarization using a scene saliency model, demonstrating that selecting salient scenes improves the performance of summarization models both in terms of automatic and factuality-based metrics. While content selection reduces input length and improves summarization quality, the encoder-decoder models used in these approaches still rely on truncated inputs during training due to memory constraints, creating a mismatch between training and inference conditions.

In this chapter, we address this limitation by introducing CachED—a gradient caching framework that enables full-document end-to-end training for encoder-decoder models without truncation. We demonstrate that CachED allows existing summarization models to utilize entire documents during training, resulting in substantial improvements in summarization performance on both long and extremely long document datasets. We apply our method to BART and T5 models and evaluate across several benchmarks, including MENSA, BookSum, GovReport, and SummScreenFD.

5.1 Introduction

Summarization aims to reduce extensive information into its most essential content (see Chapter 2). Despite the success of transformer-based (Vaswani et al., 2017) pretrained language models, these models face significant challenges when applied to long document summarization tasks (Shaham et al., 2022; Gorinski and Lapata, 2015; Kryscinski et al., 2022b). One of the primary limitations is their inability to handle long

input during training, due to the memory requirement being quadratic with respect to the sequence length. This often necessitates truncation of the input text during training, resulting in a loss of crucial information and hampering the quality of the generated summaries. This problem is particularly pronounced in domains that require processing of extremely long text, such as book summarization (Kryscinski et al., 2022b), where maintaining the full context is essential for producing accurate and meaningful summaries.

Prior work has attempted to address the limitation of processing long input, including designing attention mechanisms that are more memory efficient (Beltagy et al., 2020), dividing an input document into chunks (Bertsch et al., 2023a; Ivgi et al., 2023a; Xie et al., 2024; Yen et al., 2024), or extending context at test time (Ratner et al., 2023; Han et al., 2024). Despite all the effort, truncation during training (typically at 16K tokens) is ubiquitous and is the standard approach to dealing with memory issues during training, causing a mismatch between training and test conditions.

To tackle the problem of truncation, we propose CachED (Gradient **Cach**ing for **E**ncoder-**D**ecoder Models), a simple and efficient approach that enables end-to-end training of existing encoder-decoder transformer models for long document summarization. We follow the chunking approach because it is model-agnostic, allowing us to apply it to any pretrained model, but more importantly, providing us the opportunity to release memory between the computation of chunks. We only keep the final output of the encoder, and release the intermediate results of the encoder whenever possible. The fusion of encoder output happens at the decoder (Izacard and Grave, 2021). Gradients are computed as usual but are cached at the encoder output. The cached gradients are then propagated to the encoder chunk by chunk. The peak memory usage of our approach is greatly reduced, allowing us to train encoder-decoder models on entire input documents without truncation. Our method primarily mitigates the encoder-side cost of self-attention, which scales quadratically with input length. The decoder, however, still incurs a complexity proportional to the product of input and output lengths due to cross-attention, as all input tokens are attended in parallel during training.

We apply our approach to BART (named CachED BART) on several long document summarization benchmarks, including GovReport, SummScreenFD, QMSum, MENSA, and BookSum. CachED BART achieves superior performance compared to existing approaches even when using a small model with a context size of 1024 tokens. Our approach is also general and can be applied to any pretrained encoder-decoder models.

In summary, the contributions of this work are:

1. We propose CachED, a simple and efficient approach that enables end-to-end training of any existing encoder-decoder transformer models on long input without truncation.

2. We show that CachED BART achieves superior performance on extremely long document summarization, such as book summarization.

3. Our results properly and correctly doing gradient descent without truncation can lead to improvements and strong performance.

5.2 Related Work

5.2.1 Efficient Transformers

Prior work has investigated reducing the quadratic complexity of self-attention through efficient attention mechanisms. BigBird (Zaheer et al., 2020a), Longformer (Beltagy et al., 2020), LongT5 (Guo et al., 2022), and ETC (Ainslie et al., 2020) utilize sparse attention by restricting attention to a set of local and global tokens, thereby enabling the processing of long documents. Additionally, Linformer (Wang et al., 2020) computes self-attention using a low-rank matrix. Routing transformer (Roy et al., 2021) applies a sparse routing module based on online k-means to self-attention, reducing the overall complexity. Performer (Choromanski et al., 2021) employs a kernel-based estimation of attention, while Reformer (Kitaev et al., 2020) uses locality-sensitive hashing to reduce attention complexity. These methods typically require pretraining from scratch instead of being directly integrated into existing pre-trained models. Recently, Han et al. (2024) proposed an attention mask for zero-shot length generalization of large language models (LLMs), which extends context only at inference time.

5.2.2 Parallel Encoder/Chunk Processing

Previous work has also attempted to overcome the limitation of processing long context lengths by dividing the input into chunks and processing each chunk individually. SLED (Ivgi et al., 2023a) splits the long sequence into overlapping chunks and processes each chunk with the encoder, then fuses it in the decoder. Unlimiformer (Bertsch et al., 2023a) extends SLED by offloading the cross-attention to k-nearest neighbors

(*k*NN) indexing. Both approaches are similar to our work as they can be applied to any pre-trained encoder-decoder model without additional parameters; however, they truncate the input length during training to 16K tokens and do not utilize the full context. PageSum (Liu et al., 2022b) performs encoding and decoding separately for individual chunks, with the final outputs being a weighted combination of local predictions. This method adds new parameters for weighting, and the generated tokens have strict locality due to independent decoding. Ratner et al. (2023) employ parallel context windows for LLMs to extend context length during inference, improving in-context learning, but they do not extend the context length of the models during training.

More recently, Xie et al. (2024) propose parallel chunking with reinforcement learning-based selection, adding parameters to the model and truncating the input during training to 16K tokens. Yen et al. (2024) proposed extending the context length of decoder-only models with parallel encoding by freezing the decoder layer and adding new cross-attention layers. Our approach does not add any additional parameters to the model and can perform full fine-tuning.

5.2.3 Long Context Modeling

Recent work on LLMs has focused on extending the context length of the models. Ratner et al. (2023) employ parallel context windows for LLMs to extend context length during inference, improving in-context learning. However, they do not extend the context length of the models during training. Other research includes extrapolating positional embeddings to extend the context window without fine-tuning (Chen et al., 2023; Peng et al., 2024). Additionally, Xiao et al. (2024) have proposed window-based attention during inference. Our approach aims to utilize the full context during training. Recently, Munkhdalai et al. (2024) propose infini-attention by incorporating compressive memory into the attention mechanism. This method adds new parameters to the model and requires continual pretraining.

5.3 Abstractive Summarization with Encoder-Decoder Models

The task of summarization is to produce a summary of M tokens $y_1, y_2, \ldots, y_M$ given an input document of L tokens $x_1, x_2, \ldots, x_L$, where a token can be a word or a word-piece (Wu et al., 2016). The dominant approach to abstractive summarization is to use

an encoder-decoder model (Bahdanau et al., 2016; Raffel et al., 2020; Beltagy et al., 2020), where the encoder turns the input document into a sequence of hidden vectors and the decoder produces a summary attending to the hidden vectors. More formally,

$$h_1, \ldots, h_L = \text{Enc}(x_1, \ldots, x_L), \tag{5.1}$$

where Enc is the encoder, and

$$y_m = \text{Dec}(y_1, \ldots, y_{m-1}, h_1, \ldots, h_L) \tag{5.2}$$

where the decoder Dec is repeatedly called for $m = 1, \ldots, M$. A transformer-based encoder typically consists layers of self-attention, and a vanilla implementation requires $O(L^2)$ of memory (Vaswani et al., 2017). Long-document summarization is the setting where L is large, making it difficult to store the intermediate results of the entire input in memory.

A naive approach to solving the memory problem is truncating the input, only taking, say, the first 16,000 tokens as input and capping the length at $\min(16,000, L)$. Depending on the types of summarization, this approach can be sufficient, for example, for summarizing news articles. For long documents, such as books (Kryscinski et al., 2022b) or movie scripts (Chapter 3), naively truncating the input makes it impossible to properly perform the task, as a model has no access to the truncated input. Despite the obvious limitation, truncation is widely used during training (Beltagy et al., 2020; Guo et al., 2022; Bertsch et al., 2023a; Xie et al., 2024), and is sometimes the only option when scaling up the model size.

Another approach is to divide the input document into chunks, with each chunk encoded individually. More formally, the input document of length L is divided into K chunks, with each chunk of size $\lfloor L/K \rfloor$. Each chunk, denoted as $x_{(k-1)\lfloor L/K \rfloor+1}, \ldots, x_{k\lfloor L/K \rfloor}$, is encoded into $h_{(k-1)\lfloor L/K \rfloor+1}, \ldots, h_{k\lfloor L/K \rfloor}$, for $k = 1, \ldots, K$. The memory requirement of this approach is $O(\lfloor L/K \rfloor^2)$ per chunk, i.e., $O(K \cdot \lfloor L/K \rfloor^2) = O(L^2/K)$ in total, less than the $O(L^2)$ when running self-attention on the entire sequence. The runtime complexity of the decoder is $O(M^2 + LM)$, and does not dominate $O(L^2)$ when $M < L$. Diving the input document into chunks is sometimes called a chunk-based approach (Xie et al., 2024), the sliding window approach (Ivgi et al., 2023a), or parallel context (Yen et al., 2024; Ratner et al., 2023), in which chunks might or might not have overlaps. This approach also makes a modeling assumption: text representation can only be contextualized within each chunk, delaying further contextualization or fusion in the decoder.

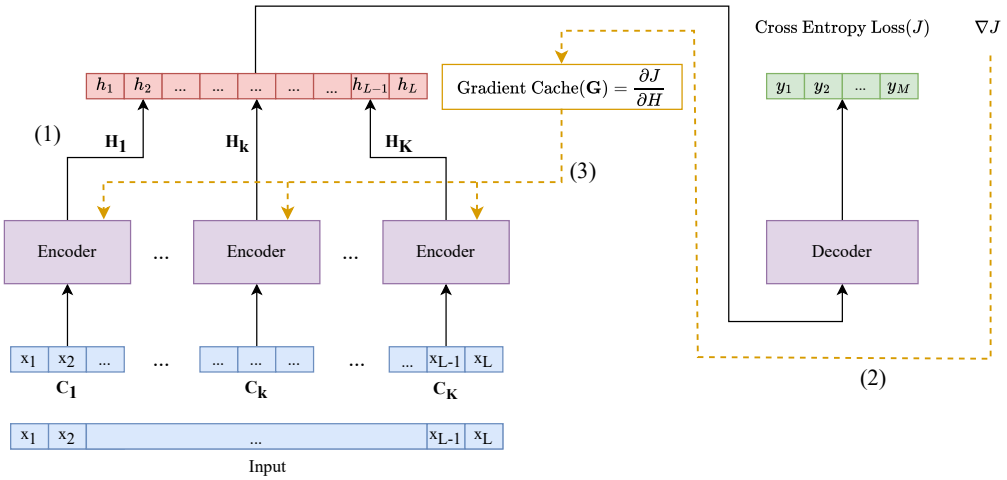


Figure 5.1: An overview of a CachED approach to long document summarization. The model is trained end-to-end by splitting and encoding the input into chunks (Step 1), then passing the concatenated hidden vectors to the decoder. Gradients are cached for the encoded tokens (Step 2) and passed back to the chunks individually (Step 3).

Despite the memory saving with sliding windows, the input documents are still truncated before chunking (Ivgi et al., 2023a; Bertsch et al., 2023a; Xie et al., 2024), because intermediate results are not released from memory after the computation of each chunk. Truncation of input documents leads to a mismatch between training and test conditions. It is still an open question whether properly and correctly doing gradient descent end to end without truncation would be better than that with truncation, a question to be addressed in this chapter.

5.4 CachED: Gradient Caching for Encoder-Decoder Models

To address the compromise of truncating input documents, we propose gradient caching for encoder-decoder models (CachED), an approach that turns any existing transformer-based encoder-decoder models into a model for long document summarization without truncation.

Recall that an input document of length L is divided into K chunks and fitting $O(L^2/K)$ in memory is still difficult, especially when there are many layers. The $O(L^2/K)$ memory requirement is due to K calls to the encoder, each of which requires

$O(L^2/K^2)$. Instead of maintaining K calls simultaneously in memory, we decide to call the encoder K times in sequence, releasing the memory after each call and reducing the memory requirement to $O(L^2/K^2)$. This can be easily done during inference, but not maintaining all K calls in memory makes computing the gradient difficult during training.

To compute the full gradient without truncation, we ideally want to break the computation up with respect to the K chunks. If we use J to denote the loss function and Θ to denote the parameters in the encoder, the relationship between the derivative of the individual K chunks and the gradient is

$$\frac{\partial J}{\partial \Theta} = \sum_{k=1}^{K} \frac{\partial J}{\partial H_k} \frac{\partial H_k}{\partial \Theta}, \tag{5.3}$$

where $H_k = \begin{bmatrix} h_{(k-1)\lfloor L/K \rfloor + 1} & \cdots & h_{k\lfloor L/K \rfloor} \end{bmatrix}$ is the concatenation of the hidden vectors from the k-th chunk. The total derivative naturally leads to the following three steps.

1. Compute the encoder output H_k for $k = 1, \ldots, K$ in sequence without storing the intermediate layers.

2. Compute the loss J based on the concatenated encoder outputs $H = [H_1, \ldots, H_K]$ and obtain the gradient $\partial J / \partial H$ with regular backpropagation.

3. Re-compute H_k and the intermediate layers and use the cached $\partial J / \partial H_k$ to compute $\partial H_k / \partial \Theta$ in sequence for $k = 1, \ldots, K$, accumulating the final gradient to the parameters $\partial J / \partial \Theta$.

Figure 5.1 illustrates the process of the CachED approach.

Step 1 can be implemented as a simple for loop,

```
Hs = []
for k in range(K):
    Ck = X[k*(L//K):(k+1)*(L//K)]
    Hk = Enc(Ck)
    Hk.detach()
    Hs.append(Hk)
```

where `X` is the concatenation of the tokens $x_1, \ldots, x_L$ in the input document, and `Enc` is the forward function of the encoder, and `L // K` is the chunk size $\lfloor L/K \rfloor$. Note that `detach()` makes it explicit that the intermediate results before `Hk` can be discarded and

do not occupy memory.[1] Though we present this step as a `for` loop, the K calls within the `for` loop are trivially parallelizable and can be batched.

Step 2 can be implemented with regular backpropagation as follows.

```
H = torch.cat(Hs)
H.retain_grad()
Yhat = Dec(Y, H)
loss = cross_entropy(Yhat, Y)
loss.backward()
```

where `Dec` is the forward function of the decoder, and `Y` is the concatenation of output tokens $y_1, \ldots, y_M$. Since the gradients are not normally stored unless the variables are parameters, the call `retain_grad()` is necessary to guarantee that the gradient to `H` is computed and cached when `loss.backward()` is called.

Step 3 continues the incomplete backpropagation from Step 2 to the encoder,

```
for k in range(K):
    Ck = X[k*(L//K):(k+1)*(L//K)]
    Hk = Enc(Ck)
    Gk = H.grad[k*(L//K):(k+1)*(L//K)]
    torch.autograd.backward(Hk, Gk)
```

Again, though this step is presented as a `for` loop, the K calls within the `for` loop are trivially parallelizable and can be batched. At the end of Step 3, we have the full gradient with respect to both the encoder and the decoder model parameters, and are ready to make a gradient update.

While our CachED approach is reminiscent of gradient checkpointing (Chen et al., 2016), it differs in key respects. Standard gradient checkpointing reduces memory usage by discarding intermediate activations and recomputing them within the same forward pass. However, checkpointing alone does not address the truncation problem in long-document training: the entire input sequence still needs to be processed at once. One drawback of the CachED approach is that the encoders need to be called twice, but the runtime cost is typically marginal if the K calls are properly parallelized and batched. Similar to Ivgi et al. (2023a), ours is a fusion-in-decoder approach (Izacard and Grave, 2021). We assume that the encoder can sufficiently contextualize input tokens within a chunk, while the decoder is responsible for managing long-range dependencies. Similar to fusion in decoder, we do not modify the positional encoding

[1] We use the language of `pytorch`, such as `detach()`, to describe the implementation, but similar concepts exist in other automatic differentiation toolkits.

of the underlying model, making our method independent of the positional embedding used by the backbone. We will study the runtime, memory, and efficacy of our approach in the experiments.

5.5 Experimental Settings

To showcase the CachED approach, we use BART and T5 as the backbone models. BART is chosen because it performs well on short text summarization but is less effective on longer text due to its input size limit, while T5 offers robust performance across diverse tasks. We will show how applying our approach to BART and T5, resulting in CachED BART and CachED T5, significantly outperforms their respective baselines, highlighting the benefits of our method. We experiment with both $\text{BART}_{\text{base}}$ and $\text{BART}_{\text{large}}$, as well as T5_{large}, comparing to other approaches that also fine-tune these models. In all experiments, we use a chunk size of 1024 tokens for BART and a context size of 512 tokens for T5. See Appendix A.3 for more details on the implementation and hyperparameters of the model.

We report ROUGE F1 (1/2/L) scores (Lin, 2004) and BERTScore F1 (Zhang et al., 2020b) to evaluate the performance of our method on long document summarization tasks.

5.5.1 Datasets

We categorize the long document summarization datasets into 1) long documents, with a mean token length of less than 16K tokens, and 2) extremely long documents, with a mean length greater than 16K tokens. Figure 5.2 shows the mean and standard deviation of the input document length in tokens for the datasets we used.

5.5.1.1 Long Document Summarization

GovReport (Huang et al., 2021) is a large-scale summarization dataset consisting of reports published by the U.S. Government Accountability Office on national policy issues. The task is to write an executive summary of each report. The mean length of the documents is 9,616 tokens.

SummScreenFD (Chen et al., 2022) consists of community-contributed transcripts of television show episodes collected from the ForeverDream (FD) website. The summaries are recaps collected from Wikipedia and TVMaze. The mean length of the

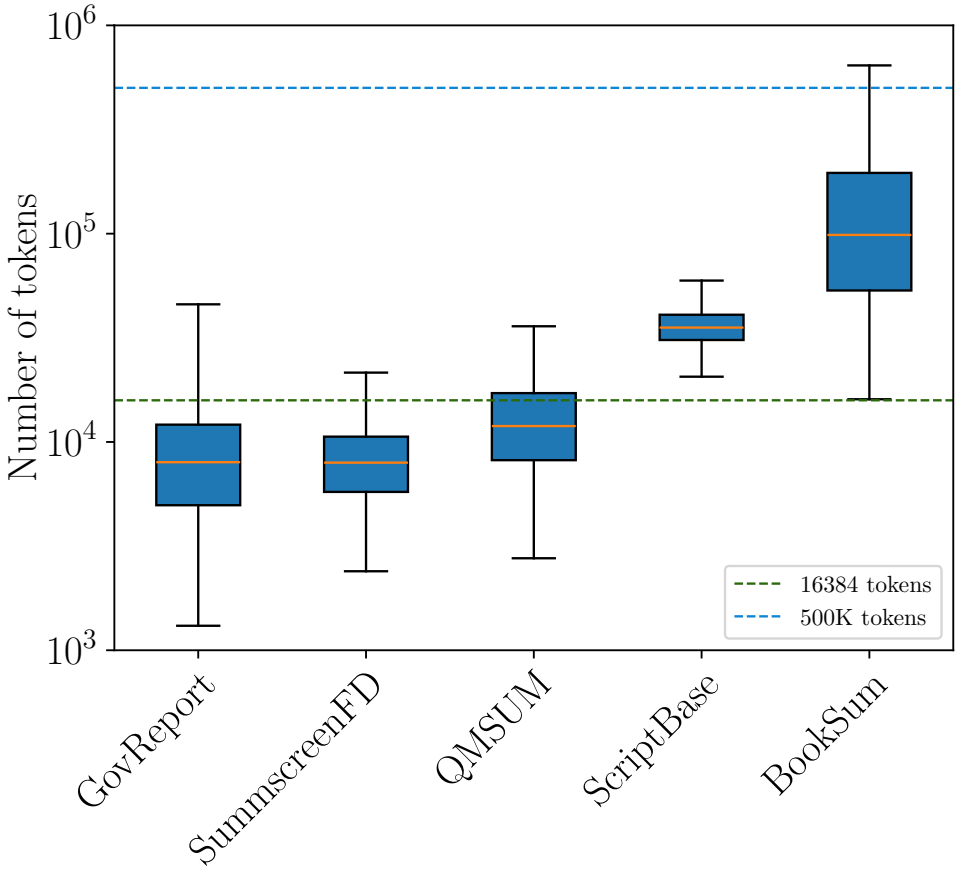


Figure 5.2: The mean and standard deviation of document lengths across summarization datasets, plotted on log scale.

documents is 8,417 tokens.

QMSUM (Zhong et al., 2021) is a query-based multi-domain meeting summarization dataset. The dataset consists of tuples of a query, document, and its corresponding summary. The mean length of the documents is 13,325 tokens.

5.5.1.2 Extremely Long Document Summarization

MENSA (Chapter 3) consists of full-length movie scripts and corresponding Wikipedia summaries. The mean length of the movie scripts is 35,956 tokens.

BookSum (Kryscinski et al., 2022b) consists of book text along with their summaries. BookSum features three subsets: paragraph, chapter, and book-level tasks. We focus on the most challenging BOOKSUM-Book task, which involves generating a summary of an entire book using the full text of the book. The mean length of the books is 139,219 tokens. The longest document in this dataset consists of 642,376 number of tokens.

5.5.2 Baselines

BART (Lewis et al., 2020) is a pretrained encoder-decoder model with 139M parameters in $\text{BART}_{\text{base}}$ and 406M parameters in $\text{BART}_{\text{large}}$. Its maximum input sequence length is 1024 tokens. We fine-tune both the base model and the large model each dataset separately.

T5 (Raffel et al., 2020) is a pretrained encoder-decoder model designed for text-to-text tasks. We fine-tune T5_{large} which contains 770M parameters. Its maximum input length is 512 tokens. **LED** (Beltagy et al., 2020) is a Longformer-Encoder-Decoder (162M) parameters with a maximum input length of 16,384 tokens. We fine-tune the base version of this model.

Unlimiformer (Bertsch et al., 2023a) augments pretrained encoder-decoders and offloads the cross-attention computation to a *k*NN index, allowing for unlimited context. We compare our approach with their custom fine-tuned $\text{BART}_{\text{base}}$ and PRIMERA model as its backbone. The maximum number of tokens is 16K during training. For PRIMERA, we report numbers from the paper, denoted by (*), as we could not replicate the result.

SLED (Ivgi et al., 2023a) extends pretrained encoder-decoder models for longer contexts by encoding the long input in chunks, and applying fusion in decoder. Since ours and theirs are both model-agnostic approaches, we compare our approach with theirs applied to $\text{BART}_{\text{base}}$ and $\text{BART}_{\text{large}}$ (their best settings). During training, the maximum

context length is 16K.

5.6 Results

5.6.1 Long Document Summarization

Table 5.1 presents the evaluation results of various models on the long document summarization datasets: GovReport, SummScreenFD, and QMSum. Our CachED approach demonstrates superior performance on the SummScreenFD and QMSum datasets and competitive performance on the GovReport dataset compared to the baseline models and previously proposed methods. Both CachED $\text{BART}_{\text{base}}$ and CachED $\text{BART}_{\text{large}}$ achieve substantial improvements over standard fine-tuning (SFT) and other competitive approaches such as SLED and Unlimiformer. CachED $\text{BART}_{\text{base}}$ outperforms bigger models on the SummScreenFD and QMSum datasets highlighting the effectiveness of our method in processing long context. In all the experiments, CachED T5_{large}, did not surpass CachED BART but demonstrated a notable improvement over standard T5 fine-tuning, showcasing the applicability of our method across different backbone architectures.

GovReport: For the GovReport dataset, CachED $\text{BART}_{\text{base}}$ achieves a ROUGE-2 score of 26.3, matching the best performance among all models. The large version, CachED $\text{BART}_{\text{large}}$, further improves the ROUGE-L score to 28.19 compared to the previous methods. SLED ($\text{BART}_{\text{large}}$) achieves slightly better ROUGE-1 across the models, and Unlimiformer ($\text{BART}_{\text{base}}$) achieves the best BERTScore F1.

SummScreenFD: On the SummScreenFD dataset, CachED $\text{BART}_{\text{large}}$ outperforms all the models with ROUGE scores of 37.2/9.1/20.1. These results surpass all competing models, including Unlimiformer (PRIMERA) and SLED ($\text{BART}_{\text{large}}$) on ROUGE and BERTScore.

QMSum: For the QMSum dataset, CachED $\text{BART}_{\text{large}}$ achieves the highest scores across all metrics with a ROUGE 38.9/14.0/24.6. Unlimiformer performs poorly, even worse compared to standard fine-tuned BART with a 1024-token context. Compared to SLED, our approach is better by 5.7/3.0/2.6 ROUGE scores, a substantial advantage without even using additional parameters.

Table 5.1: Test results on long document summarization datasets using different base models wtih standard fine-tuning (SFT). The best metric in every dataset is marked in **bold**. Some results of Unlimiformer with PRIMERA, denoted by (*), are not reported due to out-of-memory issues, and the reported values are taken from the original paper.

Method	Parameters	ROUGE 1 / 2 / L / BERTScore F1		
		GovReport	**SummScreenFD**	**QMSum**
SFT ($T5_{large}$)	770M	45.3 / 18.8 / 22.4 / 61.7	28.9 / 5.3 / 16.9 / 58.1	29.6 / 7.0 / 19.4 / 57.6
SFT ($BART_{base}$)	139M	50.2 / 19.0 / 23.7 / 63.4	31.5 / 6.7 / 18.1 / 58.2	32.9 / 8.8 / 21.1 / 59.5
SFT ($BART_{large}$)	406M	54.4 / 21.1 / 25.1 / 65.6	34.1 / 7.5 / 18.9 / 59.6	33.0 / 7.9 / 20.0 / 59.5
SFT (LED_{base})	162M	56.3 / 25.8 / 27.4 / 65.6	33.8 / 8.3 / 19.8 / 59.5	30.9 / 7.8 / 19.5 / 57.6
SLED ($BART_{base}$)	139M	54.7 / 24.4 / 25.4 / 67.0	32.7 / 7.9 / 19.1 / 58.4	33.8 / 11.7 / 22.6 / 59.1
Unlim. ($BART_{base}$)	139M	56.6 / **26.3** / 27.6 / **68.2**	34.7 / 8.5 / 19.9 / 58.5	30.9 / 8.00 / 19.9 / 58.2
SLED ($BART_{large}$)	406M	**57.5** / **26.3** / 27.4 / 66.9	35.2 / 8.7 / 19.4 / 59.9	34.2 / 11.0 / 22.0 / 58.3
Unlim. (PRIMERA)*	447M	57.4 / 26.2 / 28.0 / 68.1	33.3 / 7.6 / 18.9 / 57.7	
CachED $T5_{large}$	770M	51.9 / 22.5 / 24.5 / 63.4	32.8 / 8.2 / 19.6 / 60.2	32.9 / 8.4 / 19.9 / 59.4
CachED $BART_{base}$	139M	56.8 / **26.3** / 27.8 / 67.0	36.6 / 8.8 / 19.9 / 61.3	38.4 / 13.5 / 24.4 / 62.2
CachED $BART_{large}$	406M	57.0 / **26.3** / **28.19** / 67.3	**37.2** / **9.1** / **20.1** / **61.59**	**38.9** / **14.0** / **24.6** / **62.5**

Table 5.2: Test results on extremely long document summarization datasets using different base models with standard fine-tuning (SFT). The best metric in every dataset is marked in **bold**. Some results of Unlimiformer with PRIMERA, denoted by (*), are not reported due to out-of-memory issues, and the reported values are taken from the original paper.

Method	ROUGE 1 / 2 / L / BERTScore F1	
	MENSA	**BookSum**
SFT ($T5_{large}$)	33.4 / 4.5 / 14.5 / 55.9	19.9 / 3.0 / 11.4 / 47.2
SFT ($BART_{base}$)	39.2 / 8.2 / 17.2 / 57.8	23.6 / 5.0 / 13.2 / 49.0
SFT ($BART_{large}$)	42.9 / 8.9 / 17.8 / 60.1	24.7 / 5.8 / 14.0 / 48.3
SFT (LED_{base})	45.5 / 10.5 / 19.4 / 60.8	26.2 / 3.8 / 16.9 / 47.3
Unlim. (PRIMERA)*		38.2 / 7.1 / 16.0 /
Unlim. ($BART_{base}$)	44.8 / 12.3 / 18.3 / 58.7	36.7 / 7.3 / 15.5 / 51.5
SLED ($BART_{large}$)	45.2 / 11.9 / 17.8 / 58.3	38.9 / 7.5 / 15.8 / 52.4
CachED $T5_{large}$	36.7 / 7.7 / 17.6 / 56.9	29.5 / 5.6 / 15.9 / 49.8
CachED $BART_{base}$	48.9 / 14.4 / 19.8 / 64.1	39.4 / 9.2 / 17.0 / 53.6
CachED $BART_{large}$	**50.2** / **14.9** / **20.4** / **64.6**	**42.8** / **10.5** / **18.8** / **54.4**

5.6.2 Extremely Long Document Summarization

Table 5.2 presents the evaluation results of various models on MENSA and BookSum, where we report ROUGE (1/2/L) and BERTScore F1 metrics. Our proposed method substantially outperforms the baseline and previous methods across both datasets, showcasing its efficacy in handling extremely long document summarization tasks.

MENSA: For the MENSA dataset, CachED $BART_{base}$ achieves a notable improvement with ROUGE-1 of 48.9, ROUGE-2 of 14.4, ROUGE-L of 19.8, and BERTScore F1 of 64.1. The large version, CachED $BART_{large}$, further enhances performance, setting new state-of-the-art scores with ROUGE-1 of 50.2, ROUGE-2 of 14.9, ROUGE-L of 20.4, and BERTScore F1 of 64.6.

BookSum: On the BookSum dataset, CachED $BART_{base}$ demonstrates substantial gains with ROUGE scores of 39.4/9.2/17.0, and BERTScore F1 of 53.6. The large version, CachED $BART_{large}$, achieves the highest scores across all metrics, improving

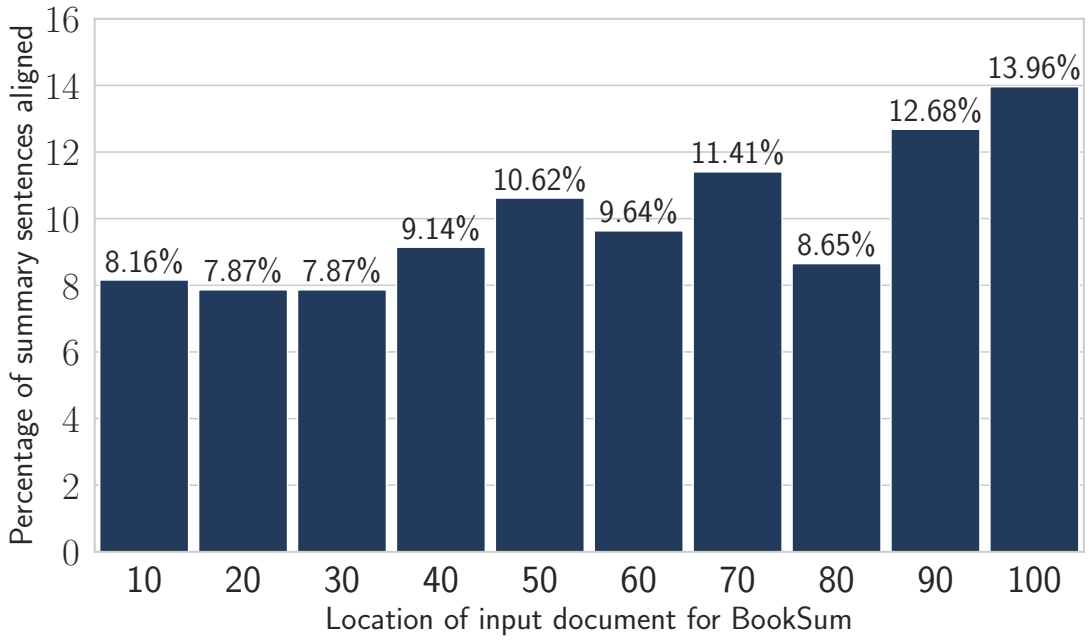


Figure 5.3: Percentage of generated summary sentences aligned with different segments (bins) of a book in the BookSum test set. The CachED $\text{BART}_{\text{large}}$ model uses the entire document, with the alignment evenly distributed across segments.

ROUGE 1/2/L scores by 3.9/3/3 compared to the best previous method. Appendix A.6 shows the generated summary of a book using our method.

5.6.3 Summary

In both dataset categories, CachED $\text{BART}_{\text{base}}$ shows substantial improvements over other variants of BART, especially on extremely long documents, compared to models twice its size in terms of the number of parameters. Similar to long document datasets, CachED T5_{large} did not surpass CachED BART but demonstrated considerable improvement over T5_{large} standard fine-tuning on extremely long document datasets.

Similar to SLED, our approach is model-agnostic, applicable to any existing encoder-decoder models. Comparing to SLED, the fact that our approach performs better shows that properly doing gradient descent without truncation can significantly improve performance. Together with SLED, our results show that a context as small as 1024 tokens can have strong performance compared to models with much longer context. Overall, CachED $\text{BART}_{\text{large}}$ outperforms CachED $\text{BART}_{\text{base}}$, showcasing that our approach can also benefit from scale.

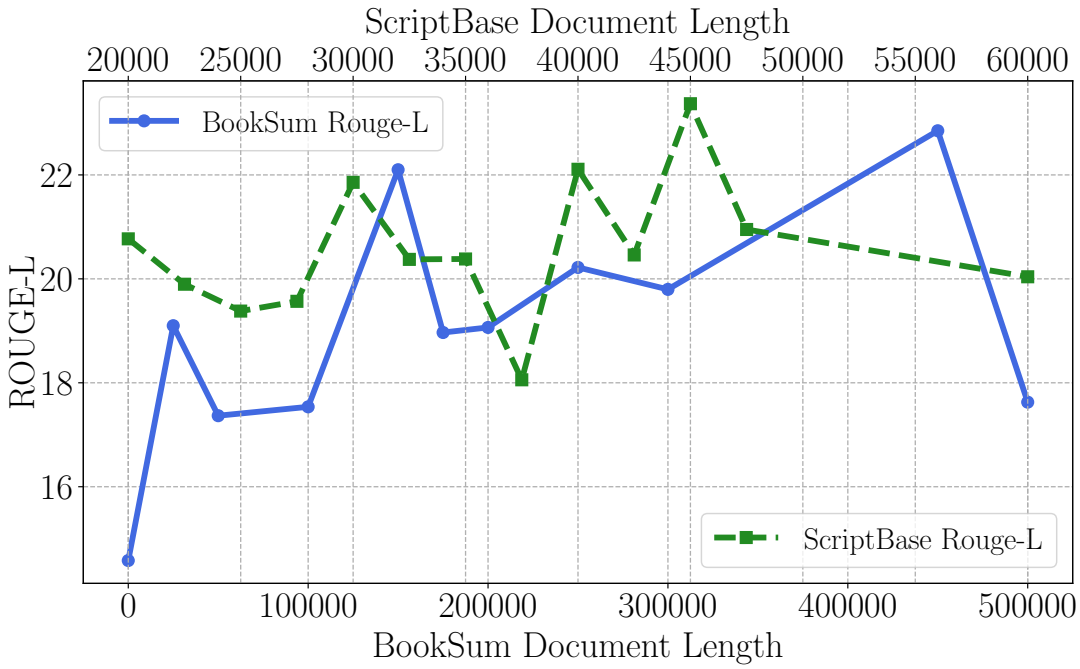


Figure 5.4: ROUGE-L of the summaries generated by CachED $\text{BART}_{\text{large}}$ model for documents of different lengths. The model maintains consistent performance across various document lengths.

5.7 Analyses

In this section, we present analyses of various aspects related to our approach and its performance. For our analysis, we will use the CachED $\text{BART}_{\text{large}}$ model, which performs the best in our experiments. We first examine the utilization of the full context to understand how effectively our method handles and processes long inputs. Next, we analyze the method's performance in relation to document length, assessing how changes in length impact the model's efficiency and accuracy. We also address time and memory usage to evaluate the computational resources required by our approach. Finally, we assess the factual consistency of the model's outputs, ensuring that the generated summaries are not only better in terms of ROUGE but are also factually accurate.

5.7.1 Utilization of the entire document

To evaluate whether CachED models effectively use the full context of a document, we use the alignment between the generated summary and the document as a proxy. We first divide the document into equal segments (bins) and map the summary sentences to

these segments to estimate from which part of the document they were generated. We use ROUGE-L-based alignment (Chen and Bansal, 2018; Zhang et al., 2022), a method previously used for the automatic generation of source-summary pairs. Based on the ROUGE-L scores, we select and map the best segment for each generated summary sentence. This approach allows us to analyze the parts of the document that contribute to the summary.

Figure 5.3 presents the alignment statistics, showing where in a document that a summary sentence aligns to in the BookSum test set. Our results indicate that the CachED $\text{BART}_{\text{large}}$ model does not exhibit significant position bias and utilizes the document uniformly. Overall, the summary sentences align slightly more to the end of a book compared to the first half of the book.

5.7.2 Performance and document length

To further verify the use of full context, we study how sensitive the performance of the CachED $\text{BART}_{\text{large}}$ model is to documents of different lengths. Figure 5.4 shows ROUGE-L scores of our model on the MENSA and BookSum test sets. The ROUGE-L performance remain relatively consistent across documents of different lengths, indicating that our approach is robust in handling extremely long documents.

In the case of the BookSum dataset, the ROUGE-L scores exhibit a stable trend around the mean of 18.8, despite the document lengths reaching up to 500K tokens. Similarly, for the MENSA dataset, the model achieves ROUGE-L scores between 18 and 22, relatively insensitive to the document length.

The consistent performance across varying document lengths shows that the CachED $\text{BART}_{\text{large}}$ model is using the full context when summarizing documents.

5.7.3 Time and Memory Usage

To understand the time and memory consumption, we measure GPU memory and wall-clock time for SLED, Unlimiformer, and our proposed approach across varying numbers of tokens. For a fair comparison, all experiments were performed on a single A100 80GB GPU with the same batch size and full precision.

Figure 5.5 (top) compares GPU memory usage (in GB) as the number of tokens increases. SLED and Unlimiformer show a steep increase in memory consumption as the token count grows. In contrast, CachED $\text{BART}_{\text{large}}$ demonstrates more linear memory scaling. Its memory usage remains below 20GB, even when the input reaches

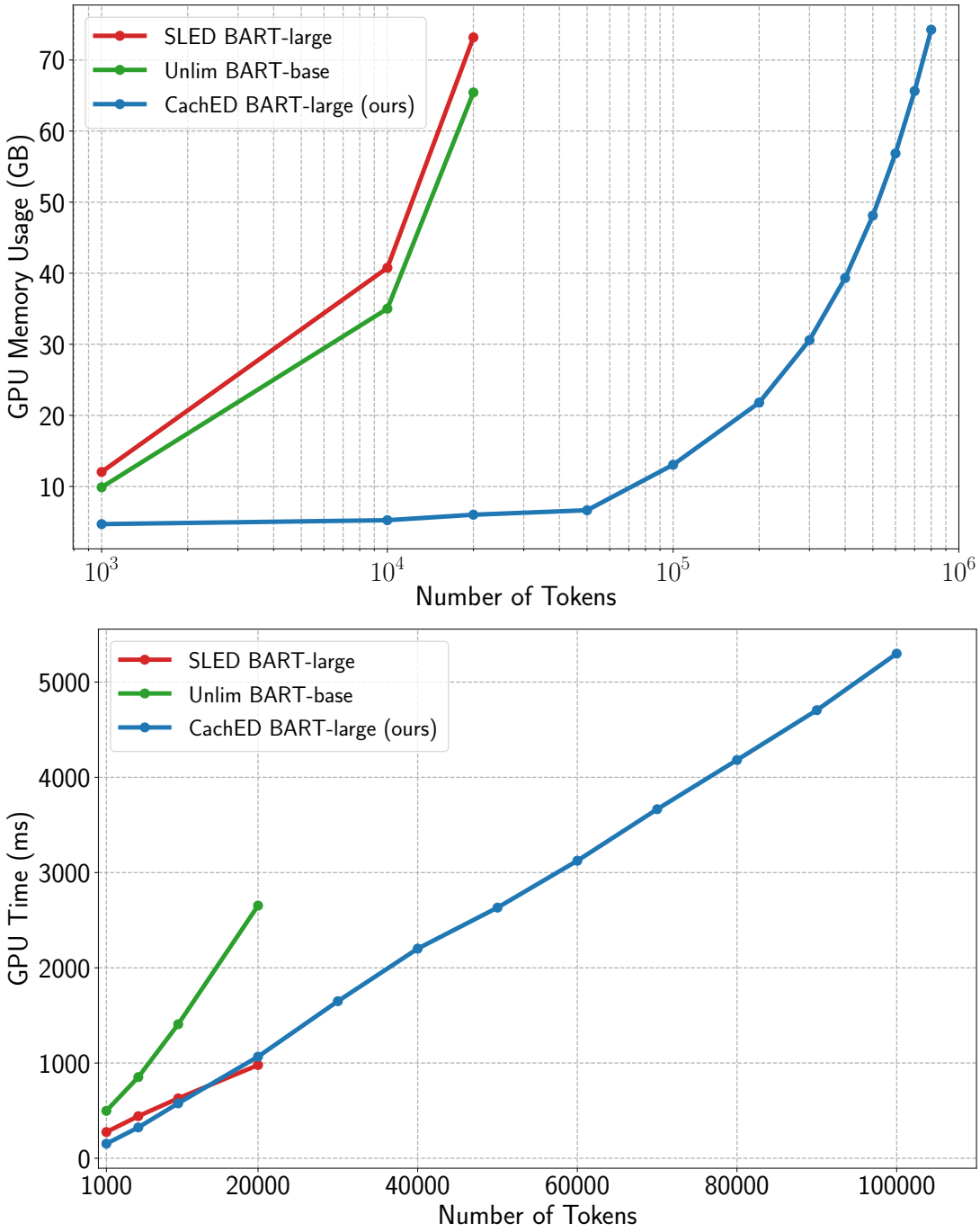


Figure 5.5: Comparison of GPU memory usage (top) and time usage (bottom) for SLED, Unlimiformer, and CachED $\text{BART}_{\text{large}}$.

10^5 tokens. Given a GPU with 80GB of memory, CachED $\text{BART}_{\text{large}}$ can compute gradients up to nearly one million tokens, whereas both SLED and Unlimiformer hit memory limitations at much smaller scales (around 20,000 tokens).

Next, we analyze the impact of feeding forward the encoder twice in our approach. Figure 5.5 (bottom) shows the wall-clock time (in milliseconds) required to process varying token lengths. As the input size grows, Unlimiformer shows a sharp increase in GPU time consumption, likely due to its k-nearest neighbor search operation. In contrast, SLED and CachED maintain a similar linear growth as the token count increases. CachED incurs a slight increase in wall-clock time compared to SLED at around 10,000 tokens, which can be attributed to the recomputation of the encoder during backpropagation. Despite this minor overhead, the processing time remains within a manageable range, ensuring that the model can process extensive texts without huge time overhead. This result confirms our intuition that feeding forward the encoder does not take up much of the time during training compared to other parts of the computation, and that our approach is efficient in practice.

Method	**MENSA**	
	AlignScore	FActScore
Unlim. ($\text{BART}_{\text{base}}$)	31.39	44.00
SLED ($\text{BART}_{\text{large}}$)	32.25	41.10
CachED $\text{BART}_{\text{large}}$	**35.04**	**51.80**
Method	**BookSum**	
	AlignScore	FActScore
Unlim. ($\text{BART}_{\text{base}}$)	35.00	39.30
SLED ($\text{BART}_{\text{large}}$)	36.92	37.70
CachED $\text{BART}_{\text{large}}$	**41.33**	**52.90**

Table 5.3: Results of automatic evaluation of factual consistency on generated summaries for MENSA and BookSum dataset.

5.7.4 Evaluation of Factual Consistency

To evaluate the performance of our method in generating factually correct summaries, we compute the AlignScore (Zha et al., 2023) and FActScore (Min et al., 2023) on Unlimiformer, SLED, and our CachED BART. AlignScore measures factual consistency based on unified information alignment between the context (input document)

and claims (summary sentences). FActScore parses the generated summary into atomic facts and determines whether these facts are supported by a knowledge source. We use the variant proposed by Zha et al. (2023), which uses the gold reference summary as the knowledge source for summary evaluation. We use the GPT-3.5-Turbo model for FActScore and AlignScore-large for AlignScore.

Table 5.3 shows the results for both metrics on summaries generated for the more challenging extremely long document summarization, i.e., on MENSA and BookSum. We compare the results of CachED $BART_{large}$ to Unlimiformer and SLED. Our model generated more factually consistent summaries and substantially outperformed the other models in terms of both the unified alignment of AlignScore and the number of summary facts supported by the gold reference summary measured by FActScore.

MENSA		
Method	**Mean**	**Median**
Gold Summary	878.32	906.5
Unlim. ($BART_{base}$)	856.72	851.0
SLED ($BART_{large}$)	847.54	848.5
CachED $BART_{large}$	844.2	887.0
BookSum		
Gold Summary	859.39	1024.0
Unlim. ($BART_{base}$)	964.7	1022.0
SLED ($BART_{large}$)	995.04	1017.0
CachED $BART_{large}$	847.89	927.0

Table 5.4: Mean and median lengths (in tokens) of gold and generated summaries across the MENSA and BookSum datasets.

5.7.5 Summary length statistics

We also report the summary length statistics to better understand how generated summaries compare to the gold references in terms of length. Table 5.4 presents the mean and median token lengths of the gold summaries and those generated by different models on the MENSA and BookSum datasets. The results show that summaries generated by CachED $BART_{large}$ are generally close in length to the human-written references, indicating that the model generates concise outputs. This is particularly relevant for

MENSA				
Method	**R1**	**R-2**	**R-L**	**BS-F1**
Llama 3.1 8B (ZS)	11.9	2.0	11.2	44.1
MInference (Llama)	22.5	6.3	19.8	50.1
GPT-4o (ZS)	26.6	6.7	**22.5**	52.8
CachED $BART_{large}$	**50.2**	**14.9**	20.4	**64.6**
BookSum				
Llama 3.1 8B (ZS)	13.3	2.9	12.3	41.5
MInference (Llama)	19.7	3.5	16.8	48.8
GPT-4o (ZS)	20.3	3.5	17.7	47.2
CachED $BART_{large}$	**42.8**	**10.5**	**18.8**	**54.4**

Table 5.5: Comparison of our method with GPT-4o and Llama 3.1 8B on extremely long document summarization datasets.

evaluating summary quality, as excessively long summaries can affect the readability and automatic metrics. This also demonstrates that our model produces concise summaries without compromising on factual accuracy.

5.7.6 Comparison with Large Language Models

Table 5.5 compares the performance of our approach, CachED $BART_{large}$, with the large language models GPT-4o (OpenAI et al., 2024) and Llama 3.1 8B (Grattafiori et al., 2024) on two extremely long-document summarization datasets: MENSA and BookSum. Both models were evaluated in a zero-shot setting with a context length of 128K tokens. In addition, we include MInference (Jiang et al., 2024), a training-free sparse attention method that dynamically selects a subset of query–key interactions during the prefill phase, significantly reducing the cost of handling long contexts without requiring retraining. Please refer to Appendix A.5 for prompt templates.

On MENSA, CachED $BART_{large}$ outperforms both GPT-4o and Llama 3.1 8B, achieving the highest ROUGE-1 (50.2), ROUGE-2 (14.9), and BERTScore F1 (64.6), although GPT-4o performs better on ROUGE-L (22.5). Llama-3.1-8B with MInference improves substantially over the base Llama model, suggesting that dynamic sparsification recovers useful long-range dependencies. For the BookSum dataset, CachED $BART_{large}$ achieves the best performance across all metrics, demonstrating a consist-

ent advantage over zero-shot GPT-4o and both Llama 3.1 8B variants. MInference again improves performance for Llama-3.1-8B.

We observed that the large language models we tested occasionally perform well on specific examples, possibly due to memorization, but they frequently fail to generate coherent summaries and often generate repetitive text extracted directly from the source document.

5.7.7 Human Evaluation

We conducted a human evaluation to assess the quality of summaries generated by our model in comparison to baseline approaches. A total of 40 crowdworkers were recruited through Prolific, meeting the following criteria: L1 English speakers, holders of an undergraduate degree, and a minimum of 100 previously approved studies. Participants were compensated at a rate of $17 per hour.

The evaluation focused on four key dimensions: **Fluency**, **Coherence**, **Faithfulness**, and **Relevance**, similar to Fabbri et al. (2021). Each participant rated summaries on a Likert scale ranging from 1 (Poor) to 5 (Excellent) for each dimension. The evaluation was conducted on a sample of 20 summaries from the QMSUM dataset. We selected QMSUM because its reference and generated summaries are relatively shorter than those of other datasets, making it more practical for a focused human evaluation. This allowed for direct comparison of summaries from different models while reducing the cost and complexity of the evaluation process. Full instructions for the evaluation are provided in Appendix A.4. We compared the performance of CachED BART against the top-performing variant of SFT BART, SLED, and Unlimiformer.

Result: The mean scores for each metric across the four models are shown in Figure 5.6. All models performed comparably in Fluency and Coherence with CachED BART achieving slightly higher scores, with means of 3.6 and 3.4, respectively. CachED BART achieved the highest mean scores for Faithfulness (2.6) and Relevance (2.8), outperforming SFT BART, SLED, and Unlimiformer.

Statistical Analysis: We performed a one-way ANOVA to assess whether there were statistically significant differences among the models across the four dimensions. The results showed no significant differences for Fluency ($F = 0.305$, $p = 0.822$) and Coherence ($F = 0.634$, $p = 0.594$), indicating that model performance did not differ

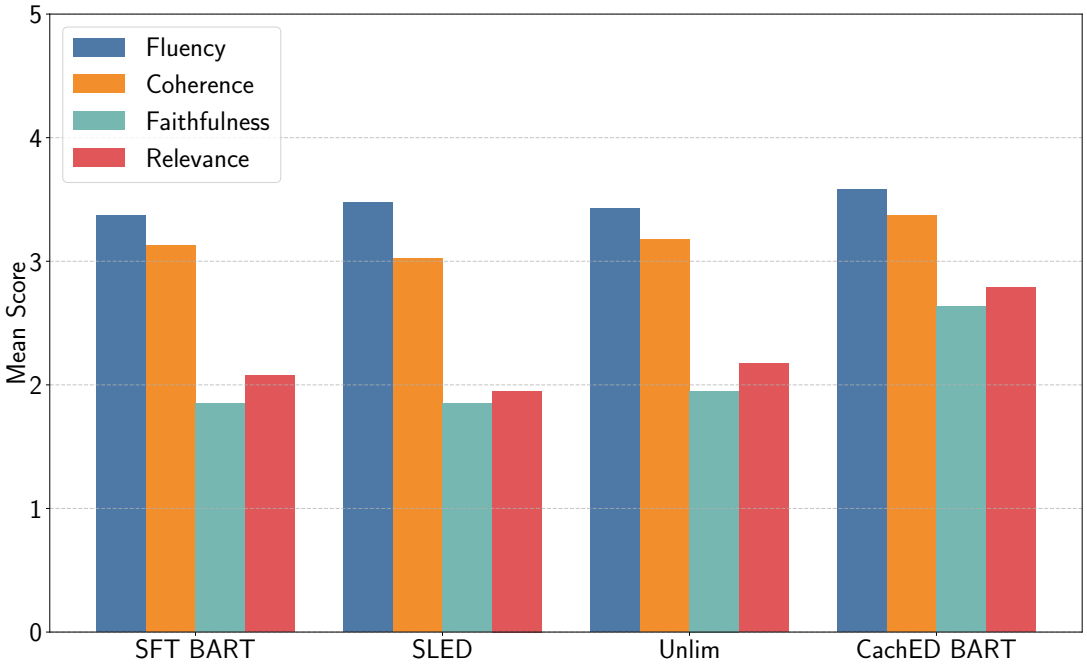


Figure 5.6: Mean scores for Fluency, Coherence, Faithfulness, and Relevance from the human evaluation of generated summaries on the QMSUM dataset. CachED BART achieves the highest scores for all the dimensions outperforming baseline models (SFT BART, SLED, and Unlimiformer).

on these metrics. In contrast, significant differences were observed for Faithfulness ($F = 4.875$, $p = 0.0029$) and Relevance ($F = 4.242$, $p = 0.0065$).

To identify the specific differences among the models, Tukey's HSD post-hoc tests were performed for the significant dimensions. For Faithfulness, CachED BART significantly outperformed SFT BART ($p = 0.0075$), SLED ($p = 0.0075$), and Unlimiformer ($p = 0.0261$). Similarly, for Relevance, CachED BART was significantly better than SFT BART ($p = 0.0288$) and SLED ($p = 0.0066$), while other comparisons did not yield significant differences. These results suggest that CachED BART generates more faithful and relevant summaries compared to the baseline models.

5.8 Summary

In this chapter, we presented CachED, a novel approach for enabling end-to-end training of existing encoder-decoder models for extremely long document summarization by leveraging gradient caching. Our approach efficiently processes long input sequences without truncation, allowing for full-context utilization during both training and inference. The experimental results show that our approach surpasses existing approaches across multiple datasets. We show substantial improvements in ROUGE and BERTScore, even with a smaller $\text{BART}_{\text{base}}$ model, highlighting the effectiveness of our method. Moreover, our approach does not add additional parameters to the model, maintaining the original model's architecture while enhancing its capability to handle long documents.

CachED offers a model-agnostic solution for long-document summarization and opens new directions for improving long-context modeling using existing pretrained architectures. Future research can explore the applicability of our framework to a broader range of models, including decoder-only large language models, and further improve chunk processing strategies to capture more global context information.

Chapter 6

Conclusions and Future Work

In this final chapter, we summarize the contributions of this thesis and discuss the broader implications of our work on long-form narrative summarization. We addressed three key challenges in this domain. First, in Chapter 3, we introduced two new datasets, MovieSum and MENSA, which provide richly formatted movie screenplays and scene-level saliency annotations, respectively. These resources lay a robust foundation for the detailed study of narrative structure and the extraction of salient content from long documents. Second, in Chapter 4, we presented a two-stage summarization framework that leverages explicit content selection via scene saliency detection to improve the performance of abstractive summarization models. Third, in Chapter 5, we introduced CachED, a gradient caching framework that allows encoder-decoder models to be trained on full-length documents without truncation, thereby resolving the mismatch between training and inference conditions.

The work presented here advances the field by (i) expanding the dataset resources available for movie script summarization, (ii) demonstrating that explicit content selection via scene saliency can improve summary quality, and (iii) proposing an efficient training strategy for long-document summarization. In this chapter, we first review and discuss our key contributions and their impact. We then outline several promising directions for future research before concluding with a summary of our overall findings and their implications.

6.1 Summary of Contributions

6.1.1 Novel Datasets: MovieSum and MENSA

A major challenge in narrative summarization is the limited availability of high-quality datasets. To overcome this, we introduced two new resources:

- **MovieSum** comprises 2,200 manually formatted movie screenplays paired with their corresponding Wikipedia plot summaries. This dataset spans a wide range of genres and release years, with each screenplay meticulously formatted using a professional screenplay tool.
- **MENSA (Movie Scene Saliency)** complements MovieSum and focuses on scene-level saliency. It provides fine-grained, gold-standard annotations that align individual scenes from movie screenplays with sentences in the plot summaries. By formalizing scene saliency as a property defined by the presence of a scene in the summary, MENSA enables a fine-grained analysis of narrative structure. The availability of these annotations supports the development and evaluation of content selection models that identify the most important parts of a narrative.

Together, these datasets fill an important gap and offer a robust testbed for narrative summarization.

6.1.2 Content Selection via Scene Saliency

Recognizing that key information in lengthy narratives is often dispersed throughout a document, we proposed a two-stage summarization framework that leverages explicit content selection:

1. **Scene Saliency Classification:** We developed a supervised classification model to predict scene saliency. Using silver-standard labels generated via an alignment method tailored for movie scripts, our model was trained to perform sequence labeling of scene saliency on movie scenes. Our experiments showed that our transformer-based classifier, which incorporates contextualized scene embeddings, outperforms unsupervised baselines such as TextRank and simpler models like Bi-LSTM.

2. **Salient Scene Summarization:** The identified salient scenes serve as input to a pre-trained abstractive summarizer. By reducing the input to only the most critical content, our approach not only improves the relevance and factual consistency of the summaries (as evidenced by higher ROUGE, BERTScore, and QA-based evaluation scores) but also mitigates the challenges posed by extremely long texts.

Our experiments demonstrate that explicit content selection is effective in focusing the summarizer on key narrative scenes, thus producing more coherent and contextually rich summaries.

6.1.3 Efficient Long-Document Training with CachED

Transformer-based encoder-decoder models face significant memory challenges when processing long documents due to the quadratic growth of self-attention. To address this, we introduced CachED (Gradient **Cach**ing for **E**ncoder-**D**ecoder models), a framework that enables end-to-end training on full-length documents without truncation:

- **Chunking and Gradient Caching:** CachED operates by dividing the input document into non-overlapping chunks and performing the forward pass sequentially, caching the encoder outputs. During backpropagation, the model recomputes the hidden states for each chunk, using the cached gradients to update the parameters. This process dramatically reduces peak memory usage, allowing models such as BART and T5 to be trained on documents containing hundreds of thousands of tokens.

- **Plug-and-Play Efficiency:** Our approach can be integrated into existing encoder-decoder models such as BART and T5 without modifying their architectures. Experimental evaluations on datasets like GovReport, SummScreenFD, QMSum, MENSA, and BookSum reveal that CachED not only achieves superior ROUGE and BERTScore metrics but also maintains low memory and time overhead.

This efficient training strategy enables models to utilize the full context of long documents, resulting in improved summarization quality and factual consistency.

Together, these contributions advance the field of long-form narrative summarization by expanding the available resources, improving summarization quality through

explicit content selection, and enabling full-context training without truncation. While saliency-based selection and long-context training may seem to address the same challenge in different ways, they are in fact complementary: saliency helps focus models on narratively important content, whereas CachED ensures that models can exploit the entire document context during training. Future systems can combine these approaches to balance efficiency with coverage, further improving the handling of long and complex narratives.

6.2 Discussion and Findings

Our experiments across multiple datasets revealed several key findings:

Improved Content Selection: The integration of a supervised scene saliency classifier into the summarization pipeline significantly improved summary quality. By selectively focusing on salient narrative scenes, our model produced summaries that more accurately captured critical story elements, as measured by improvements in ROUGE and BERTScore. This result underscores the importance of explicit content selection in long-form narrative summarization, where not all parts of the text contribute equally to the overall story.

Full-Context Processing: The CachED approach enabled our models to process entire documents, preserving the long-range dependencies that are critical for narrative summarization. Traditional summarization models often rely on truncated inputs, which may suffice for news articles but fall short when applied to narratives that extend over hundreds of pages. Movie scripts and books, for instance, contain multiple subplots and character interactions that unfold gradually over time. Truncation in such cases not only leads to the loss of critical information but also disrupts the natural flow of the narrative. Our full-context training method produced summaries with better coverage and improved factual consistency. This demonstrates that maintaining full context during training, despite its computational challenges, is essential for generating high-quality narrative summaries.

Scalability and Efficiency: Although processing full-length documents imposes significant computational demands, our gradient caching strategy reduced memory consumption to manageable levels. The experimental results indicate that even when hand-

ling inputs exceeding 100K tokens, our approach scales effectively without compromising performance. These findings suggest that by using our method, it is possible to overcome the limitations of Transformer-based models and process long narratives end-to-end without truncation.

6.2.1 Limitations

Despite the promising results, several limitations remain. First, our approach relies heavily on the quality of the scene saliency annotations. Errors or inconsistencies in the human-labeled data can propagate through the model and adversely affect the quality of the generated summaries. This issue is compounded by the inherent variability in data quality: narratives from diverse sources often contain inconsistent formatting and noise, which can hinder model training. While our MovieSum and MENSA datasets address some of these challenges, further work is needed to develop robust preprocessing techniques and create higher-quality, standardized datasets across different narrative domains.

Second, in our two-stage summarization pipeline, errors in the scene saliency classifier may cascade into the downstream summarization model, reducing overall quality. Although CachED mitigates memory constraints by processing document chunks sequentially, this comes with additional training time overhead compared to models that operate on shorter contexts end-to-end. Moreover, CachED assumes independence between encoder chunks, leaving long-range contextualization to the decoder; this may limit modeling capacity compared to approaches that propagate context across chunks. Finally, while our method preserves full encoder context, the decoder still scales with the product of input and output length, which remains a computational bottleneck.

Lastly, while our evaluation metrics (e.g., ROUGE, BERTScore, QA-based metrics) provide a useful quantitative measure of performance, they may not fully capture nuanced aspects of narrative summaries such as emotional impact or stylistic coherence. These limitations highlight areas for future research and improvements.

6.3 Future Work

The contributions of this thesis open up several exciting avenues for future research. In this section, we outline potential directions that can further advance the field of long-document summarization and narrative understanding.

6.3.1 Multimodal Summarization

Movie narratives are inherently multimodal, often involving visual and audio cues that provide essential context. Future research could integrate multimodal signals of visual, audio, and textual cues to develop a more holistic model of scene saliency and summary generation. Integrating video frames, soundtrack analysis, or even metadata could help the model better understand the importance of certain scenes (Mahon and Lapata, 2024). For instance, Jiang et al. (2023) exploits pseudo image captions to bridge the gap between visual and textual content, while He et al. (2023) leverage cross-modal alignment with dual contrastive losses to improve multimodal summarization. Incorporating such cross-modal information into our scene saliency framework could lead to richer, more context-aware summaries that better reflect the full narrative. An important challenge for multimodal extensions is that it not only adds additional modalities but also greatly increases context length (e.g., multiple video frames and audio signals over time). Future methods must therefore address long-context modeling across modalities, which compounds the scalability issues already present in purely textual summarization.

6.3.2 Memory-Efficient Training for Decoder-Only Models

Recent work has demonstrated that decoder-only models, such as the GPT family or LLaMA models, exhibit impressive zero-shot capabilities. However, these models still face memory constraints when processing long documents. Techniques like dynamic contextual compression (Qin et al., 2024) and efficient fine-tuning approaches (Zhang et al., 2024) have shown promise in extending the context window of decoder-only models. Adapting gradient caching methods similar to CachED for decoder-only architectures is an exciting direction that could combine the strong generative performance of LLMs with the efficiency required for long-context processing.

6.3.3 Personalized Summarization

An important future direction is the development of summarization systems that can adapt to individual user preferences. Current evaluation metrics often overlook personalization, even though what is salient can vary significantly among users. Vansh et al. (2023) argue that accuracy alone is insufficient for assessing personalization, proposing new metrics such as P-Accuracy to capture user-specific relevance. Additionally,

Ghodratnama and Zakershahrak (2024) presents an interactive framework that learns from user feedback to generate tailored summaries. Integrating personalization into our content selection and summarization framework could yield systems that not only produce high-quality summaries but also adapt dynamically to individual interests, leading to more engaging and useful outputs.

6.3.4 Advanced Evaluation Metrics and Interactive Systems

Despite the success of traditional metrics like ROUGE and BERTScore, these measures have limitations in fully capturing narrative quality, especially in long documents. While ROUGE primarily evaluates lexical overlap and BERTScore assesses semantic similarity, both metrics tend to focus on content coverage. Future work could focus on developing specialized evaluation frameworks that assess how well summaries preserve the core narrative structure, including the progression of key events, the development of character relationships, and causal dependencies within the story. Such narrative-focused metrics could also combine automated measures with human-in-the-loop assessments to account for nuances that are otherwise difficult to quantify.

In conclusion, the work presented in this thesis contributes substantially to the field of narrative summarization by addressing both content selection and long-context processing. By developing robust datasets, leveraging narrative saliency, and introducing novel training methodologies, we have made significant strides toward creating summarization systems that better capture the essence of long, complex narratives. While challenges remain, especially in terms of data quality and scalability, the promising results of this thesis lay a strong foundation for future research. Advances in multimodal and personalized summarization, in particular, hold the potential to further refine these systems, ultimately bridging the gap between human narrative understanding and computational summarization.

Appendix A

Appendix

A.1 Implementation Details of MovieSum

For TextRank, we set the parameter words=1024. We randomly split the dataset into 1800/200/200 as a train/val/test set to train the models. We used the base variants of Pegasus-X, LongT5, and LED for fine-tuning. Each input sequence for the movie is truncated to 16,384 tokens (including special tokens) to fit into the maximum input length of the model. We used AdamW as an optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.99$). For LED and LongT5, we used a learning rate of 2e-5 with a cosine scheduler and a warmup ratio of 0.01. We set the max_new_token to 1024 with greedy decoding for all the experiments. For Pegasus-X, we found that a learning rate of 5e-5 performed better with a linear warmup strategy and a warmup ratio of 0.01. All models were trained for 50 epochs, and the best model was selected using the ROUGE-1 on the validation set. The rest of the configurations for the models were kept as default. All the models were trained on A100 GPU with 80GB memory. We used the Huggingface *evaluate* library for the implementation of the metrics.

A.2 Samples of Movie Summaries

Movie: Lincoln

Gold Summary

In January 1865, United States President Abraham Lincoln expects the Civil War to end soon, with the defeat of the Confederate States. He is concerned that his 1863 Emancipation Proclamation may be discarded by the courts after the war and that the

proposed Thirteenth Amendment will be defeated by the returning slave states. He feels it imperative to pass the amendment beforehand, to remove any possibility that freed slaves might be re-enslaved. The Radical Republicans fear the amendment will be defeated by some who wish to delay its passage; support from Republicans in the border states is not yet assured. The amendment also requires the support of several Democratic congressmen to pass. With dozens of Democrats being lame ducks after losing their re-election campaigns in the fall of 1864, some of Lincoln's advisors believe he should wait for a new Republican-heavy Congress. Lincoln remains adamant about having the amendment in place before the war is concluded and the southern states are re-admitted. Lincoln's hopes rely upon Francis Preston Blair, a founder of the Republican Party whose influence could win over members of the border state conservative faction. With Union victory in the Civil War highly likely but not yet secured, and with two sons serving in the Union Army, Blair is keen to end hostilities quickly before the spring thaw arrives and the armies march again. Therefore, in return for his support, Blair insists that Lincoln allow him to engage the Confederate government in peace negotiations. However, Lincoln knows that significant support for the amendment comes from Radical Republicans, for whom negotiated peace is unacceptable. Unable to proceed without Blair's support, Lincoln reluctantly authorizes Blair's mission. In the meantime, Lincoln and Secretary of State William Seward work to secure Democratic votes for the amendment. Lincoln suggests they concentrate on the lame-duck Democrats, as they will feel freer to vote as they choose and soon need employment; Lincoln will have many federal jobs to fill as he begins his second term. Though Lincoln and Seward are unwilling to offer monetary bribes to the Democrats, they authorize agents to contact Democratic congressmen with offers of federal jobs in exchange for their support. Meanwhile, Lincoln's son, Robert, returns from law school and announces his intention to discontinue his studies and enlist in the Union Army, hoping to earn a measure of honor and respect outside of his father's shadow before the war's end. Lincoln reluctantly secures an officer's commission for Robert. The First Lady is aghast, fearing that he will be killed. She furiously presses her husband to pass the amendment and end the war, promising woe upon him if he should fail. At a critical moment in the debate in the House of Representatives, racial-equality advocate Thaddeus Stevens agrees to moderate his position and argue that the amendment represents only legal equality, not a declaration of actual equality. Meanwhile, Confederate envoys are ready to meet with Lincoln to discuss terms for peace, but he instructs they be kept out of Washington as the amendment approaches a vote on the House

floor. Rumor of their mission circulates, prompting both Democrats and conservative Republicans to advocate postponing the vote. In a carefully worded statement, Lincoln denies there are envoys in Washington, and the vote proceeds, passing by a margin of just two votes. Black visitors to the gallery celebrate, and Stevens returns home to his ”housekeeper” and lover, a black woman. When Lincoln meets with the Confederates, he tells them slavery cannot be restored, as the North is united for ratification of the amendment, and several of the southern states' reconstructed legislatures would also vote to ratify. As a result, the peace negotiations fail, and the war continues. On April 3, Lincoln visits the battlefield at Petersburg, Virginia, where he exchanges a few words with Lieutenant General Ulysses S. Grant. On April 9, Grant receives General Robert E. Lee's surrender at Appomattox Courthouse. On April 14, a cheerful Lincoln expresses to his wife that they will be happy in the future and later meets members of his cabinet to discuss future measures to enfranchise blacks, before leaving for Ford's Theatre. That night, while Lincoln's son Tad is watching Aladdin and the Wonderful Lamp at Grover's Theatre, the manager suddenly stops the play to announce that the President has been shot. The next morning, at the Petersen House, Lincoln dies with a peaceful expression across his face; in a flashback, Lincoln finishes his second inaugural address on March 4.

Question	**Answer**
What city are Confederate envoys ready to meet with Lincoln to discuss terms for peace?	Washington
As a result of the failed peace negotiations, what continues?	war
On April 3, who visits the battlefield at Petersburg, Virginia, where he exchanges a few words with Lieutenant General Ulysses S. Grant?	Lincoln
What approaches a vote on the House floor while Lincoln keeps Confederate envoys out of Washington?	the amendment
What are Confederate envoys trying to negotiate with Lincoln before the amendment vote?	peace
What does Lincoln deny exist in Washington through carefully worded statements?	envoys

Table A.1: Gold reference summary for the movie *Lincoln* with sample question-answer pairs for evaluation. Correct answers shown in green.

Model Generated Summary

In 1863, U.S. President Abraham Lincoln and his Cabinet prepare for war against the Confederate States of America. Secretary of War Edwin M. Stanton issues a war proclamation declaring slaves in the Confederate States to be property confiscated during the American Civil War. Drawing on his war powers, Lincoln declares that slaves in the thirteen southern states are property of the Union, and that he can confiscate them and free them with his 1863 Emancipation Proclamation. This, however, is challenged by the states' refusal to ratify the Thirteenth Amendment to the United States' Constitution, which would end slavery. The courts refuse to recognize the document, which would ban slavery, forcing Lincoln to act by proclamation. He reluctantly authorizes the Emancipation Proclamation, but suspends its implementation pending a vote in Congress. The Southern states refuse to ratify the amendment, preventing it from being put before a vote until they are readmitted to the Union. Only two years later, on Good Friday, Lincoln takes his wife, Mary Todd Lincoln, out for a buggy ride. They talk about plans to travel to the Holy Land and how much they want to get out of Washington, D.C. Before returning to the White House, they view a picture of Tad Lincoln, Abraham Lincoln's son, with his mistress, Mary Keckley. Mary reveals that she was a slave, and that her husband was killed by their son. Lincoln is deeply disturbed by this revelation. A year later, on April 4, 1865, the southern states are readmitted to the Union, but they are not readmitted to block ratification of the Thirteenth Amendment. Secretary of the Navy Gideon Welles issues a report that the southern states have refused to accede to the amendment, preventing it from being ratified. This leads to friction between Lincoln's pro-slavery advisers and the pro-slavery delegation in Congress. They attempt to convince the reluctant Wood, Baird, and Blair to delay a vote on the amendment in the House until they have information from the President on the whereabouts of the Confederate States' commissioners, who are supposed to have arrived in Washington, D.C., to negotiate a solution to end the war. The White House attempts to find evidence of the commissioners' arrival, but they rely on affidavits from loyal citizens recently returned from Richmond, Virginia. Meanwhile, Lincoln's popularity continues to rise, and he continues to insist on action on the war. Privately, he expresses his hope that the war will end soon and they can begin a new life together, but his Vice President, Jefferson Davis, persuades him to delay the vote on the amendment, hoping to secure enough votes in the Senate to block ratification. When the Senate fails to act on the amendment, and the House refuses to act, Lincoln

is forced to take action, forcing a shutdown of the government. Seward urges Lincoln to declare the war over and negotiate a peace, but Lincoln refuses. Seward pressures him to declare a truce, which would allow the states to resume their former position in the Union and block ratification of the Thirteenth Amendment. On April 5, 1865, the war ends at Appomattox Courthouse in Appomattox, Virginia. Confederate General Robert E. Lee surrenders to General Ulysses S. Grant at Appomattox. A great deal of animosity remains between Lincoln and Lee, and the two men ride together to the McLean House, where the wounded are brought to view the final Confederate corpses from the Appomattox Massacre. When Lee approaches the McLeans, Grant stops him, takes his hat, and salutes him. Lee is visibly moved by this gesture of respect. The film ends with Lincoln and Mary walking arm in arm to the edge of the balcony of the McLean House, looking down on the battlefield.

Question	**Answer**
What city are Confederate envoys ready to meet with Lincoln to discuss terms for peace?	Washington
As a result of the failed peace negotiations, what continues?	war
On April 3, who visits the battlefield at Petersburg, Virginia, where he exchanges a few words with Lieutenant General Ulysses S. Grant?	Lincoln
What approaches a vote on the House floor while Lincoln keeps Confederate envoys out of Washington?	Thirteenth Amendment
What are Confederate envoys trying to negotiate with Lincoln before the amendment vote?	end the war
What does Lincoln deny exist in Washington through carefully worded statements?	commission-ers

Table A.2: Model-generated summary of the movie *Lincoln* with answer validation. Correct answers shown in green, incorrect in red, with green/red combinations indicating partially correct responses.

Movie: Black Panther

Gold Summary

Thousands of years ago, five African tribes war over a meteorite containing the metal vibranium. One warrior ingests a ”heart-shaped herb” affected by the metal and gains superhuman abilities, becoming the first ”Black Panther”. He unites all but the Jabari Tribe to form the nation of Wakanda. Over centuries, the Wakandans use the vibranium to develop advanced technology and isolate themselves from the world by posing as a Third World country. In 1992, Wakanda king T’Chaka visits his brother N’Jobu, who is working undercover in Oakland, California. T’Chaka accuses N’Jobu of assisting black-market arms dealer Ulysses Klaue with stealing vibranium from Wakanda. N’Jobu’s partner reveals he is Zuri, another undercover Wakandan, and confirms T’Chaka’s suspicions. In the present day, following T’Chaka’s death, his son T’Challa returns to Wakanda to assume the throne. He and Okoye, the leader of the Dora Milaje regiment, extract T’Challa’s ex-lover Nakia from an undercover assignment so she can attend his coronation ceremony with his mother Ramonda and younger sister Shuri. At the ceremony, the Jabari Tribe’s leader M’Baku challenges T’Challa for the crown in ritual combat. T’Challa defeats M’Baku and persuades him to yield rather than die. When Klaue and his accomplice Erik Stevens steal a Wakandan artifact from a London museum, T’Challa’s friend and Okoye’s lover W’Kabi urges him to bring Klaue back alive. T’Challa, Okoye, and Nakia travel to Busan, South Korea, where Klaue plans to sell the artifact to CIA agent Everett K. Ross. A firefight erupts, and Klaue attempts to flee but is caught by T’Challa, who reluctantly releases him to Ross’ custody. Klaue tells Ross that Wakanda’s international image is a front for a technologically advanced civilization. Erik attacks and extracts Klaue as Ross is gravely injured protecting Nakia. Rather than pursue Klaue, T’Challa takes Ross to Wakanda, where their technology can save him. While Shuri heals Ross, T’Challa confronts Zuri about N’Jobu. Zuri explains that N’Jobu planned to share Wakanda’s technology with people of African descent around the world to help them conquer their oppressors. As T’Chaka arrested N’Jobu, the latter attacked Zuri and forced T’Chaka to kill him. T’Chaka ordered Zuri to lie that N’Jobu had disappeared and left behind N’Jobu’s American son to maintain the lie. This boy grew up to be Stevens, a black ops U.S. Navy SEAL who adopted the name ”Killmonger”. Meanwhile, Killmonger kills Klaue and takes his body to Wakanda. He is brought before the tribal elders, revealing his identity to be N’Jadaka and stating his claim to the throne. Killmonger challenges

T'Challa to ritual combat, where he kills Zuri, defeats T'Challa, and hurls him over a waterfall to his presumed death. Killmonger ingests the heart-shaped herb and orders the rest incinerated, but Nakia extracts one first. Killmonger, supported by W'Kabi and his army, prepares to distribute shipments of Wakandan weapons to operatives around the world. Nakia, Shuri, Ramonda, and Ross flee to the Jabari Tribe for aid. They find a comatose T'Challa, rescued by the Jabari in repayment for sparing M'Baku's life. Healed by Nakia's herb, T'Challa returns to fight Killmonger, who dons his own nanotech suit, similar to T'Challa's. W'Kabi and his army fight Shuri, Nakia, and the Dora Milaje, while Ross remotely pilots a jet and shoots down planes carrying the vibranium weapons. M'Baku and the Jabari arrive to reinforce T'Challa. Confronted by Okoye, W'Kabi and his army stand down. Fighting in Wakanda's vibranium mine, T'Challa disrupts Killmonger's suit and stabs him. Killmonger refuses to be healed, choosing to die a free man rather than be incarcerated; T'Challa takes him to the waterfall where they fought, where Killmonger dies peacefully. T'Challa establishes an outreach center at the building where N'Jobu died, to be run by Nakia and Shuri. In a mid-credits scene, T'Challa appears before the United Nations to reveal Wakanda's true nature to the world.

Question	**Answer**
Thousands of years ago, five African tribes war over a meteorite containing what metal?	vibranium
Where do T'Challa and Okoye travel to South Korea where Klaue plans to sell the artifact?	Busan
What does one warrior ingest affected by the metal and gain superhuman abilities?	heart-shaped herb
Wakandans use the vibranium to develop advanced technology and isolate themselves from what?	the world
Who do T'Challa, Okoye, and Nakia travel to Busan to sell the artifact to?	Everett K. Ross
While Shuri heals Ross, who does T'Challa confront about N'Jobu?	Zuri

Table A.3: Gold reference summary for the movie *Black Panther* with sample question-answer pairs generated for evaluation.

Generated Summary

After the death of his father King T'Chaka at the United Nations, Prince T'Challa of the African country of Wakanda adopts the title "Black Panther" and assumes the name T'Challa. A meteorite lands on Africa, striking the continent of Africa, and five tribes settle on it including Wakanda. The native people of Wakanda named Wakanda "Wakanda". The people live in harmony with each other and use vibranium, a mineral that gives them superhuman strength, speed, and other powers. A warrior from Wakanda named N'Jobu uses vibranium to make the Heart Shaped Herb, which gives his son T'Challa his powers. A CIA agent, Eric Stevens, traces the Heart Shaped Herb to Ulysses Klaue, a black marketeer attempting to obtain vibranium. T'Challa's aunt and uncle, Nakia and Okoye, lead a mission to retrieve vibranium from Klaue at the request of the African Dora Milaje. They capture Klaue and bring him to T'Challa at the Great Mound. T'Challa challenges Klaue for the throne of Wakanda. The meeting is interrupted by the arrival of Young T'Chaka, T'Challa's half-brother and the son of King T'Chaka. T'Challa takes Klaue on a train trip back to Wakanda with the Dora Milaje, but Klaue injures Ross, a CIA agent. T'Challa and Ross confront Klaue at the train station, and Klaue declares his intention to sell the vibranium to an American buyer in South Korea. T'Challa's aunt and uncle, Nakia and Okoye, tell him that if he sells the vibranium, the Dora Milaje will recover the money and return it to Wakanda. T'Challa agrees to this, and they fly to South Korea to retrieve the Heart Shaped Herb. At the U.S. embassy in Busan, South Korea, they learn that Klaue is actually an operative of the CIA, and that the Heart Shaped Herb has been smuggled into Wakanda by an unknown party. T'Challa and Ross confront Klaue in a casino. Klaue threatens to kill Ross if T'Challa does not hand over the vibranium. T'Challa offers Ross a deal: if he delivers the vibranium, he will be allowed to return to Wakanda. Ross refuses, and Klaue attempts to kill T'Challa, but is stopped by Young T'Challa. T'Challa heals Ross, and the two embark to return to Wakanda. Along the way, Ross reveals himself to T'Challa as an operative of the CIA, and T'Challa takes him back to the CIA. T'Challa and Nakia arrive in Wakanda, where they meet Nakia's mother, Queen Mother Ramonda, and the Dora Milaje. They purchase vibranium from a merchant at the port of Busan. T'Challa presents the Heart Shaped Herb to the people of Wakanda, and Nakia gives birth to a son, T'Challa born T'Chaka. T'Challa and Nakia take the Heart Shaped Herb to Jabariland, where it is discovered that Nakia is a spy for the CIA. The Heart Shaped Herb gives T'Challa superhuman strength and other

powers. He goes to Jabariland to find Nakia and finds her serving as a midwife to a dying M'Baku. M'Baku reveals himself to be a spy sent by the CIA to find T'Challa and is confronted by T'Challa. A gunfight ensues, in which T'Challa gains the upper hand, but M'Baku seizes the vibranium and attempts to kill T'Challa. T'Challa uses the Heart Shaped Herb to keep M'Baku at bay, and returns to Wakanda with Nakia, Ross, and the Heart Shaped Herb to deliver T'Challa to his people. Nakia betrays T'Challa to the CIA, and Ross is sent to retrieve the vibranium. T'Challa and Nakia escape with the Heart Shaped Herb and take it to Jabariland, where it is recovered by W'Kabi, the leader of the Jabari tribe. The tribe's leader, W'Kabi, uses it to heal T'Challa, and he returns to Wakanda with T'Challa and the Heart Shaped Herb to reunite the tribes. In the present, T'Challa and Nakia celebrate the opening of Wakanda's new center for science and information exchange with the rest of the world.

Question	**Answer**
Thousands of years ago, five African tribes war over a meteorite containing what metal?	vibranium
Where do T'Challa and Okoye travel to South Korea where Klaue plans to sell the artifact?	Busan
What does one warrior ingest affected by the metal and gain superhuman abilities?	heart shaped herb
Wakandans use the vibranium to develop advanced technology and isolate themselves from what?	United Nations
Who do T'Challa, Okoye, and Nakia travel to Busan to sell the artifact to?	American buyer
While Shuri heals Ross, who does T'Challa confront about N'Jobu?	M'Baku

Table A.4: Model-generated summary of the movie *Black Panther* with answer validation. Correct answers are shown in green, incorrect in red, and mixed responses show both colors for partial correctness.

A.3 CachED Implementation Details

All experiments were performed on an single A100 GPU with 80GB memory. GovReport was trained for 10 epochs, QMSUM for 20 epochs, and MENSA, BookSum, and

Parameter	Value
Learning rate	1e-5
Optimizer	AdamW (Kingma and Ba, 2015)
AdamW β_1	0.9
AdamW β_2	0.99
batch size	1
Effective batch size	2
Warmup Strategy	linear
Warmup Steps	1024
Decoding Strategy	Greedy
Sample	False
Beam size	4
Chunk Size (BART)	1024
Chunk Size (T5)	512

Table A.5: Hyperparameter values used for our experiments.

SummscreenFD were trained for 15 epochs. For evaluation, we used ROUGE metric Perl package [1] and BERTScore with the microsoft/deberta-xlarge-mnli model.

For MInference, we follow the public HF integration (dynamic sparse attention during prefill only), keeping decoding dense; all prompts, decoding parameters, and maximum context (128K) are identical to the dense Llama 3.1 runs.

A.4 Instructions for Human Evaluation

In this task, you will assess the quality of computer-generated summaries by comparing them against gold (reference) summaries. You will be provided with a **Gold Summary** and a corresponding **Generated Summary**.

Your task is to evaluate the generated summary on the following four dimensions: **Fluency**, **Coherence**, **Relevance**, and **Faithfulness**.

You will assign a score between **1 (Poor)** and **5 (Excellent)** for each dimension. **Dimensions of Evaluation:**

[1] https://github.com/summanlp/evaluation/tree/ master/ROUGE-RELEASE-1.5.5

1. Fluency: This dimension evaluates whether the generated summary is grammatically correct, easy to read, and well-structured.

2. Coherence: This dimension assesses whether the sentences in the generated summary flow logically and maintain a consistent narrative.

3. Faithfulness: This dimension checks if all the facts presented in the generated summary are accurate and can be directly inferred from the gold summary.

4. Relevance: This dimension evaluates whether all the important facts from the gold summary are present in the generated summary.

Task Instructions: For each gold summary and the corresponding generated summary:

1. Rate the generated summary for each dimension (Fluency, Coherence, Faithfulness, Relevance) on a scale of 1 to 5 based on the definitions provided.
2. Ignore minor grammatical errors or abrupt endings that do not significantly impact the overall quality of the summary.
3. Make your selection based solely on the definitions and examples provided for each dimension.

A.5 Prompt Template

The prompt templates used in our experiments.

Prompt Template for LLaMA 3.1 8B

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
You are a helpful assistant<|eot_id|>
<|start_header_id|>user<|end_header_id|>
Summarize the following text:[Input Text]
<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

Prompt Template for GPT-4

```
Summarize the following text.\n[Input Text]
```

A.6 Sample of Book Summaries

We present sample summaries of the book **Sense and Sensibility** generated using SLED, Unlimiformer, and our approach CachED BART.

A.6.1 Gold Summary

The Dashwood family is introduced; Mr. and Mrs. Dashwood and their three daughters live at Norland Park, an estate in Sussex. Unfortunately, Mr. Dashwood's wife and daughters are left with very little when he dies and the estate goes to his son, John Dashwood. John and his wife Fanny have a great deal of money, yet refuse to help his half-sisters and their mother. Elinor, one of the Dashwood girls, is entirely sensible and prudent; her sister, Marianne, is very emotional and never moderate. Margaret, the youngest sister, is young and good-natured. Mrs. Dashwood and her daughters stay at Norland for a few months, mostly because of the promising friendship developing between Elinor and Edward Ferrars, Fanny's shy, but very kind, brother. Elinor likes Edward, but is not convinced her feelings are mutual; Fanny is especially displeased by their apparent regard, as Edward's mother wants him to marry very well. A relative of Mrs. Dashwood's, Sir John Middleton, offers them a cottage at Barton Park in Devonshire; the family must accept, and are sad at leaving their home and having to separate Edward and Elinor. They find Barton Cottage and the countryside around it charming, and Sir John Middleton a very kind and obliging host. His wife, Lady Middleton, is cold and passionless; still, they accept frequent invitations to dinners and parties at Barton Park. The Dashwoods meet Mrs. Jennings, Sir John's mother-in-law, a merry, somewhat vulgar older woman, and Colonel Brandon, a gentleman and a bachelor. The Colonel is soon taken with Marianne, but Marianne objects to Mrs. Jennings attempts to get them together, and to the "advanced" age and serious demeanor of the Colonel. Marianne falls and twists her ankle while walking; she is lucky enough to be found and carried home by a dashing man named Willoughby. Marianne and Willoughby have a similar romantic temperament, and Marianne is much pleased to find that Willoughby has a passion for art, poetry, and music. Willoughby and Marianne's attachment develops steadily, though Elinor believes that they should be more restrained in showing their regard publicly. One pleasant day, the Middletons, the Dashwoods, and Willoughby are supposed to go on a picnic with the Colonel, but their plans are ditched when Colonel Brandon is forced to leave because of distressing news. Willoughby becomes an even more attentive guest at the cottage, spending a great deal more time there than

Allenham with his aunt. Willoughby openly confesses his affections for Marianne and for all of them, and hopes they will always think of him as fondly as he does of them; this leaves Mrs. Dashwood and Elinor convinced that if Marianne and Willoughby are not engaged, they soon will be. One morning, Mrs. Dashwood, Elinor, and Margaret leave the couple, hoping for a proposal; when they return, they find Marianne crying, and Willoughby saying that he must immediately go to London. Mrs. Dashwood and Elinor are completely unsettled by this hasty departure, and Elinor fears that they might have had a falling-out. Marianne is torn up by Willoughby's departure, and Elinor begins to question whether Willoughby's intentions were honorable. But, whether Willoughby and Marianne are engaged remains a mystery, as Marianne will not speak of it. Edward comes to visit them at Barton, and is welcomed very warmly as their guest. It is soon apparent that Edward is unhappy, and doesn't show as much affection for Elinor; when they spot a ring he is wearing, with a lock of hair suspiciously similar to Elinor's, even Elinor is baffled. Edward finally forces himself to leave, still seeming distressed. Sir John and Mrs. Jennings soon introduce Mrs. Jennings' other daughter, Mrs. Palmer, and her husband to the family. Mrs. Palmer says that people in town believe that Willoughby and Marianne will soon be married, which puzzles Elinor, as she knows of no such arrangements herself. Elinor and Marianne meet the Middletons' new guests, the Miss Steeles, apparently cousins; they find Miss Steele to be nothing remarkable, while Lucy is very pretty but not much better company. However, the Miss Steeles instantly gain Lady Middleton's admiration by paying endless attention to her obnoxious children. Elinor, unfortunately, becomes the preferred companion of Lucy. Lucy inquires of Mrs. Ferrars, which prompts Elinor to ask about her acquaintance with the Ferrars family; Lucy then reveals that she is secretly engaged to Edward. It turns out that Edward and Lucy knew each other while Edward studied with Lucy's uncle, Mr. Pratt, and have been engaged for some years. Although Elinor is first angry about Edward's secrecy, she soon sees that marrying Lucy will be punishment enough, as she is unpolished, manipulative, and jealous of Edward's high regard for Elinor. The Miss Steeles end up staying at Barton Park for two months. Mrs. Jennings invites Marianne and Elinor to spend the winter with her in London. Marianne is determined to go to see Willoughby, and Elinor decides she must go too, because Marianne needs Elinor's polite guidance. They accept the invitation, and leave in January. Once in town, they find Mrs. Jennings' house comfortable, and their company less than ideal; still, they try their best to enjoy it all. Marianne anxiously awaits Willoughby's arrival, while Elinor finds her greatest enjoyment in Colonel Brandon's daily visits. Elinor is much

disturbed when Colonel Brandon tells her that the engagement between Marianne and Willoughby is widely known throughout town. At a party, Elinor and Marianne see Willoughby; Marianne approaches him, although he avoids Marianne, and his behavior is insulting. Marianne angrily writes Willoughby, and receives a reply in which he denies having loved Marianne, and says he hopes he didn't lead her on. Marianne is deeply grieved at being deceived and dumped so coldly; Elinor feels only anger at Willoughby's unpardonable behavior. Marianne then reveals that she and Willoughby were never engaged, and Elinor observes that Marianne should have been more prudent in her affections. Apparently, Willoughby is to marry the wealthy Lady Grey due to his constant need for money. Colonel Brandon calls after hearing the news, and offers up his knowledge of Willoughby's character to Elinor. Colonel Brandon was once in love with a ward to his family, Eliza, who became a fallen woman and had an illegitimate daughter. Colonel Brandon placed the daughter, Miss Williams, in care after her mother's death. The Colonel learned on the day of the Delaford picnic that she had become pregnant, and was abandoned by Willoughby. Elinor is shocked, though the Colonel sincerely hopes that this will help Marianne feel better about losing Willoughby, since he was not of solid character. The story convinces Marianne of Willoughby's guilt, though it does not ease her mind. Out of sympathy, Marianne also stops avoiding the Colonel's company and becomes more civil to him. Willoughby is soon married, which Marianne is grieved to hear; then, again unfortunately, the Miss Steeles come to stay with the Middletons. John and Fanny Dashwood arrive, and are introduced to Mrs. Jennings, and to Sir John and Lady Middleton, deeming them worthy company. John reveals to Elinor that Edward is soon to be married to Miss Morton, an orphan with a great deal of money left to her, as per the plans of his mother. At a dinner party given by John and Fanny for their new acquaintance, Mrs. Ferrars is present, along with the entire Barton party. Mrs. Ferrars turns out to be sallow, unpleasant, and uncivil; she slights Elinor, which hurts Marianne deeply, as she is Edward's mother. The Miss Steeles are invited to stay with John and Fanny. But, Mrs. Jennings soon informs them that Miss Steele told Fanny of Lucy and Edward's engagement, and that the Ferrars family threw the Steele girls out in a rage. Marianne is much grieved to hear of the engagement, and cannot believe that Elinor has also kept her knowledge of it a secret for so long. Edward is to be disinherited if he chooses to marry Lucy; unfortunately, Edward is too honorable to reject Lucy, even if he no longer loves her. Financial obstacles to their marriage remain; he must find a position in the church that pays enough to allow them to marry. Much to Elinor's chagrin, the Colonel, although he barely knows

Edward, generously offers the small parish at Delaford to him. Elinor is to convey the offer to Edward, though she regrets that it might help the marriage. Edward is surprised at the generous offer, since he hardly knows the Colonel. Edward decides to accept the position; they say goodbye, as Elinor is to leave town soon. Much to Elinor's surprise, Robert Ferrars, Edward's selfish, vain, and rather dim brother, is now to marry Miss Morton; he has also received Edward's inheritance and money, and doesn't care about Edward's grim situation. It is April, and the Dashwood girls, the Palmers, and Mrs. Jennings, and Colonel Brandon set out for Cleveland, the Palmer's estate. Marianne is still feeling grief over Willoughby; she soon becomes ill after her walks in the rain, and gets a serious fever. The Palmers leave with her child; Mrs. Jennings, though, helps Elinor nurse Marianne, and insists that Colonel Brandon stay, since he is anxious about Marianne's health. Colonel Brandon soon sets off to get Mrs. Dashwood from Barton when Marianne's illness worsens. At last, Marianne's state improves, right in time for her mother and the Colonel's arrival; but Willoughby makes an unexpected visit. Elinor is horrified at seeing him; he has come to inquire after Marianne's health and to explain his past actions. Willoughby says he led Marianne on at first out of vanity; he finally began to love her as well, and would have proposed to her, if not for the money. By saying that he also has no regard for his wife, and still loves Marianne, he attempts to gain Elinor's compassion; Elinor's opinion of him is somewhat improved in being assured of his regard for Marianne. Elinor cannot think him a total blackguard since he has been punished for his mistakes, and tells him so; Willoughby leaves with this assurance, lamenting that Marianne is lost to him forever. Mrs. Dashwood finally arrives, and Elinor assures her that Marianne is out of danger; both Mrs. Dashwood and the Colonel are relieved. Mrs. Dashwood tells Elinor that the Colonel had confessed his love for Marianne during the journey from Barton; Mrs. Dashwood wishes the Colonel and Marianne to be married. Elinor wishes the Colonel well in securing Marianne's affections, but is more pessimistic regarding Marianne's ability to accept the Colonel after disliking him for so long. Marianne makes a quick recovery, thanking Colonel Brandon for his help and acting friendly toward him. Marianne finally seems calm and happy as they leave for Barton, which Elinor believes to signal Marianne's recovery from Willoughby. She is also far more mature, keeping herself busy and refusing to let herself languish in her grief. When Marianne decides to talk about Willoughby, Elinor takes the opportunity to tell her what Willoughby had said at Cleveland, and Marianne takes this very well. Marianne also laments her selfishness toward Elinor, and her lack of civility toward most of their acquaintance. Marianne finally says that she could not

have been happy with Willoughby, after hearing of his cruelty toward Miss Williams, and no longer regrets him. The family is stunned when one of their servants returns with news that Edward is married to Lucy, as he just saw them in the village. Elinor knows now that Edward is lost to her forever. Mrs. Dashwood sees how upset Elinor is, and realizes that Elinor felt more for Edward than she ever revealed. One afternoon, Elinor is convinced that the Colonel has arrived at the cottage, but is surprised to find that it is Edward instead. Their meeting is awkward at best; he soon informs them that it is his brother who has been married to Lucy, and not him. Elinor immediately runs from the room, crying out of joy; Edward then senses Elinor's regard for him, and proposes to her that afternoon. Elinor accepts and he gains Mrs. Dashwood's consent to the match. Edward admits that any regard he had for Lucy was formed out of idleness and lack of knowledge; he came to regret the engagement soon after it was formed. After leaving London, Edward received a letter from Lucy saying that she had married his brother Robert, and has not seen her since; thus, he was honorably relieved of the engagement. After receiving the letter, he set out for Barton immediately to see Elinor. Edward will still accept the position at Delaford, although he and Elinor again will not have enough money to live on comfortably. The Colonel visits Barton, and he and Edward become good friends. Edward then becomes reconciled with his family, although he does not regain his inheritance from Robert. His mother even gives her consent for his marriage to Elinor, however much she is displeased by it; she gives them ten thousand pounds, the interest of which will allow them to live comfortably. Edward and Elinor are married at Barton that fall. Mrs. Dashwood and her two remaining daughters spend most of their time at Delaford, both to be near Elinor, and out of the hope that Marianne might accept the Colonel. In the two years that have passed, Marianne has become more mature and more grounded; and she does finally change her mind about the Colonel, and accepts his offer of marriage. The Colonel becomes far more cheerful, and soon Marianne grows to love him as much as she ever loved Willoughby. Mrs. Dashwood remains at Barton with Margaret, now fifteen, much to the delight of Sir John, who retains their company. And Elinor and Marianne both live together at Delaford, and remain good friends with each other and each other's husbands.

A.6.2 Generated Summary using CachED BART

Marianne and Elinor Dashwood, two young cousins of the Dashwood family, live with their mother and sister at their cottage in Norland Park, a small country estate in Sussex.

The Dashwoods are not wealthy, but they are in good financial circumstances due to the large estate and the money their uncle left them. Marianne and her sister Elinor, rather, Marianne Dashwood are the only children of a wealthy uncle and aunt. Marianne and Elinor are close in age, but Elinors' relationship with Edward Ferrars, the Dashwoods' brother, is more complicated. Edward is the son of a woman who is not wealthy but who has a reputation for being selfish and unworldly. They are in love, but Edward is not interested in them. They are both too young to be married, and they are both unaware of their parents' financial situation. Marianna and Elinor are invited to visit the Dashwars at their home, Barton Park, by their cousin, Sir John Middleton. They spend a pleasant time there, but Marianne is not happy there. She is attracted to Edward, but he is not attracted to her. Marianne's mother, Mrs. Jennings, is not pleased with Marianne's relationship with her son, and she is not impressed by Elinors relationship to Edward. Marianne is also upset that Elinor has been invited to stay at Barton Park. In the meantime, Elinor Dashwood's cousin, John Dashwood has been visiting from London. John is a friend of the family, and is a good-looking young man. He is the brother-in-law to Elinor to Elinsor's sister, Marianne. John and El inor are friends, and have been friends for years. John has been engaged to Elina's sister Lucy, but she has not yet told him she is engaged to Edward Ferrar. John's mother is not at all happy about the match, and John is not sure if he should tell Elinor about it. Marianne does not want to upset her sister, but is happy to hear that Elina has been in London. Elinor and John are friends and neighbors of the Middleton family. John Dashwood and Marianne are friends of the Jenningses, and the Dash wer happy to be friends with them. John, Elinor, and Marianne are all friends of Sir John's, and Elinor is close to them. When John's sister dies, he and Elinor decide to visit her in town. They go to visit John's house, where they are met by Willoughby, Elinor's brother- in-law. Elinor and John Dashwards are friends for a while, and are happy to see each other. John also knows that Willoughby is engaged, but Willoughby is not ready to announce it. Elinor or John Dash waffles on whether to tell Marianne about it, and when Marianne finds out, she is devastated. She decides to keep it a secret from Elinoor and John, and to keep Willough by herself. Marianne becomes ill, and after a few days, Elinor in or out of town with John Dashwalts, Ellinor decides to tell John about Willoughy. John comes to visit, and he and Mariane are very happy to meet Willoughby. It turns out that Willougby is engaged. Marianne tells Elinor that Willohby is married to Lucy, and that she is happy for him. Ellinar is happy that she has been able to keep the secret from John Dashwalds. Marianne and John go to London, and find that Willoughby

is a very good-natured, kind, and intelligent young man, and will be a good husband. They become friends, but John is disappointed when he learns that Marianne has been seeing Edward. He does not like Edward, and wants to see Marianne again. Marianne is happy when she learns that Willoughby is in town, and plans to tell El in return. Elinor is happy about this news, and tells John that she will not tell John that Willoyby is marrying Lucy. John does not believe her, and so he does not tell Elinor.

A.6.3 Generated Summary using SLED

The novel begins with the Dashwood family of Norland, England. The Dashwood sisters, Elinor, and Marianne Dashwood, are engaged to a man named Edward Ferrars, a wealthy and wealthy man who lives in the town of Barton Park, near Norland. They are engaged for a short period of time, but they are unable to afford to live there because of their mother's ill health. They decide to move to Barton Park in order to be closer to their family, and they decide to stay there for a few years. They move to Norland to live with their mother-in-law, Mrs. Dashwood and her three daughters, and the Dashwoods decide to live at Barton Park for a while. They begin to fall in love with their new neighbor, Mr. John Dashwood. However, they soon realize that they are not going to be able to afford their new home, and that they will have to move away from Norland for the sake of their children. They also realize that their new house will be too small for them to afford, and so they are forced to move out of their old home. Meanwhile, they decide that they should stay at Norland and move to a new house in Barton Park. They plan to spend their time there, as they have no money to spend on their own. They will move into the new house when they are old enough to afford it, and then they will move to the new home at Barton, where they will live with the family of their older brother, Edward, and his sister, Marianne, who will be living with them for the rest of their lives. In the meantime, they plan to move into a new home in the village of Barton, which will be built by their new neighbors. They hope to find a place to live for their children, but their plans are thwarted when they find out that they cannot afford to move. In order to make this possible, they have to leave Norland with their two daughters, whom they have never met before. They have to decide whether they want to live in Barton or not, and decide whether or not they should move there. When they do decide to do so, they are surprised to learn that they have been given a house in Norland by a wealthy man named Sir John Middleton, a friend of their father's, who is also a wealthy, well-liked man. They make plans to move there, but Sir John decides

to stay at Barton for the time being, and he decides to move with his sister and her two daughters. He decides that he will stay with his family in Barton, but he does not want to spend any more time with them than he can afford to spend with them. He also decides to marry a woman of his own, a woman whom he has never seen before, and who he will marry in the future. He is very happy to be with her, and is very fond of her, but his feelings for her are not reciprocated by her. When he is married, he is devastated by the loss of his wife and children, and feels that he cannot live with her. He wants to marry her, so he asks her to marry him, but she refuses. He tells her that she cannot marry him because she is too young, and she cannot afford it. He then decides that she will marry him. He leaves Norland at the end of the year, and when he returns to England, he plans to spend his time at Barton with his sisters. He plans to marry Marianne and their two young daughters, but Marianne refuses to marry them, because she feels that she is not worthy of the money he has given her. She decides to go with him to Barton, and decides to spend her time there with her sister. She and her sister decide to spend some time with him there, and after a few months, she decides to leave the house and move with him. She is very pleased to see that he is happy with the way he is living, and wishes to spend time with her and her daughters. When she returns home, she finds that she has not been able to spend much time with his children, so she is forced to leave them. She then decides to return to her old home, where she has been living with her mother and her sisters, who are all in the same house. She has no money and she is very unhappy with her situation. She does not know what to do with the money she has left, but her mother is very kind and generous. She tells him that she wants to stay with him and that she would be happy with him if he would marry her. After a short time, she and her mother decide to go to Barton to spend a few days with him, where he will spend the summer with his younger sister. They spend a lot of time together, and it is not long before they are reunited. They return to their old house at Barton.

A.6.4 Generated Summary using Unlimiformer

Elinor Dashwood, the eldest daughter of the Dashwood family, is engaged to Edward Ferrars. Elinor and Marianne, who have been engaged to Willoughby for the past four years, are devastated by the news of Edward's engagement to Marianne. The Dashwood sisters, Mrs. and Mrs. Jennings, are also devastated by their sister's news of her sister's engagement with Edward. Marianne and Edward, who are both in love with

their younger sister, are sent to Barton Street, where they are invited to meet Colonel Brandon and his wife, Lady Middleton, and the Dashwoods are to visit their sister, Lady Hurst, at their estate at Barton Park. The next morning, Elinore and Elinaor are invited by Colonel Brandon, who is in town to visit his sister, Marianne's sister, and her niece, Elinnor. The Colonel Brandon is very interested in Marianne as well as her sister, but he is not interested in her sister. The following morning, the Colonel Brandon arrives at Barton Street and asks to see Marianne at her house, where he is staying with his sister and his sister's friend, Lady Crawwood, and Mr. Dashwood. When Marianne arrives, she is shocked to see her sister and her sister in the same room. She tells her sister that Marianne has been engaged for three years, and that she has not seen her sister since her engagement to Edward. When she sees Marianne in the garden, she realizes that her sister has been in bed with her brother, and she has no way of knowing what she is going to do with her sister or her niece. She is devastated to find that her brother and sister are in love, but she is not sure what to do about Marianne or her sister; she is also devastated to learn that her younger sister is engaged, to a young man named Colonel Brandon. She does not know what Marianne is doing, but her sister does not want Marianne to know what she has been doing, so she is forced to tell Marianne about her sister-in-law's affair with her, and to try to make Marianne feel better by telling Marianne what she knows. She tries to convince Marianne that she is wrong, but Marianne does not believe her sister is right to know, and so she does not tell her. She decides that she will go to her uncle's house to visit Marianne when Marianne returns to Barton, where she will be staying with her mother and sister. She also decides to ask Marianne if she is interested in seeing her sister at Barton, for she is sure that she would be able to get her sister to marry her brother. She agrees to go to Barton and visit her sister when she returns from Barton, but when she comes back to Barton the next day, she finds that she cannot go with Marianne because she is afraid of Marianne returning to Barton. She knows that she must tell her sister if she ever sees her brother or Marianne again. She goes to her mother's house and asks Marianne for help. She asks Marianna to tell her about the fact that she was engaged to her sister before she left Barton. When her sister comes to her, she tells Marianne the truth about her brother's affair, she feels very guilty about her own feelings for Marianne; and she fears that she may be in danger. She sends Marianne away to Barton to visit her mother, who has recently returned from her wedding to Edward, and then to her sisters and her aunt, Lady Claywood. She and her sisters decide that they must leave Barton and go to London to find a man to marry, and they decide to stay at Barton to see if Marianne

can get Marianne out of her feelings for Edward. They decide to leave Barton for a few days, and Ellinor and Elineor decide to return to Barton in order to find Marianne before Marianne leaves. They plan to meet Marianne on her way to Barton when she is ill, but they do not know who will come to see them. They do not have any way of getting Marianne back, so they leave Barton. They go to a nearby cottage, where Marianne visits her sister for a couple of days before she leaves for Barton, and when she sees her sister again, she thinks she is in trouble. She thinks Marianne might be in trouble, and is afraid that she might get her brother to marry Marianne–or Marianne will be in a situation where she is unable to be with her. When they are in Barton, they go to the house of Mr. Hurwood and his brother, Mr. Crawwood. They visit the family at Barton and are surprised to see their sister in a room with a woman who looks like Marianne but is in a state of disinor. They also see a man named Mr. Crawford, whom they have never seen before, and who seems to be in the middle of their situation.

Bibliography

Agarwal, D., Fabbri, A. R., Han, S., Kryscinski, W., Ladhak, F., Li, B., McKeown, K., Radev, D., Zhang, T., and Wiseman, S. (2022). CREATIVESUMM: Shared task on automatic summarization for creative writing. In *Proceedings of The Workshop on Automatic Summarization for Creative Writing*, pages 67–73, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Ainslie, J., Ontanon, S., Alberti, C., Cvicek, V., Fisher, Z., Pham, P., Ravula, A., Sanghai, S., Wang, Q., and Yang, L. (2020). ETC: Encoding long and structured inputs in transformers. 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 268–284, Online. Association for Computational Linguistics.

Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate.

Barthes, R. and Duisit, L. (1975). An introduction to the structural analysis of narrative. *New Literary History*, 6:237.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *arXiv:2004.05150*.

Bertsch, A., Alon, U., Neubig, G., and Gormley, M. (2023a). Unlimiformer: Long-range transformers with unlimited length input. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 35522–35543. Curran Associates, Inc.

Bertsch, A., Alon, U., Neubig, G., and Gormley, M. R. (2023b). Unlimiformer: Long-range transformers with unlimited length input. *arXiv preprint arXiv:2305.01625*.

Brandow, R., Mitze, K., and Rau, L. F. (1995). Automatic condensation of electronic publications by sentence selection. *Inf. Process. Manage.*, 31(5):675–685.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020a). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020b). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Bruner, J. (1991). The narrative construction of reality. *Critical inquiry*, 18(1):1–21.

Carbonell, J. and Goldstein, J. (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, page 335–336, New York, NY, USA. Association for Computing Machinery.

Chen, M., Chu, Z., Wiseman, S., and Gimpel, K. (2022). SummScreen: A dataset for abstractive screenplay summarization. In *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, T., Xu, B., Zhang, C., and Guestrin, C. (2016). Training deep nets with sublinear memory cost.

Chen, Y.-C. and Bansal, M. (2018). Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the As-*

sociation for Computational Linguistics (Volume 1: Long Papers), pages 675–686, Melbourne, Australia. Association for Computational Linguistics.

Cheng, J. and Lapata, M. (2016). Neural summarization by extracting sentences and words. In *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.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. (2021). Rethinking attention with performers. In *International Conference on Learning Representations*.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al. (2022). Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Cohan, A., Dernoncourt, F., Kim, D. S., Bui, T., Kim, S., Chang, W., and Goharian, N. (2018). A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Correia, G. M., Niculae, V., and Martins, A. F. T. (2019). Adaptively sparse transformers. 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 2174–2184, Hong Kong, China. Association for Computational Linguistics.

Deutsch, D., Bedrax-Weiss, T., and Roth, D. (2021). Towards question-answering as an automatic metric for evaluating the content quality of a summary. *Transactions of the Association for Computational Linguistics*, 9:774–789.

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.

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H.-W. (2019). *Unified language model pre-training for natural language understanding and generation*. Curran Associates Inc., Red Hook, NY, USA.

Dou, Z.-Y., Liu, P., Hayashi, H., Jiang, Z., and Neubig, G. (2021). GSum: A general framework for guided neural abstractive summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4830–4842, Online. Association for Computational Linguistics.

Edmundson, H. P. (1969). New methods in automatic extracting. *J. ACM*, 16(2):264–285.

Erkan, G. and Radev, D. R. (2004). Lexrank: graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479.

Ernst, O., Shapira, O., Pasunuru, R., Lepioshkin, M., Goldberger, J., Bansal, M., and Dagan, I. (2021). Summary-source proposition-level alignment: Task, datasets and supervised baseline. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 310–322. Association for Computational Linguistics.

Fabbri, A., Wu, C.-S., Liu, W., and Xiong, C. (2022). QAFactEval: Improved QA-based factual consistency evaluation for summarization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2587–2601, Seattle, United States. Association for Computational Linguistics.

Fabbri, A. R., Kryściński, W., McCann, B., Xiong, C., Socher, R., and Radev, D. (2021). SummEval: Re-evaluating Summarization Evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Fisher, W. R. (1989). *Human Communication as Narration: Toward a Philosophy of Reason, Value, and Action*. University of South Carolina Press.

Ghodratnama, S. and Zakershahrak, M. (2024). SumRecom: A personalized summarization approach by learning from users' feedback.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

Gorinski, P. J. and Lapata, M. (2015). Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1066–1076, Denver, Colorado. Association for Computational Linguistics.

Gorinski, P. J. and Lapata, M. (2018). What's this movie about? a joint neural network architecture for movie content analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1770–1781, New Orleans, Louisiana. Association for Computational Linguistics.

Graesser, A. C., Singer, M., and Trabasso, T. (1994). Constructing inferences during narrative text comprehension. *Psychological review*, 101 3:371–95.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., and et al., A. K. (2024). The Llama 3 herd of models.

Gu, A., Goel, K., and Ré, C. (2022). Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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.

Han, C., Wang, Q., Peng, H., Xiong, W., Chen, Y., Ji, H., and Wang, S. (2024). LM-infinite: Zero-shot extreme length generalization for large language models. In Duh, K., Gomez, H., and Bethard, S., editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies (Volume 1: Long Papers), pages 3991–4008, Mexico City, Mexico. Association for Computational Linguistics.

He, B., Wang, J., Qiu, J., Bui, T., Shrivastava, A., and Wang, Z. (2023). Align and attend: Multimodal summarization with dual contrastive losses. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 14867–14878. IEEE.

Hendrycks, D. and Gimpel, K. (2023). Gaussian error linear units (gelus).

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Huang, L., Cao, S., Parulian, N., Ji, H., and Wang, L. (2021). Efficient attentions for long document summarization. In *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.

Ivgi, M., Shaham, U., and Berant, J. (2023a). Efficient long-text understanding with short-text models. *Transactions of the Association for Computational Linguistics*, 11:284–299.

Ivgi, M., Shaham, U., and Berant, J. (2023b). Efficient Long-Text Understanding with Short-Text Models. *Transactions of the Association for Computational Linguistics*, 11:284–299.

Izacard, G. and Grave, E. (2021). Leveraging passage retrieval with generative models for open domain question answering. In Merlo, P., Tiedemann, J., and Tsarfaty, R., editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Jiang, C., Xie, R., Ye, W., Sun, J., and Zhang, S. (2023). Exploiting pseudo image captions for multimodal summarization. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 161–175, Toronto, Canada. Association for Computational Linguistics.

Jiang, H., LI, Y., Zhang, C., Wu, Q., Luo, X., Ahn, S., Han, Z., Abdi, A. H., Li, D., Lin, C.-Y., Yang, Y., and Qiu, L. (2024). MInference 1.0: Accelerating pre-filling

for long-context LLMs via dynamic sparse attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Jones, K. S. (1998). Automatic summarising: factors and directions.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kintsch, W. and van Dijk, T. A. (1978). Toward a model of text comprehension and production. *Psychological Review*, 85:363–394.

Kitaev, N., Kaiser, L., and Levskaya, A. (2020). Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. (2018). The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Kryscinski, W., Rajani, N., Agarwal, D., Xiong, C., and Radev, D. (2022a). BOOKSUM: A collection of datasets for long-form narrative summarization. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6536–6558, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kryscinski, W., Rajani, N., Agarwal, D., Xiong, C., and Radev, D. (2022b). BOOKSUM: A collection of datasets for long-form narrative summarization. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6536–6558, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kupiec, J., Pedersen, J., and Chen, F. (1995). A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, page 68–73, New York, NY, USA. Association for Computing Machinery.

Ladhak, F., Li, B., Al-Onaizan, Y., and McKeown, K. (2020). Exploring content selection in summarization of novel chapters. In *Proceedings of the 58th Annual Meeting*

of the Association for Computational Linguistics, pages 5043–5054, Online. Association for Computational Linguistics.

Landauer, T. K., Foltz, P. W., and and, D. L. (1998). An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259–284.

Lehnert, W. G. (1981). Plot units and narrative summarization. *Cognitive Science*, 5(4):293–331.

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.

Li, Y., Huang, Y., Yang, B., Venkitesh, B., Locatelli, A., Ye, H., Cai, T., Lewis, P., and Chen, D. (2024). SnapKV: LLM knows what you are looking for before generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Lin, C.-Y. (1999). Training a selection function for extraction. In *Proceedings of the Eighth International Conference on Information and Knowledge Management*, CIKM '99, page 55–62, New York, NY, USA. Association for Computing Machinery.

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. *arXiv preprint arXiv:2307.03172*.

Liu, S., Cao, J., Yang, R., and Wen, Z. (2022a). Long text and multi-table summarization: Dataset and method. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1995–2010, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Liu, Y. and Lapata, M. (2019). Text summarization with pretrained encoders. In *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.

Liu, Y., Ni, A., Nan, L., Deb, B., Zhu, C., Awadallah, A. H., and Radev, D. (2022b). Leveraging locality in abstractive text summarization. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6081–6093, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

Mahon, L. and Lapata, M. (2024). A modular approach for multimodal summarization of TV shows. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8272–8291, Bangkok, Thailand. Association for Computational Linguistics.

Manakul, P. and Gales, M. (2021). Long-span summarization via local attention and content selection. In *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.

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

Mikolov, T., Karafiát, M., Burget, L., Černocký, J. H., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*.

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.

Mirza, P., Abouhamra, M., and Weikum, G. (2021). AligNarr: Aligning narratives on movies. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 427–433, Online. Association for Computational Linguistics.

Moskvichev, A. and Mai, K.-V. (2023). NarrativeXL: a large-scale dataset for long-term memory models. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15058–15072, Singapore. Association for Computational Linguistics.

Munkhdalai, T., Faruqui, M., and Gopal, S. (2024). Leave no context behind: Efficient infinite context transformers with infini-attention.

Nawrot, P., Łańcucki, A., Chochowski, M., Tarjan, D., and Ponti, E. M. (2024). Dynamic memory compression: retrofitting llms for accelerated inference. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

OpenAI, :, Hurst, A., Lerer, A., Goucher, A. P., and et al., A. P. (2024). GPT-4o system card.

Oren, M., Hassid, M., Yarden, N., Adi, Y., and Schwartz, R. (2024). Transformers are multi-state RNNs. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N., editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18724–18741, Miami, Florida, USA. Association for Computational Linguistics.

Osborne, M. (2002). Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization*, pages 1–8, Phildadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Papalampidi, P., Keller, F., Frermann, L., and Lapata, M. (2020). Screenplay summarization using latent narrative structure. In *Proceedings of the 58th Annual Meeting*

of the Association for Computational Linguistics, pages 1920–1933, Online. Association for Computational Linguistics.

Papalampidi, P., Keller, F., and Lapata, M. (2019). Movie plot analysis via turning point identification. In *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 1707–1717, Hong Kong, China. Association for Computational Linguistics.

Papalampidi, P., Keller, F., and Lapata, M. (2021). Movie summarization via sparse graph construction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13631–13639.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. (2024). YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

Phang, J., Zhao, Y., and Liu, P. (2023). Investigating efficiently extending transformers for long input summarization. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3946–3961, Singapore. Association for Computational Linguistics.

Phang, J., Zhao, Y., and Liu, P. J. (2022). Investigating efficiently extending transformers for long input summarization. *arXiv preprint arXiv:2208.04347*.

Pu, D., Hong, X., Lin, P.-J., Chang, E., and Demberg, V. (2022). Two-stage movie script summarization: An efficient method for low-resource long document summarization. In *Proceedings of The Workshop on Automatic Summarization for Creative Writing*, pages 57–66, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Qin, G., Rosset, C., Chau, E., Rao, N., and Van Durme, B. (2024). Dodo: Dynamic contextual compression for decoder-only LMs. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9961–9975, Bangkok, Thailand. Association for Computational Linguistics.

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. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Ratner, N., Levine, Y., Belinkov, Y., Ram, O., Magar, I., Abend, O., Karpas, E., Shashua, A., Leyton-Brown, K., and Shoham, Y. (2023). Parallel context windows for large language models. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6383–6402, Toronto, Canada. Association for Computational Linguistics.

Roy, A., Saffar, M., Vaswani, A., and Grangier, D. (2021). Efficient Content-Based Sparse Attention with Routing Transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. In Màrquez, L., Callison-Burch, C., and Su, J., editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Saxena, R., Bhat, S., and Pedanekar, N. (2017). Live on TV, alive on Twitter: Quantifying continuous partial attention of viewers during live television telecasts. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 1042–1049.

Saxena, R. and Keller, F. (2024a). MovieSum: An abstractive summarization dataset for movie screenplays. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4043–4050, Bangkok, Thailand. Association for Computational Linguistics.

Saxena, R. and Keller, F. (2024b). Select and summarize: Scene saliency for movie script summarization. In Duh, K., Gomez, H., and Bethard, S., editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3439–3455, Mexico City, Mexico. Association for Computational Linguistics.

Schluter, N. (2017). The limits of automatic summarisation according to ROUGE. In Lapata, M., Blunsom, P., and Koller, A., editors, *Proceedings of the 15th Conference*

of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 41–45, Valencia, Spain. Association for Computational Linguistics.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In Barzilay, R. and Kan, M.-Y., editors, *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.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. (2024). RoFormer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C).

Tay, Y., Dehghani, M., Tran, V. Q., Garcia, X., Wei, J., Wang, X., Chung, H. W., Bahri, D., Schuster, T., Zheng, S., Zhou, D., Houlsby, N., and Metzler, D. (2023). UL2: Unifying language learning paradigms. In *The Eleventh International Conference on Learning Representations*.

Todorov, T. (1971). The 2 principles of narrative. *Diacritics*, 1:37.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vansh, R., Rank, D., Dasgupta, S., and Chakraborty, T. (2023). Accuracy is not enough: Evaluating personalization in summarizers. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2582–2595, Singapore. Association for Computational Linguistics.

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.

Wang, F., Song, K., Zhang, H., Jin, L., Cho, S., Yao, W., Wang, X., Chen, M., and Yu, D. (2022). Salience allocation as guidance for abstractive summarization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020). Linformer: Self-attention with linear complexity.

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2022). Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Wu, Y., Rabe, M. N., Hutchins, D., and Szegedy, C. (2022). Memorizing transformers. In *International Conference on Learning Representations*.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. (2024). Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Xie, J., Cheng, P., Liang, X., Dai, Y., and Du, N. (2024). Chunk, align, select: A simple long-sequence processing method for transformers.

Yen, H., Gao, T., and Chen, D. (2024). Long-context language modeling with parallel context encoding.

You, Y., Jia, W., Liu, T., and Yang, W. (2019). Improving abstractive document summarization with salient information modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2132–2141, Florence, Italy. Association for Computational Linguistics.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. (2020a). Big Bird: Transformers for longer sequences. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. (2020b). Big bird: Transformers for longer sequences. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.

Zha, Y., Yang, Y., Li, R., and Hu, Z. (2023). AlignScore: Evaluating factual consistency with a unified alignment function. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11328–11348, Toronto, Canada. Association for Computational Linguistics.

Zhang, J., Zhao, Y., Saleh, M., and Liu, P. (2020a). PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Zhang, P., Shao, N., Liu, Z., Xiao, S., Qian, H., Ye, Q., and Dou, Z. (2024). Extending Llama-3's context ten-fold overnight.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019). BertScore: Evaluating text generation with BERT. *arXiv preprint arXiv:1904.09675*.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2020b). BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*.

Zhang, Y., Ni, A., Mao, Z., Wu, C. H., Zhu, C., Deb, B., Awadallah, A., Radev, D., and Zhang, R. (2022). Summ^n: A multi-stage summarization framework for long input dialogues and documents. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1592–1604, Dublin, Ireland. Association for Computational Linguistics.

Zhao, C., Brahman, F., Song, K., Yao, W., Yu, D., and Chaturvedi, S. (2022). NarraSum: A large-scale dataset for abstractive narrative summarization. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 182–197, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zheng, H. and Lapata, M. (2019). Sentence centrality revisited for unsupervised summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6236–6247, Florence, Italy. Association for Computational Linguistics.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Zhong, M., Liu, Y., Xu, Y., Zhu, C., and Zeng, M. (2022). Dialoglm: Pre-trained model for long dialogue understanding and summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11765–11773.

Zhong, M., Yin, D., Yu, T., Zaidi, A., Mutuma, M., Jha, R., Awadallah, A. H., Celikyilmaz, A., Liu, Y., Qiu, X., and Radev, D. (2021). QMSum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921, Online. Association for Computational Linguistics.

Zhou, Q., Yang, N., Wei, F., Huang, S., Zhou, M., and Zhao, T. (2018). Neural document summarization by jointly learning to score and select sentences. In Gurevych, I. and Miyao, Y., editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663, Melbourne, Australia. Association for Computational Linguistics.

Zhu, C., Yang, Z., Gmyr, R., Zeng, M., and Huang, X. (2021). Leveraging lead bias for zero-shot abstractive news summarization. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 1462–1471, New York, NY, USA. Association for Computing Machinery.

Zwaan, R. A. and Radvansky, G. A. (1998). Situation models in language comprehension and memory. *Psychological bulletin*, 123 2:162–85.