NEURAL ABSTRACTIVE SUMMARIZATION FOR LONG DOCUMENTS

SHUAIQI LIU

PhD

The Hong Kong Polytechnic University

2024

The Hong Kong Polytechnic University
Department of Computing

Neural Abstractive Summarization for Long Documents

Shuaiqi LIU

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
August 2023

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: Shuaiqi LIU

Abstract

Long documents, like academic literature, financial reports, and legal instruments, are important information sources. Nowadays, people can access massive long documents through the Internet. Reading through all their acquired documents and finding their desired content would be a heavy burden. The high-quality summaries can help people quickly grasp the key information from original documents. Automatic text summarization techniques can be employed to produce concise summaries for long documents. Abstractive summarization methods can approximate how humans write summaries by capturing input documents' salient content and generating novel sentences as summaries.

In this thesis, I study neural abstractive summarization for long documents. I aim to train neural network models to generate informative, fluent, and non-redundant summaries covering the multi-granularity, multi-document, and multimodal salient content in various long documents. Some new challenges arise in order to accomplish this objective: 1) the scarcity of available datasets, 2) identifying the multi-granularity salient information scattered in long inputs, 3) incorporating multi-document and multimodal content when generating summaries, 4) evaluating the quality of the generated summaries, 5) improving the efficiency of model training and inference. To tackle the above challenges, I built multiple large-scale datasets, novel summarization methods, and evaluation metrics, which are summarized below.

First, I built multiple large-scale long document summarization datasets for academic

literature, financial reports, and legal instruments, which can be the foundation of long document summarization research. Meanwhile, my datasets support extending long document summarization research from unimodal to multimodal, from summarizing a limited number of documents to a large number of documents.

Second, I propose a series of techniques to identify the multi-granularity salient information scattered in long documents. This thesis introduces novel attention mechanisms, category-based content alignment method, and the multistage content selection schema for identifying and encoding phrase-level, sentence-level, and segment-level salient content.

Besides, my research validates the importance of jointly considering multimodal or multi-document content when summarizing long documents. This thesis proposes multiple methods incorporating salient content from text and tables into summary generation. Besides, this thesis also proposes methods to summarize multiple categories of salient content from a large number of documents and generate structured summaries.

To evaluate various summarization methods, my research not only employs commonly used automatic evaluation metrics but also proposes novel evaluation metrics. We also compare different models' generated summaries by human evaluation.

Last but not least, my research leverages various techniques to improve the efficiency of model training and inference. This thesis not only proposes efficient summarization models but also adopts some memory-efficient training methods. These techniques enable training large neural summarization models over long inputs on an off-the-shelf GPU.

I hope this thesis can promote the long document summarization research. Although this thesis presents novel datasets, methods, and evaluation metrics for this topic, it still has many open problems. I list some future research directions at the end of this thesis.

Publications Arising from the Thesis

1. Shuaiqi Liu, Jiannong Cao, Ruosong Yang, and Zhiyuan Wen, "Generating a Structured Summary of Numerous Academic Papers: Dataset and Method", In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI 2022)*, pages 4259–4265, 2022.

2. Shuaiqi Liu, Jiannong Cao, Ruosong Yang, and Zhiyuan Wen, "Long Text and Multi-Table Summarization: Dataset and Method", In *Findings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022)*, pp. 1995-2010, 2022.

3. Shuaiqi Liu, Jiannong Cao, Zhongfen Deng, Wenting Zhao, Ruosong Yang, and Zhiyuan Wen, "Neural Abstractive Summarization for Long Text and Multiple Tables", *IEEE Transactions on Knowledge and Data Engineering (TKDE).*

4. Shuaiqi Liu, Jiannong Cao, Ruosong Yang, and Zhiyuan Wen, "Key phrase aware transformer for abstractive summarization", *Information Processing & Management (IP&M)* 59, no. 3 (2022): 102913.

5. Shuaiqi Liu, Jiannong Cao, Ruosong Yang, and Zhiyuan Wen, "Highlight-Transformer: Leveraging Key Phrase Aware Attention to Improve Abstractive Multi-Document Summarization", In *Findings of the 59th Annual Meeting of the Association for*

Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021), pp. 5021-5027. 2021.

6. Shuaiqi Liu, Jiannong Cao, Yicong Li, Ruosong Yang, and Zhiyuan Wen, "Leveraging Foundation Models to Improve Low-Resource Court Judgment Summarization for Common Law Systems", manuscript submitted to *Information Processing & Management (IP&M)*.

7. Chenxi Hu, Tao Wu, Shuaiqi Liu, Chunsheng Liu, Chao Chang, and Fang Yang, "Joint Unsupervised Contrastive Learning and Robust GMM for Text Clustering", *Information Processing & Management (IP&M)*.

8. Zhiyuan Wen, Jiannong Cao, Yu Yang, Haoli Wang, Ruosong Yang, Shuaiqi Liu, "DesPrompt: Personality-descriptive Prompt Tuning for Few-shot Personality Recognition", *Information Processing & Management (IP&M)*.

9. Zhiyuan Wen, Jiannong Cao, Ruosong Yang, Shuaiqi Liu, and Jiaxing Shen. "Automatically Select Emotion for Response via Personality-affected Emotion Transition." In *Findings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)*, pp. 5010-5020. 2021.

10. Zhiyuan Wen, Jiannong Cao, Ruosong Yang, Shuaiqi Liu, Jiaxing Shen, Maosong Sun, "Personality-affected Emotion Generation in Dialog Systems", manuscript submitted to *ACM Transactions on Information Systems (TOIS)*.

11. Zhiyuan Wen, Jiannong Cao, Yu Yang, Ruosong Yang, and Shuaiqi Liu. "Affective-NLI: Towards Accurate and Interpretable Personality Recognition in Conversation", manuscript accepted by *The 22nd International Conference on Pervasive Computing and Communications (PerCom 2024)*.

Acknowledgments

Before I started my Ph.D., I set ambitious goals and plans and naively hoped that everything would move forward as planned if I tried hard enough. I used to seek certainty and was uncomfortable with occasional uncertainty. Growing up in a booming social environment brings natural optimism that everything will keep growing and life will get better. However, I gradually realized that uncertainty is the norm and certainty is accidental after some tumultuous months. I learned to reconcile with myself and find inner peace in the ups and downs of life. As I look back on my Ph.D. journey, the most unforgettable moments would be those spent with great teachers and friends.

Firstly, I want to thank my family for their unlimited support during my growth. I am so lucky to have perfect parents and grandparents who always patiently communicate with me, respect my decisions, and selflessly support me to achieve my dream. No matter when and where I can feel their infinite love as long as I call them. My father's motto: 'There is no flaming mountain you cannot get past' helped me stay optimistic during some tough days. My hard-working grandparents and mother are my role models, encouraging me to overcome various difficulties and keep moving forward. As I grew up, I understood the difficulties of being a parent. Compared with their experiences, many difficulties in front of me are so trivial.

Next, I would like to give my biggest thanks to my Ph.D. supervisor Prof. Jiannong Cao. I really admire that he is always energetic and treats everything with his heart.

He is my role model who taught me to constantly explore the endless frontiers of science, to be enthusiastic and sensitive to new technologies, and to keep learning new knowledge. I would not have had the chance to write this thesis without his patient guidance and enthusiastic help. I am also very grateful to my supervisor, Prof. Philip S. Yu, during my visiting period at University of Illinois Chicago. He is a famous scholar and excellent supervisor who always patiently listens to students' ideas, accurately grasps the key points, and gives his wise suggestions.

My colleagues in the IMCL group offered me constant support and cooperation throughout these years. Dr. Ruosong Yang and Dr. Zhiyuan Wen helped me a lot when my research career started. They taught me many basic skills for doing research and their understanding of technology. We are the IMCL-NLP group and have done a lot of work together. Dr. Yannni Yang has helped me a lot since I applied for the scholarship. Without her help, it would be very difficult for us to purchase the necessary equipment for our research. Dr. Yannni Yang and Dr. Linchuan Xu helped me a lot when I applied for the research student attachment program and started my life in the United States. I also want to thank Dr. Yu Yang, Dr. Jia Wang, Dr. Zhuo Li, Dr. Wanyu Lin, and Dr. Jinlin Chen, who provide many useful suggestions for my research career. Dr. Qianyi Chen, Dr. Mingjin Zhang, Dr. Zhixuan Liang, and I joined the IMCL group in 2019. We helped each other and made progress together. I emphasize my appreciation to Dr. Yicong Li. She helped me a lot in research and life and provided crucial emotional support, which lit up my life.

I would like to thank my friends in the BDSC Lab. Thank Mr. Liangwei Yang, Mr. Chen Wang, Mr. Xiaolong Liu, Mr. Jiangshu Du, Dr. Ziwei Fan, Dr. Yingtong Dou, Dr. Xiaohan Li, Ms. Zhongfen Deng, Ms. Wenting Zhao, Ms. Yueqing Liang, Mr. Kay Liu, Mr. Hengrui Zhang, Mr. Mingdai Yang, Ms. Yuwei Cao, Ms. Tao Zhang, Ms. Yu Wang, Ms. Yibo Wang, Ms. Yuqing Liu, Mr. Weizhi Zhang, and Mr. Yuanjie Zhu for their warm welcome and great support during my visiting period at University of Illinois Chicago.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Long documents, like academic literature, financial reports, government reports, and legal instruments, are important information sources. Nowadays, people can access massive long documents through the Internet. Reading through all their acquired documents and finding their desired content would be a heavy burden [72, 74]. During the COVID-19 pandemic, hundreds of thousands of academic papers about COVID-19 were released on the Internet[1]. Researchers were drowned in the torrent of coronavirus papers, while a large amount of them are not what people care about [73]. People need advanced tools to support efficiently reading and selecting long documents[2].

Providing readers with high-quality summaries of long documents can help alleviate the above problems. High-quality summaries can help people efficiently obtain the key information in original documents [73]. Meanwhile, people can first read the summary to determine if one document is worth further reading, which enables people

[1]https://www.nature.com/articles/d41586-020-03564-y

[2]https://www.sciencemag.org/news/2020/05/scientists-are-drowning-covid-19-papers-can-new-tools-keep-them-afloat

to quickly filter out undesired documents and save time and effort. Employing human experts to write summaries costs a lot of time and effort, making it challenging to cover new or unpopular documents from diverse sources [73]. Automatic document summarization techniques can be utilized to produce summaries [75]. Users can adjust the input documents flexibly and immediately get summaries from the automatic summarization system.

Automatic document summarization techniques aim to produce a concise summary of one or more documents [72, 74]. Previous document summarization methods can be broadly categorized into extractive and abstractive methods [73, 75]. Extractive summarization methods [7, 36, 37, 78, 84, 85, 112] identify and select the most salient sentences from the input document to form the summary, while abstractive summarization methods [19, 41, 77, 86, 97, 109] can approximate the way humans write summaries by capturing and merging input documents' salient information and then generating a summary that may contain new expressions [74]. Besides, previous summarization methods can be classified into neural-network-based and non-neural methods according to whether the neural network is used.

This thesis mainly focuses on the neural abstractive summarization methods. They can be further divided into recurrent neural network (RNN) based methods [86, 116], transformer-based methods (trained from scratch) [38, 41, 72, 74], and pre-trained foundation model-based methods [73, 75, 78, 149], according to their backbone models. RNN model [18, 53] was widely used in short document summarization [86, 116], but it has limited ability to learn the long-range dependencies [131]. The transformer model relying on the attention mechanisms can model the long-range dependencies [131] and shows better performance on long document summarization [72, 74].

The performance of supervised models trained from scratch is limited by the size of the labeled dataset. Labeling large-scale datasets can cost a lot, while unlabeled data can be easily crawled from the Internet. Researchers pre-train large foundation models with self-supervised tasks on a huge amount of unlabeled data to learn better

text representations [13, 32, 102]. After being fine-tuned on labeled datasets for downstream tasks, pre-trained models can outperform models trained from scratch. This thesis covers both our novel model trained from scratch and our summarization methods built on pre-trained foundation models.

In addition to the development of neural summarization methods, publicly available summarization datasets also facilitated improving generated summaries' quality. In recent years, researchers released various large-scale datasets [23, 38, 52, 70, 81, 88, 118], which made it possible to train large neural models for document summarization [74]. Summarization tasks or datasets can be categorized into single-document summarization (SDS) and multi-document summarization (MDS) based on the number of documents taken into consideration for a summary. Considering different types of content in input documents, previous work can be divided into unimodal and multimodal summarization. My research built datasets for both SDS and MDS and generalized long document summarization from unimodal (text) to multimodal.

Automatic text summarization techniques have been widely applied in many domains, like news article summarization, forum post summarization, and product review summarization. This thesis studies summarizing various long documents from different domains, including academic literature, financial reports, and legal instruments.

1.2 Motivation

Long documents, like academic literature, financial reports, government reports, and legal instruments, have some properties:

- *Plentiful but scattered salient content:* Compared with short documents, long documents often contain more salient content, which can be scattered in multiple parts of long documents.
- *Containing multiple categories of content:* Long documents can describe an

object from multiple aspects. For example, public companies' financial reports usually describe multiple aspects of the company, including business, operations, and cash flow. Academic papers usually describe the background, problem definition, methods, and experimental results of research work.

- *Containing multimodal content:* The long documents may contain multimodal content (e.g., text, tables, and figures). Some key information may only appear in non-textual content (e.g., tables and figures), like accounting data in a company's financial statements [75, 76].

- *Following some structures or templates:* Compared with the free-form short text, long documents usually follow some structures to organize plentiful content. There are some given structures (templates), like the division of sections or chapters. For example, the U.S. Securities and Exchange Commission stipulates companies' annual reports' format [75, 76]. Academic journals and conferences also require manuscripts following their templates. Meanwhile, some structures are author-defined. For instance, authors may arrange sentences introducing different content in the same paragraph.

Previous summarization research usually focuses on summarizing short documents, like short news [38, 44], forum posts [59], and online product reviews [82, 153]. When generalizing to long document summarization, previous document summarization datasets and methods did not consider the above properties of long documents and still have limitations.

Annotated datasets are the foundation of long document summarization research. Previous document summarization datasets usually focus on short documents [38, 44, 59]. The lack of large-scale datasets limits the long document summarization research. Specifically, previous long document summarization datasets only focus on text content. This limits research on generating summaries for multimodal content in long documents [75, 76]. Besides, previous multi-document summarization datasets

focus on summarizing a limited number of input documents, which limits the research on generating summaries for a large number of documents [73].

When generalizing from short documents to long documents, previous summarization methods have difficulties in completely identifying and encoding multi-granularity salient content (e.g., key phrases, sentences, and paragraphs) scattered in a large amount of input content [75, 76]. It can limit the informativeness of their generated summaries.

As for the multimodal content (e.g., text, tables, and figures) in long documents, existing summarization datasets and methods usually concentrate on textual content while disregarding non-textual content [75, 76]. However, some key information may only appear in non-textual content (e.g., tables and figures). Missing non-textual content can limit produced summaries' informativeness [75, 76].

Neural summarization models usually require more time or GPU memory when processing longer input and output [131]. For long document summarization, improving the efficiency of model training and inference is important [73, 75, 76]. The sequential nature of RNN hampers parallelization [18, 53]. Compared with RNN, the transformer model reduces the required sequential operations and is more parallelizable. However, the complexity of the transformer's self-attention mechanism scales quadratically with the input length [131], which limits transformer-based models' efficiency. Given the constraints of GPU memory size, the complexity of the transformer-based model limits the context length it can model. In addition to the complexity of the neural summarization model, some training techniques (e.g., gradient accumulation, gradient checkpointing[3], optimizer [30, 104], parameter quantization [29, 139], and parameter-efficient adapters [31, 55]) can also affect the training efficiency.

[3]github.com/cybertronai/gradient-checkpointing

1.3 Research Challenges

This thesis is confronted with the following challenges in summarizing long documents with neural abstractive summarization methods.

- The scarcity of available datasets: The lack of large-scale datasets limits the long document summarization research. The labeled dataset is necessary for training and evaluating summarization methods. To deal with this challenge, we built multiple large-scale datasets for academic literature summarization, financial report summarization, and legal instrument summarization, which can be the foundation of long document summarization research. Meanwhile, our datasets are the basis of extending long document summarization research from unimodal to multimodal, from summarizing a limited number of documents to a large number of documents.

- Identifying the salient information scattered in long inputs: Long documents often contain rich multi-granularity salient content. The salient content can be scattered in different parts of the documents. It is challenging to completely identify and encode multi-granularity salient content scattered in a large amount of input content. To tackle this challenge, we propose novel attention mechanisms, category-based content alignment, and a multistage summarization schema to identify and encode phrase-level, sentence-level, and segment-level salient content.

- Incorporating multi-document or multimodal content when generating summaries: The long documents may contain multimodal content (e.g., text, tables, and figures). How to effectively integrate multi-document and multimodal salient content into the generated summaries is an important issue. My research work validates the importance of jointly considering multimodal content in long documents and proposes multiple methods incorporating text and tables into

summary generation. We also study summarizing a large number of academic documents and generate structured summaries.

- Evaluating generated summaries' quality: How to effectively assess the quality of generated summaries from different aspects is also an important issue. My research not only employs commonly used automatic evaluation metrics but also proposes novel evaluation metrics. We also compare different models' generated summaries by human evaluation.

- Improving the efficiency of model training and inference: The efficiency of model training and inference is important when deploying summarization models in real-world scenarios. When processing very long inputs and outputs with large neural models, it is challenging to save computation time and GPU memory. My research work takes different approaches to improve summarization methods' efficiency and enable the training of large neural summarization models over long inputs on an off-the-shelf GPU. To improve the model efficiency, we propose a lightweight transformer-based model named KPAT and employ sparse attention mechanisms in multiple summarization models. Besides, we adopt some memory-efficient training methods, like gradient accumulation, gradient checkpointing, parameter quantization, memory-efficient optimizer, and adding parameter-efficient adapters. In addition, we also adopt the divide-and-conquer-based training strategy to further reduce the context length that neural summarization models need to model and the corresponding memory consumption.

1.4 Research Framework

This thesis studies neural abstractive long document summarization. As shown in Fig. 1.1, my research framework has five layers. The lower four layers correspond to the four main parts of my research, including data preparation, long document content

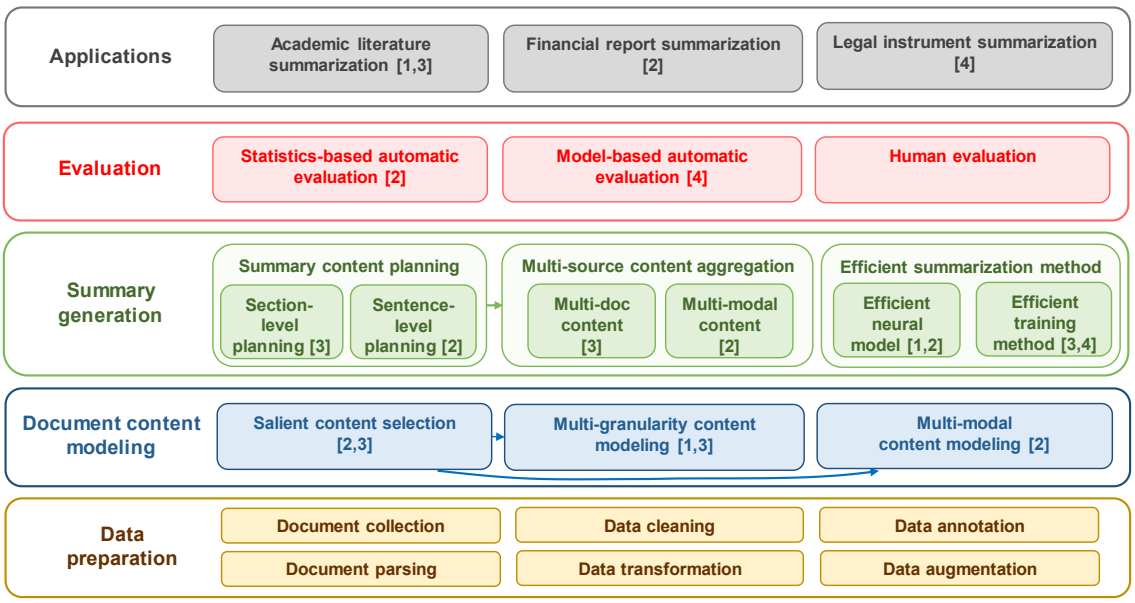


Figure 1.1: Research framework.

modeling, summary generation, and evaluation. On top of them is the application layer they support.

The bottom layer is data preparation, which is usually the first step in my research work. To build a summarization dataset, we first need to find data sources, collect raw files, and parse these raw files to extract document content. We should clean the extracted content and align input documents with their target summaries. Sometimes we need to convert the format of data. When the labeled samples are insufficient, we will consider data augmentation. Chapter 2 introduces existing summarization datasets. Chapter 4 presents my multimodal financial report summarization dataset. My academic literature summarization dataset is shown in Chapter 5. My court judgment summarization dataset is presented in Chapter 6. This thesis introduces the process of building each dataset in the corresponding chapter.

Above the data preparation layer is the long document content modeling layer. Its key problem is to identify and encode multi-granularity and multimodal salient content scattered in a large amount of input content [74–76]. Chapter 4 and Chapter 6 intro-

duce my content selection methods for text and tabular data. These content selection methods conduct a rough selection, which compresses long inputs while maximizing the recall of salient content that should be preserved in summaries [75]. The compressed inputs are fed into summarization models for fine-grained content selection and integration. After selecting the salient content, how to completely encode the multi-granularity and multimodal content is also important. Chapter 3 proposes novel attention mechanisms encoding token-level and phrase-level salient content. Chapter 5 encodes document-level salient content. Chapter 4 studies encoding salient content in text and tables.

The middle layer is the summary generation layer. We need to plan the content of the output summary and aggregate salient content from multiple sources. When planning the summary content, Chapter 5 generates structured summaries and plans the division of summary sections. Chapter 4 first generates sentence-level templates and then fills content into these templates. As for multiple sources input content aggregation, Chapter 4 studies aggregate multimodal content into summary generation. Chapter 5 aggregates the salient content from tens of input documents for each example. Improving the efficiency of model training and inference is also very important. My research work proposes efficient summarization models and employs efficient training methods [72–76].

In the evaluation layer, we usually conduct automatic evaluation and human evaluation to assess various summarization models' generated summaries. The metrics we used in automatic evaluation can be further classified into statistics-based evaluation metrics and model-based evaluation metrics. We not only employ commonly used evaluation metrics (e.g., ROUGE [67], BLEU [95], and BARTScore [145]) but also propose novel evaluation metrics [75, 76].

Lastly, my research proposes multiple long document summarization datasets and methods that can support different applications. Chapter 4 introduces my work on multimodal financial report summarization [75, 76]. Chapter 5 presents my academic

literature summarization work. My court judgment summarization work is introduced in Chapter 6.

1.5 Thesis Organization

The rest of this thesis is organized as follows:

- Chapter 2 reviews existing work in neural abstractive summarization and long document summarization. This chapter first briefly introduces the taxonomies of existing summarization work. Then, this chapter illustrates the development of neural abstractive summarization methods, including the RNN-based, transformer-based (trained from scratch), and pre-trained foundation model-based methods. This chapter also introduces existing summarization datasets.
- Chapter 3 introduces my first work that proposes the key phrase aware transformer (KPAT), a lightweight model achieving great performance on multiple abstractive summarization tasks. This work focuses on enhancing the transformer encoder to completely encode the key phrases in input documents. We present the highlighting mechanism incorporating the prior knowledge of key phrases when calculating attention weights for tokens within key phrases.
- In Chapter 4, we propose a new task named long text and multi-table summarization, which generalizes the long document summarization from unimodal (text) summarization to multimodal. Previous document summarization datasets and methods are usually restricted to summarizing the text content and excluding tables and figures from the input. In financial report documents, the key information can be distributed across both textual and non-textual content. The absence of tabular data can restrict the informativeness of generated summaries, particularly when summaries necessitate the quantitative descriptions of vital metrics within tables. Existing summarization methods and datasets

fail to meet the demands of summarizing extensive textual and tabular content within financial reports. To deal with the scarcity of available datasets, we propose FINDSum, the first large-scale dataset for long text and multi-table summarization. Besides, we present four types of summarization methods to jointly consider the text and table content when summarizing reports. Additionally, we propose a set of evaluation metrics assessing the utilization of numerical information within the generated summaries.

- In Chapter 5, we study how to summarize numerous academic papers about the same topic into a structured summary. Existing multi-document summarization (MDS) work usually focuses on producing an unstructured summary that encompasses only a limited number of input documents. Meanwhile, previous structured summarization work focuses on summarizing each document into a summary with multiple sections. Existing methods and datasets fail to fulfill the demands of summarizing numerous academic literature. We propose BigSurvey, the first large-scale dataset for generating comprehensive summaries of numerous academic papers on each topic. Besides, we propose the category-based alignment and sparse transformer (CAST) to effectively arrange the diverse content from a large number of input documents while simultaneously ensuring efficiency when processing long inputs.

- Chapter 6 illustrates my work on low-resource court judgment summarization. Judges in common law systems need to find similar precedents in all common law jurisdictions and refer to the reasoning in previous judgments. There exist hundreds of thousands of reported cases in common law jurisdictions, and the number of cases is still increasing. It can be challenging for legal practitioners to read through abundant cases' judgment documents. We aim to let the computer generate high-quality court judgment summaries, which can help readers quickly browse key information in long judgment documents. To deal with the scarcity of available datasets, we propose CLSum, the first large-scale

dataset for summarizing common law court judgment documents from multiple jurisdictions. Similar to other domain-specific tasks, court judgment summarization usually suffers from the shortage of labeled samples. To deal with this problem, we propose a foundation model-based solution for the low-resource court judgment summarization. To the best of our knowledge, we are the first to employ large language models for data augmentation, summary generation, and evaluation in court judgment summarization. Additionally, we propose an evaluation metric named LTScore to assess the quality of the generated legal text.

- In Chapter 7, I summarize this thesis's research problems and contributions. Meanwhile, I also present many open problems and future directions in the long document summarization.

Chapter 2

Literature Review

2.1 Neural Networks

This section will briefly introduce neural network models commonly used in abstractive summarization, including recurrent neural networks, transformer model, and transformer-based pre-trained models.

2.1.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of neural network for processing sequential data. To calculate the hidden state at each step, RNNs combine the current step's input with the hidden state from the previous step. The hidden state maintains the memory of the previous inputs [144]. RNNs share the same weights for each step of the sequence and can model varying-length sequences. Traditional RNNs suffer from the problem of vanishing gradients and exploding gradients. The frequently used RNN variants, like the Gated Recurrent Unit (GRU) [18] and Long Short-Term Memory (LSTM) [53] network, selectively update the hidden state. RNNs have been widely used in many applications, including machine translation, speech recognition,

Table 2.1: Characteristics of different neural network layers. The sequence length is n, the representation dimension is d, and the kernel size of convolutions is k [136].

Type of Layer	**Complexity**	**Sequential Operations**	**Maximum Path Length**
Self-attention Layer	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
CNN Layer	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
RNN Layer	$O(n \cdot d^2)$	$O(n)$	$O(n)$

and document summarization [144]. However, the sequential nature of RNN hampers parallelization. Besides, the forward and backward signals have to propagate through the long paths in the network [136], which limits the ability to learn the long-range dependencies.

2.1.2 Transformer Model

Transformer [131] is an advanced sequence transduction model that dispenses the recurrence and convolution structures and is solely built on attention mechanisms. The transformer model relying entirely on the attention mechanisms can model the long-range dependencies since the self-attention reduces the maximum path length to $O(1)$ as depicted in Table 2.1 [136]. Compared with RNN models, the transformer model also reduces the required sequential operations from $O(n)$ to $O(1)$ and is more parallelizable [131, 136].

The effectiveness of the transformer model was first verified on machine translation tasks [131]. Then it has been applied in various natural language processing tasks, including language modeling, question answering, named entity recognition, and document summarization. There have been many transformer-based models for both extractive and abstractive summarization, bringing notable performance gains. This

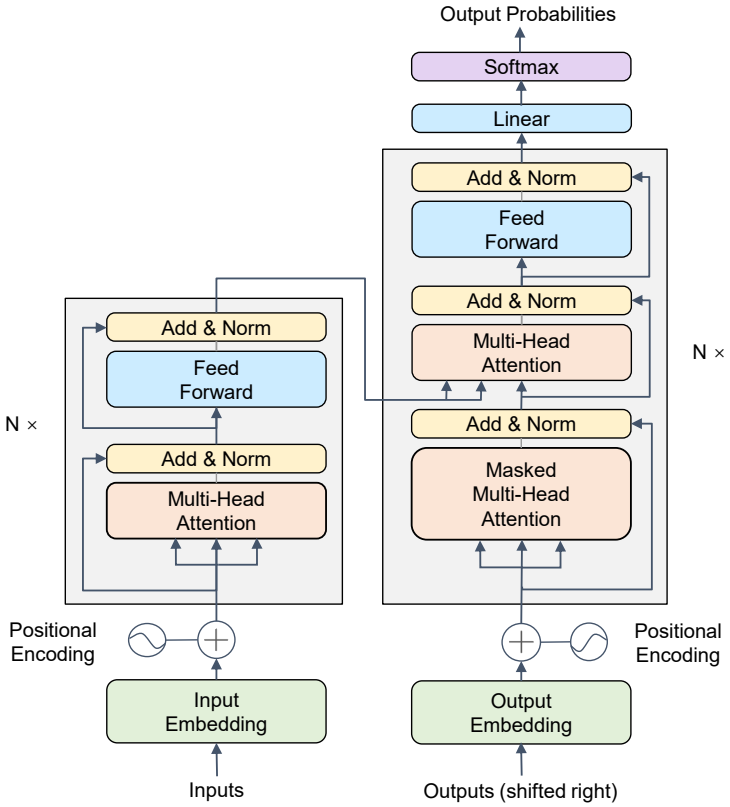


Figure 2.1: The architecture of transformer model. [131]

sub-section briefly introduces the transformer model's architecture and its attention mechanisms.

Architecture of Transformer Model

The transformer model [131] follows the encoder-decoder structure. The encoder maps input sequences to continuous representations. Given the representations of inputs, the decoder is responsible for generating the output sequences [72, 74, 131].

As shown in Fig. 2.1, the encoder part of the transformer model is comprised of N layers, each containing two sub-layers. The first sub-layer employs the multi-head self-attention mechanism. The second is a position-wise fully connected feed-forward

network [72, 74, 131]. The outputs from the stacked sub-layers are connected with the residual connection [51] and normalized with layer normalization [3].

$$\text{LayerNorm}(x + \text{Sublayer}(x)) \tag{2.1}$$

Similarly, the decoder part of the transformer model also consists of N identical layers. The multi-head self-attention sub-layers mask subsequent positions in attention weight matrices. Compared with the encoder layers, the decoder layers add an additional sub-layer, which conducts the encoder-decoder attention, enabling the interaction between the encoder's output and the output of the decoder's multi-head self-attention sub-layer [72, 74, 131].

Attention mechanisms in transformer

The transformer model [131] replaces the recurrent neural network layers with multi-headed self-attention, which reduces the sequential computation and becomes more parallelizable. The multi-head attention mechanism employs the scaled dot-product attention for each head. Its operation involves a query Q, a key K, and a value V [72, 74, 131].

$$\text{Attention}(Q, K, V) = W^m V \tag{2.2a}$$

$$W^m = \text{softmax}(\frac{QK^T}{\sqrt{d_k}}) \tag{2.2b}$$

The weight matrix $W^m \in \mathbb{R}^{n \times n}$ is calculated by Eq. (2.2b). d_k is the dimensionality of key K.

The multi-head attention utilizes the scaled dot-product attention across h heads [131].

$$\begin{aligned} &\text{MultiHead}(Q, K, V) = \text{Heads}W^o \\ &\text{Heads} = \text{Concat}(\text{Head}_1, ..., \text{Head}_h) \\ &\text{Head}_\text{i} = \text{Attention}(Q, K, V) \end{aligned} \tag{2.3}$$

The matrix Head_i is calculated by Eq. (2.2a). In Eq. (2.3), the outputs from all the heads will be concatenated and subsequently projected through a feed-forward layer with a parameter matrix $W^o \in \mathbb{R}^{hd_v \times d_{model}}$ [72, 74, 131].

There are three different types of multi-head attention layers in the transformer model [131]. Fig. 2.2 shows two types of self-attention layers in the transformer's encoder and decoder.

- The self-attention layers in the transformer encoder receive the outputs of previous encoder layers as the inputs and calculate the keys, values, and queries based on the inputs. Without the mask, each position can attend to all input positions [131].
- The self-attention layers within the transformer decoder are similar to that of the encoder, except that they adopt masking to avoid each position attending to the positions preceding it to uphold the auto-regressive property [131].
- Within the encoder-decoder attention layers, the encoder outputs keys and values. The queries come from the previous decoder layer. This configuration allows every position in the decoder to attend to all positions in the encoder [131].

2.1.3 Pre-trained Models Based on Transformer

In many language understanding and language generation tasks, a general bottleneck lies in the availability of large-scale labeled datasets. Existing labeled datasets are relatively small and not enough for training very large deep neural networks. It can cause overfitting and cannot generalize well in practice. In recent years, various pre-trained models built on the transformer model have been flourishing. They are also known as foundation models. The idea of pre-training and fine-tuning is to first

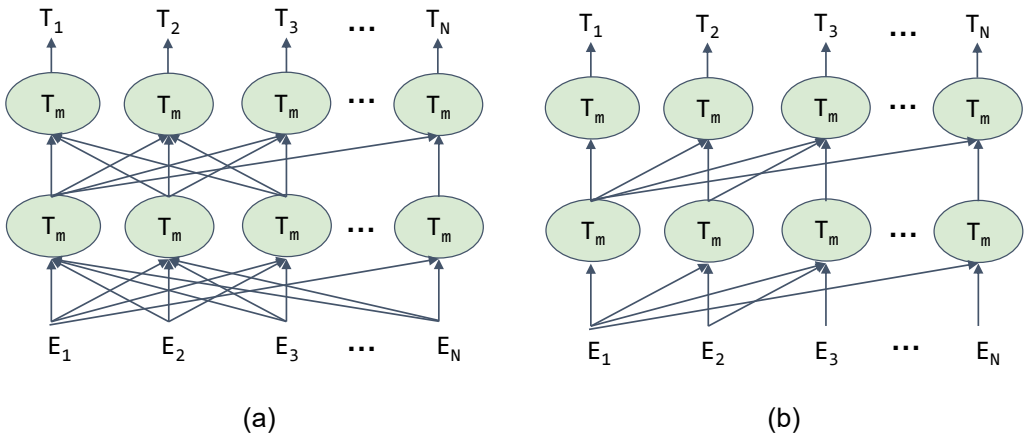


Figure 2.2: Two types of transformer self-attention layers: (a) the bidirectional self-attention layer and (b) the unidirectional self-attention layer

pre-train the large neural models on extensive unlabeled data with self-supervised learning tasks and then fine-tune these models on supervised downstream tasks [32, 101]. The transformer-based pre-trained models can be categorized according to their architectures. Some pre-trained models are built on the encoder or decoder part of the transformer model (e.g., BERT [32], RoBERTa [79], and GPT [101]), while some of them are built on the entire transformer model, like MASS [122], BART [64], and PEGASUS [149]. This sub-section briefly reviews some representative transformer-based pre-trained models.

Pre-trained Models Based on Transformer Decoder

Based on the unidirectional self-attention layers in the transformer decoder, the GPT model [101] adopts a standard left-to-right language modeling objective and is autoregressively pre-trained on the BooksCorpus dataset. The pre-trained GPT model [101] can be fine-tuned on various supervised tasks, including natural language inference, question answering, and text classification, which reveals it can generalize well to both natural language understanding and generation. The subsequent versions

of the GPT model, including GPT-2 [102] and GPT-3 [13], are large-size models with billions and hundreds of billions of parameters. They demonstrate that generative pre-training can benefit various downstream tasks, and the capacity of the language model is essential to their success on zero-shot or few-shot learning. They have been used in summarization [102]. Some previous work also tried using the pre-trained GPT model to replace the randomly initialized transformer decoder in the transformer-based summarization models.

GPT-3 model is closed-source, while many open-source large pre-trained models follow GPT's decoder-only architecture. OPT [151] is a series of large pre-trained models ranging from 125M to 175B parameters. The performance of OPT-175B is similar to GPT-3. BLOOM [113] is a multilingual language model with 176B parameters. It supports 46 languages and 13 programming languages. LLaMA [129] is a collection of foundation language models ranging from 7B to 65B parameters trained on large-scale publicly available data (1-1.4 trillion tokens). LLaMA2 [130] models are trained on more data (2 trillion tokens) and longer context length (4096 tokens).

Although these pre-trained models have good zero-shot performance, using them for downstream tasks requires further adaptation to improve performance. Researchers fine-tune these large pre-trained models with instruction-following examples to align them with user intentions. Stanford Alpaca [128] is a LLaMA model fine-tuned with 52K instruction-following demonstrations generated by OpenAI's text-davinci-003 API. Peng et al. [98] fine-tune the LLaMA model with 52K instruction-following data generated by GPT-4. Vicuna [17] is a set of LLaMA models fine-tuned with 70K user-shared ChatGPT conversations from ShareGPT.com.

To avoid generating untruthful or toxic outputs or reflecting harmful sentiments, Ouyang et al. [94] propose reinforcement learning from human feedback (RLHF). RLHF mainly includes three steps: 1) collecting human-written examples of prompts and using these examples to train a language model with supervised learning; 2) training a reward model (RM) on a dataset of human-labeled comparisons between two

model outputs to predict labelers' preferred output choice; 3) fine-tuning the GPT-3 policy with the PPO algorithm to maximize the reward. RLHF enables InstructGPT to generate more appropriate outputs with less toxicity and hallucinates [94]. Stiennon et al. [124] shows that training a model to optimize for human preferences can significantly improve the quality of generated summaries.

Pre-trained Models Based on Transformer Encoder

Considering the bidirectional contextual information is critical for various downstream natural language understanding tasks, some famous pre-trained models are built on the multi-layer transformer encoder like BERT [32] and RoBERTa [79]. The self-attention layers in the transformer encoder are pre-trained to model the bidirectional contextual information via the bidirectional attention mechanism.

Among these pre-trained encoders, BERT [32] is the most representative work. It adopts two self-supervised pre-training tasks: the masked language model (MLM) and the next sentence prediction (NSP). It is pre-trained on large corpora named BooksCorpus and Wikipedia. The MLM task encourages the transformer encoder to learn the bidirectional contextual information, which is essential for many downstream tasks [32].

RoBERTa [79] is one of the famous variants of BERT. The author found removing the NSP objective matches or slightly improves its performance on the downstream tasks. They also adopt dynamic masking and train the model with longer sequences and larger batch sizes. Besides, they also found that BERT was undertrained, and training the model over more data and for more steps brought the performance gain.

Some pre-trained transformer encoders have been fine-tuned on the extractive summarization task to [78]. Some models [78, 107] replace the randomly initialized transformer encoder with pre-trained encoders in transformer-based abstractive summarization models. The bidirectional contextual information learned by pre-trained

encoders can bring performance gains for both extractive and abstractive summarization.

Pre-trained Models Based on Encoder-Decoder Transformer Model

The encoder-decoder framework has been widely adopted in conditional text generation tasks, including machine translation, document summarization, and dialog response generation [4, 86]. The transformer model adopts the bidirectional self-attention for the encoder and the unidirectional self-attention for the decoder. It utilizes the cross attention to connect the two parts [131]. We will briefly introduce representative work on sequence-to-sequence pre-training.

MASS [122] adopts the encoder-decoder architecture and a masked sequence-to-sequence pre-training task that demands the model to predict the fragment of the sentence that is masked on the encoder side. To force the decoder to rely more on the encoded representation other than the previous tokens in the decoder, it only provides previous tokens in the masked fragment for the decoder when generating the fragment.

BART [64] is a denoising autoencoder that corrupts the input text with an arbitrary noising function and then trains a standard transformer model to reconstruct the original input text. They compared the different corruption methods and found that randomly shuffling sentences and using a novel in-filling scheme achieved the best performance on various text generation tasks. They also reveal the effectiveness of pre-training methods is highly dependent on the pre-training task.

Raffel et al. [103] take the text-to-text as a unified framework and generalize their pre-trained model T5 to various natural language processing tasks. Their experiments verify that scaling up the size of the pre-trained model and the pre-training corpus consistently improves the performance on a variety of downstream tasks.

In addition to these general pre-training objectives adopted in the above pre-trained models, the pre-training objectives tailored for the downstream task can bring extra performance gain. Zhang et al. [149] propose a new self-supervised objective named Gap Sentences Generation (GSG), which masks whole sentences from a document and aims to generate these gap sentences according to the rest of the document and is quite similar to the abstractive summarization task. They use the GSG task to pre-train a transformer model named PEGASUS and achieve the SOTA performance on various summarization datasets [149].

2.2 Automatic Document Summarization

Automatic document summarization is a classical research topic in natural language processing (NLP). It is a process of condensing the input and producing a summary that retains the salient information from the original input. In the past decades, there has been lots of work on this topic, and previous summarization methods can be categorized according to various criteria. Based on the mode of summary production, previous summarization methods can be broadly categorized into extractive and abstractive methods. Besides, previous summarization methods can be classified into neural and non-neural methods according to whether using the neural network. This section briefly reviews previous neural methods for extractive and abstractive summarization. Some hybrid methods combining extractive and abstractive summarization will also be introduced.

2.2.1 Neural Extractive Summarization Methods

Extractive summarization methods identify and select the most salient sentences from the input document to form the summary. Its objective is to maximize the coverage of salient content in input documents while minimizing repetitive content [74]. Ex-

tractive summarization methods [36, 37, 78, 84, 85, 112] have been studied for several decades. The neural-based methods garnered substantial attention in recent years. Typically, neural extractive summarization methods comprise two main steps: sentence representation acquisition and selecting the most salient sentences to compose the summary [85].

Various neural networks have been adopted to acquire accurate sentence representation, including the convolutional neural network (CNN) [91], the graph neural networks (GNN) [135], the recurrent neural networks (RNN) [87], and the transformer model [78, 140, 152]. When it comes to the step of sentence selection, previous research [16, 87, 143] usually treats it as either a sentence classification or sequence labeling problem. To mitigate the discrepancy between the training objective and the evaluation criterion, Narayan et al. [91] tried directly optimizing the ROUGE score with reinforcement learning.

Despite the notable development of extractive summarization methods over the last few decades, the extracted summaries still encounter challenges related to coherence and readability [137, 142]. Consequently, abstractive summarization methods have garnered escalating interest in recent years.

2.2.2 Neural Abstractive Summarization Methods

Abstractive summarization methods identify and select the salient content in documents and generate novel sentences as summaries. Compared with extractive methods, abstractive methods have the ability to approximate how humans write summaries by consolidating and condensing information from multiple sentences. Furthermore, abstractive summarization methods possess the capacity to generate new expressions that are not explicitly present within the input documents [46, 68, 150].

Abstractive summarization models usually adopt the encoder-decoder architecture [19, 86, 97, 109]. These models are confronted with two challenges: 1) Identifying and

encoding salient content of input document. 2) Generating high-quality summaries through the decoder [74, 75].

To identify and encode salient content, existing work employs various neural models as the encoder, including the CNN [34, 69, 108], the RNN [19, 86, 97, 116, 127], the GNN [66], and the transformer encoder [65, 77]. Besides, researchers try different attention mechanisms to encode the input context [4, 108, 116, 127].

To facilitate the summary generation, the pointer-generator network [116] employs the copy mechanism, which enables copying input words. The copy mechanism can alleviate the out-of-vocabulary problem. It is also helpful for generating factually correct summaries.

Extractive and abstractive summarization methods have their advantages and disadvantages. Researchers attempted to combine these two approaches' advantages and proposed some hybrid methods. The most straightforward way is adopting the extraction-then-generation framework [70, 80, 100, 125]. Other work [57, 78] explores the potential of integrating extractive and abstractive summarization through multi-task learning.

2.2.3 Document Summarization Datasets

Over the past few decades, the development of summarization models and the construction of large-scale summarization datasets have jointly advanced document summarization research. Large-scale labeled datasets enable the training of large neural summarization models via supervised learning.

In table 2.2, we summarize some frequently-used document summarization datasets from different domains, including news articles, forum posts, scientific literature, legal instruments, and government reports. We can discover that the average lengths of the input short news articles or forum posts in these datasets are shorter than 1,000

words. The average lengths of their corresponding summaries are usually shorter than 60 words. Examples in the Multi-News [38] dataset concatenate multiple news articles about the same event or topic as inputs. Typically, the input for each example is restricted to the text content found within the original documents, while tables and figures are usually excluded from the input.

Compared with short documents (e.g., news articles, forum posts, and product reviews), long documents, including scientific literature, government reports, and legal documents, usually contain thousands of words. The average lengths of these ground truth summaries (e.g., the abstracts in academic papers) in these datasets are usually shorter than 300 words. The government reports in the GOVREPORT [56] and the book chapters in the BookSum [62] are much longer. The average lengths of their ground truth summaries are more than 500 words.

Table 2.2: Statistical information of some public summarization datasets. "Pairs" denotes the number of examples. "Input Len" and "Target Len" denote the average number of words in input documents and ground truth summaries

Domain	**Dataset**	**Mode**	**Pairs**	**Input Len**	**Target Len**
News	XSum [90]	SDS	226.7K	431	23
	Newsroom [44]	SDS	1.21M	751	30
	NYTimes [111]	SDS	655K	549	40
	CNN/DM [52]	SDS	312K	790	56
	Multi-News[38]	MDS	56.2K	2,103	264
Forum	TIFU-short [59]	SDS	79.9K	342	9
	TIFU-long [59]	SDS	42.9K	432	23
	TLDR-Comment [133]	SDS	1.67M	225	22
	TLDR-Submission [133]	SDS	2.38M	416	34
Scientific	SCITLDR [14]	SDS	3.2K	5K	21
	BIGPATENT [118]	SDS	1.34M	3,573	117
	PubMed [23]	SDS	133K	3,016	203
	Arxiv [23]	SDS	215K	4,938	220
	CSPubSum [25]	SDS	10.3K	8.2K	226
	FacetSum [83]	SDS	58.3K	6,827	290
	Multi-XScience [81]	MDS	40.5K	778	116
	WikiSum [70]	MDS	1.57M	36,802	139
Law	BillSum-US [35]	SDS	23.5K	1,382	207
Report	GOVREPORT [56]	SDS	19.5K	9,409	553
Book	BookSum-Para [62]	SDS	142.7K	160	41
	BookSum-Chapter [62]	SDS	12.3K	5,102	505
	BookSum-Full [62]	SDS	436	112,885	1167

Chapter 3

Key Phrase Aware Transformer for Abstractive Document Summarization

3.1 Introduction

Nowadays, people can access massive text documents through the Internet. It brings a heavy burden for people to read through all their acquired documents and find their desired parts [36, 85, 150]. For example, hundreds of thousands of academic papers about the COVID-19 pandemic have been released on the Internet since 2020.[1] As a result, people were drowned in the torrent of varying-quality papers, making it harder to track frontier research. There is an urgent need to develop advanced tools to assist people in efficiently reading plentiful text documents.[2]

Automatic text summarization techniques, which produce a concise summary of one

[1] https://www.nature.com/articles/d41586-020-03564-y

[2] https://www.sciencemag.org/news/2020/05/scientists-are-drowning-covid-19-papers-can-new-tools-keep-them-afloat

or more text documents [36, 85, 112], can be utilized to alleviate the above problem. The produced summary can help people quickly grasp the key information from the original document and determine whether the document is worth further reading. Previous text summarization methods can be generally classified into two categories: extractive and abstractive summarization methods. Extractive methods [7, 36, 37, 78, 84, 85, 112] select important sentences from input documents to form summaries. While abstractive methods [19, 41, 77, 86, 97, 109, 141] capture and encode the salient content from input documents as the condition for generating novel sentences as summaries.

Many recent abstractive summarization methods [77, 78, 141] are built on the transformer model and achieve great performance in various summarization tasks. In the transformer encoder, calculating attention weights is a crucial step for encoding input documents. Encoding key phrases completely is important to encode the semantic information of input documents. However, existing transformer-based abstractive summarization models did not consider key phrases in input when determining self-attention weights, and their objective functions usually focus on sequence generation. Without the constraint of a specific objective function or the guidance of prior knowledge, it can be difficult for the model to encode these key phrases completely. When testing these models, we observed that some tokens within key phrases only receive small attention weights, which is not conducive to encoding these phrases and the salient information they convey.

Existing summarization datasets usually do not have labels of key phrases, so we cannot train the model through supervised multi-task learning to recognize keywords while generating summaries. In this chapter, we introduce some prior knowledge of key phrases into the transformer-based summarization model and guide the model to encode these key phrases.

Our work is inspired by previous studies in education and psychology. They reveal that key phrases are important for people to understand [47, 106] and summarize

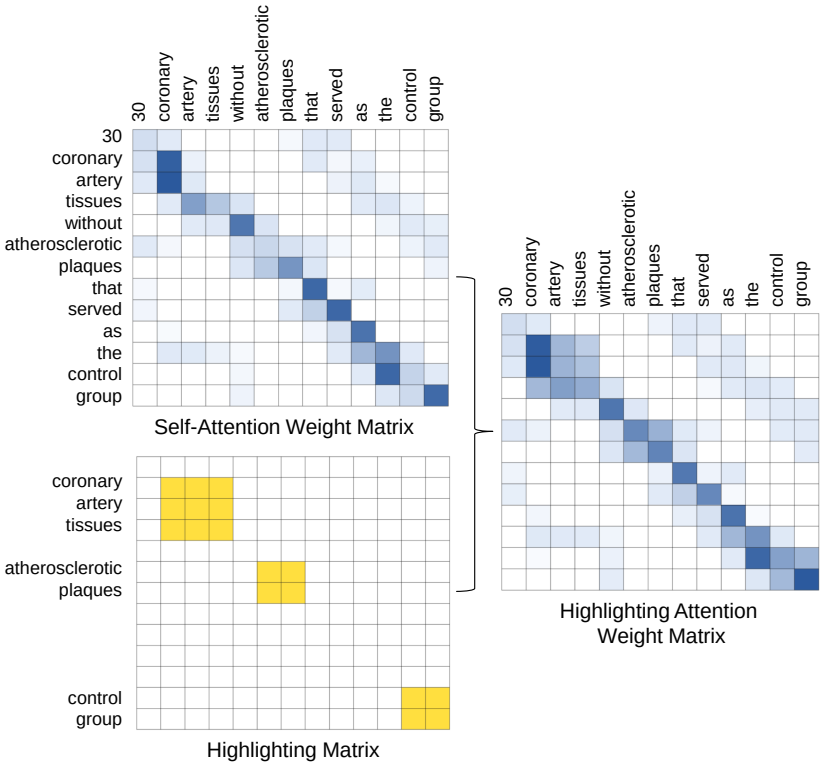


Figure 3.1: The highlighting mechanism assigns greater attention weights for tokens within key phrases indicated by the highlighting matrix.

[8, 21] the given documents. Besides, previous work found that highlighting key phrases can help people with dyslexia improve comprehension [47, 106]. One possible advantage of highlighting is that it utilizes a cognitive bias named the Von Restorff effect [96, 134]. The highlighted portion of text stands out from the surrounding non-highlighted text, making it more memorable [146]. These findings can be instructive to improve abstractive summarization models.

There are usually multiple tokens in each key phrase. These tokens should be highly related and serve as a grammatical unit together. For the contextual representation of key phrases, we assume that the tokens within the same key phrase make larger contributions than other tokens in the input sequence. Based on this assumption, we propose the Key Phrase Aware Transformer (KPAT), an abstractive summariza-

tion model with the highlighting mechanism in the encoder. As shown in Fig.3.1, the highlighting mechanism assigns greater attention weights for tokens within key phrases.

Our proposed highlighting mechanism comprises three main parts: the highlighting matrix, the highlighting attention for each head, and the multi-head highlighting attention. We build a highlighting matrix for each input token sequence to indicate key phrases' importance scores and their positions in the input sequence. To combine self-attention weights with key phrases' importance scores, we propose two structures of highlighting attention. Besides, we conduct the block-wise linear transformation on the highlighting matrix to adjust the contributions of phrases' importance scores.

We evaluate our model and various summarization baselines on a multi-document summarization (MDS) dataset named Multi-News [38] and a single document summarization (SDS) dataset named PubMed [23]. Automatic evaluation results show that our model significantly improves the ROUGE scores [67] of generated summaries. Human evaluation results also confirm that the highlighting mechanism can improve the informativeness of generated summaries. Experimental results on these two datasets verify that our proposed methods can generalize to different summarization tasks (MDS and SDS) and different domains (news articles and academic literature).

In addition, we conduct more experiments to analyze the impact of each part of our model on the summarization performance. Firstly, we compare adopting the highlighting attention in different numbers of heads and layers and discover that using it in a subset of heads or layers surpasses using it in all heads or layers. Secondly, we analyze the impact of the number of introduced key phrases and the utilized key phrase extractor. Experimental results show that introducing more accurate and adequate key phrases can bring extra performance gains for the KPAT model. The ablation studies also validate the effectiveness of each part of the highlighting mechanism.

The contribution of this work is threefold:

- We propose the highlighting mechanism assigning greater attention weights for tokens within key phrases to encode them completely.
- We design two structures of highlighting attention for each head and the multi-head highlighting attention to combine self-attention weights with key phrases' importance scores indicated by the highlighting matrix.
- Our proposed KPAT model significantly outperforms advanced summarization baselines on two datasets in different domains (news articles and academic papers) and different summarization tasks (MDS and SDS).

3.2 Objectives

The primary focus of this research is to enhance the transformer-based abstractive summarization model's ability to completely encode key phrases in input documents. We aim to introduce some prior knowledge to guide the model to encode key phrases. To achieve this goal, we need to complete three objectives:

- To introduce key phrases' importance and position information into the transformer-based abstractive summarization model.
- To modify the encoder part of the transformer model and combine self-attention weights with key phrases' importance scores.
- To evaluate our proposed model on different summarization datasets and verify its effectiveness.
- To analyze the impact of the introduced key phrases' precision and coverage on our model's summarization performance.

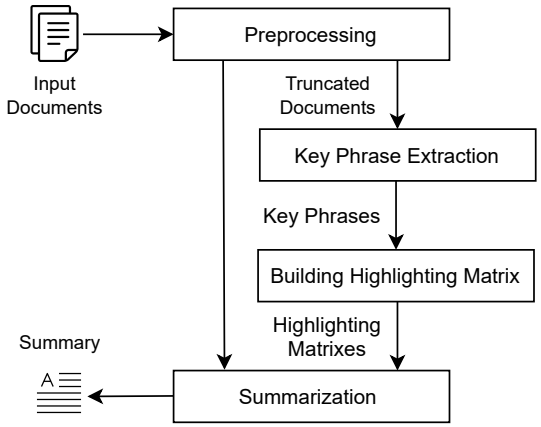


Figure 3.2: The workflow of our proposed method.

3.3 Proposed method

Our proposed method includes several steps, including data preprocessing, key phrase extraction, building highlighting matrix, and summarization, as depicted in Fig. 3.2. The procedures of data preprocessing and key phrase extraction are presented in subsection 3.3.1. Moreover, subsection 3.3.2 introduces our KPAT model, which mainly comprises the highlighting matrix, the highlighting attention mechanism for each head, and the multi-head highlighting attention mechanism.

3.3.1 Data preparation

We need to prepare the dataset for training and evaluating our proposed abstractive summarization model. Each input example of our KPAT model contains the truncated articles, key phrases, and their importance scores. We first preprocess input documents as introduced in subsection 3.3.1. In subsection 3.3.1, we adopt the automatic key phrase extraction method to assess phrases' importance and select the ones with top importance scores as key phrases.

Input documents preprocessing

Input documents need to be preprocessed to meet the requirements of neural summarization models. We only preserve the text content in input documents. To ensure the efficiency of neural summarization models, we need to truncate input documents. For the MDS dataset, we can truncate input documents within each example. In the SDS dataset, an input document usually contains multiple parts. We can consider their contributions to the summary and truncate these parts. More specific operations should depend on the nature of the dataset and the summarization models' requirements, and they will be discussed in subsection 3.5.1.

Phrase importance assessment

This chapter aims to enhance the transformer model's ability to completely encode key phrases that usually convey the salient information of input documents. As a prerequisite, the phrases' importance should be assessed, and key phrases should be identified. Since there are usually no labels of key phrases in existing summarization datasets, we utilize unsupervised key phrase extraction methods to score phrases' importance and select the phrases with top-N scores as key phrases. After removing stopwords, we tried a statistics-based method named tf-idf [110][3] and two graph-based ranking methods: TopicRank [11] and PositionRank [40].[4] We only select bigrams and trigrams since longer phrases are sparse and more likely to be compressed in summaries. Based on these extracted key phrases, we conduct the L2 normalization on their scores assigned by the extractor as their importance scores.

After the step of key phrase extraction, we build the highlighting matrix for each input example based on extracted key phrases and their importance scores. More

[3]We calculate the tf-idf score by the scikit-learn library https://scikit-learn.org/stable/index.html

[4]We adopt the implementations of TopicRank and PositionRank from https://github.com/boudinfl/pke

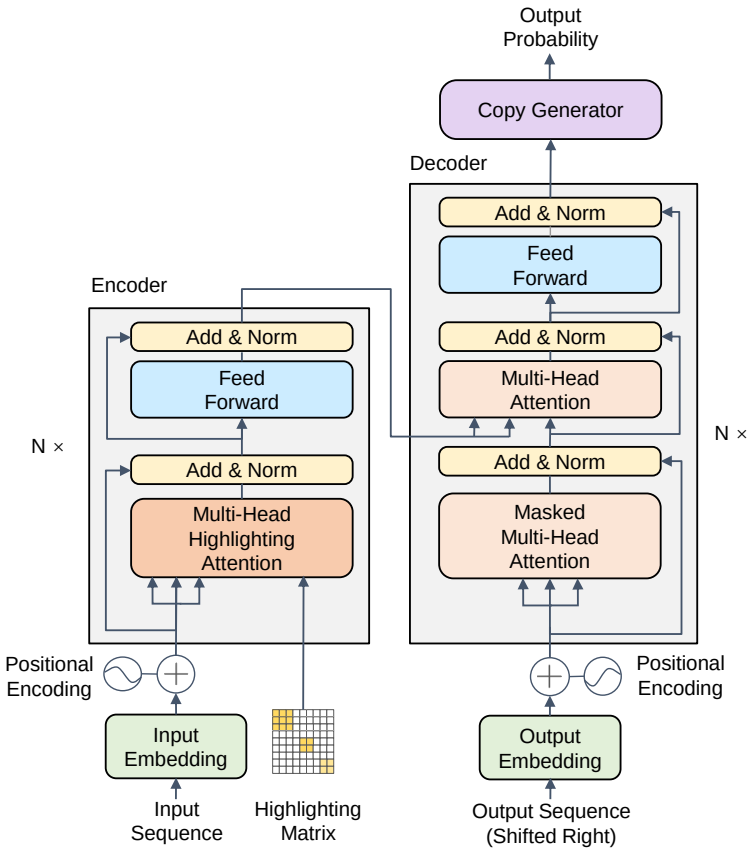


Figure 3.3: The architecture of the Key Phrase Aware Transformer (KPAT) model.

details of building the highlighting matrix will be illustrated in subsection 3.3.2. We also conduct experiments to compare the effects of adopting different key phrase extractors and selecting different numbers of key phrases. Our experimental results will be reported and analyzed in subsection 3.6.4.

3.3.2 Key phrase aware transformer model

This section introduces the Key Phrase Aware Transformer (KPAT), a model with the highlighting mechanism. We first present the architecture of the KPAT model. And then, three main components in the highlighting mechanism: the highlighting matrix, the highlighting attention for each head, and the multi-head highlighting attention

will be introduced separately.

Model architecture

Our KPAT model follows the encoder-decoder structure. This chapter mainly focuses on the encoder part since our motivation is to augment the transformer's ability to encode key phrases in input documents. The decoder part of the KPAT model follows settings in [38, 41]. Fig. 3.3 depicts the architecture of the KPAT model.

The encoder of our KPAT model consists of N identical layers. Each of them has two sub-layers: the multi-head highlighting attention layer and the position-wise fully connected feed-forward layer. These encoder layers' inputs include the previous layer's output and the highlighting matrix. We replace the multi-head self-attention layers in the original transformer model [131] with the multi-head highlighting attention layers, which will be presented in subsection 3.3.2. Each multi-head highlighting attention layer contains h heads and employs the highlighting attention on p highlighted heads to adjust attention weights according to the phrase importance. We depict the highlighting attention in subsection 3.3.2.

Highlighting matrix

The first step of the highlighting mechanism is to build a highlighting matrix for each input example based on the results of key phrase extraction. The highlighting matrix can indicate key phrases' positions in the attention weight matrix and these phrases' importance scores.

As described in subsection 3.3.1, the input of each example in the MDS dataset is the concatenation of multiple truncated articles. In the SDS dataset, the input of each example is the truncated single document. Each example's input can be represented as an input sequence $(t_1, ..., t_n)$ containing n tokens. We use $(p_1, ..., p_k)$ and $(s_1, ..., s_k)$

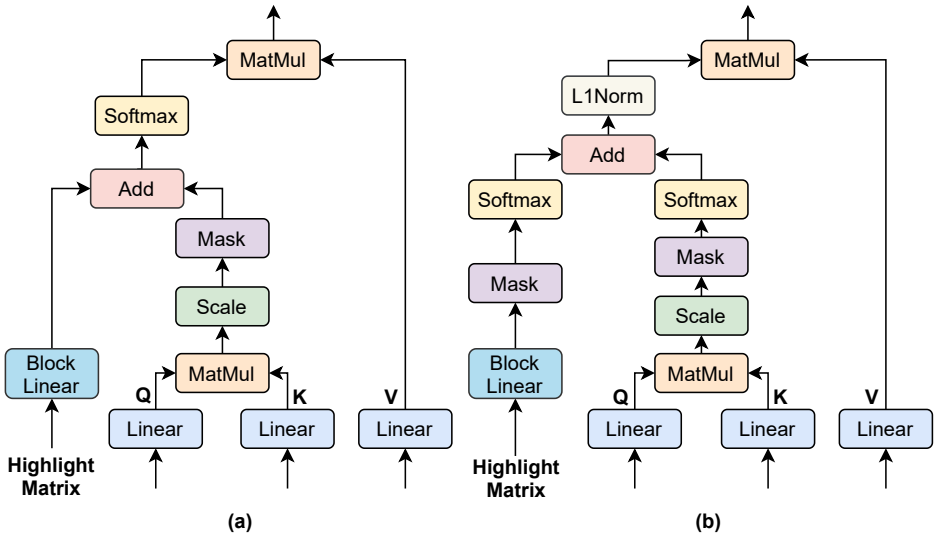


Figure 3.4: An overview of the highlighting attention mechanisms, namely (a) the weighted highlighting attention and (b) the additive highlighting attention.

to denote key phrases extracted from truncated articles and their importance scores.

For each input example, we build the highlighting matrix $H \in \mathbb{R}^{n \times n}$ with the same shape as the self-attention weight matrix. Assuming a phrase p_r contains b tokens in the input sequence $p_r = (t_a, ..., t_{a+b})$, and the phrase's importance score s_r is added to the elements $H_{i,j}$, where $i = a, \ldots, a+b, j = a, \ldots, a+b$, in the highlighting matrix. These phrases can be partially overlapping or nested, and the token t_i can be contained in c phrases $(p_r, ..., p_{r+c})$, whose importance scores are $(s_r, ..., s_{r+c})$. The element H_{ii} is assigned as the maximum value of the c phrases' importance scores. Finally, we get a block diagonal matrix as the highlighting matrix $H = \text{diag}(H_1, H_2, ..., H_t)$, in which the main-diagonal blocks are square matrices and all off-diagonal blocks are zero matrices, as depicted in Fig. 3.1.

Highlighting attention

The highlighting attention is the crucial component in our model for adjusting attention weights according to the phrase importance. For the head m, the original transformer model [131] adopts Eq. (2.2b) to calculate the scaled dot-product attention.

We propose two structures of highlighting attention, namely the weighted highlighting attention and the additive highlighting attention, to replace the scaled dot-product attention. These two structures of highlighting attention are compared in Fig. 3.4.

Within the inputs of the highlighting attention, all the keys, values, and queries come from the previous layer's output. The input highlighting matrix H can indicate which elements in the attention weight matrix should be increased.

$$H^m = \text{block-linear}(H) = \text{diag}(\text{linear}(H_1), ..., \text{linear}(H_t)) \tag{3.1}$$

The block-wise linear transformation conducts on the block diagonal highlighting matrix $H = \text{diag}(H_1, H_2, ..., H_t)$. As shown in Eq. (3.1), submatrices in the highlighting matrix are transformed to adjust the scale of key phrases' importance scores.

The weighted highlighting attention mainly modifies Eq. (2.2b) to calculate the attention weight matrix W^m for the head m. The block diagonal highlighting matrix $H = \text{diag}(H_1, H_2, ..., H_t)$ is transformed by Eq. (3.1). Then the result H^m is added to the input of the softmax function.

$$W^m = \text{softmax}(\frac{QK^T}{\sqrt{d_k}} + H^m) \tag{3.2}$$

As shown in Eq. (3.3), the softmax function applies the exponential function to each input element and divides them by the sum of all these exponentials. Since the additive operation in Eq. (3.2) is identical to calculating the weighted average, we

name it weighted highlighting attention.

$$\text{softmax}(z_i + b_i) = \frac{e^{b_i} e^{z_i}}{\sum_{j=1}^{n} e^{b_j} e^{z_j}} \quad i = 1, \ldots, n \tag{3.3}$$

The additive highlighting attention is also designed for adjusting the attention weight matrix W^m. The block-wise linear transformed result H^m is normalized by the softmax function[5] and added to the original attention weight matrix W_a^m calculated in Eq. (2.2b). After that, the matrix W_b^m produced by Eq. (3.4a) is normalized along the dimension, where the softmax function is computed, to ensure the sum of elements in that dimension is equal to one.

$$W_b^m = W_a^m + \text{softmax}(H^m) \tag{3.4a}$$

$$W_{:,\,j}^m = \frac{W_{b\,:,\,j}^m}{||W_{b\,:,\,j}^m||_1} \quad j = 1, \ldots, n \tag{3.4b}$$

Multi-head highlighting attention

In our model, the encoder with dimension d_{model} consists of N layers and h heads. Each encoder layer contains the multi-head highlighting attention as a sub-layer. We proposed the multi-head highlighting attention mechanism, which employs the highlighting attention on p highlighted heads Head_i^H and the original scaled dot-product attention on the rest of $(h - p)$ normal heads Head_i^N. The multi-head highlighting attention is calculated as follows:

$$\begin{aligned} &\text{MultiHead}(Q, K, V) = \text{Heads}W^o \\ &\text{Heads} = \text{Concat}(\text{Head}_1^H, ..., \text{Head}_p^H, \text{Head}_{p+1}^N, ..., \text{Head}_h^N) \\ &\text{Head}_\text{i} = \text{Attention}(Q, K, V) \end{aligned} \tag{3.5}$$

[5]Since the number of key phrases is limited, and the highlighting matrix can be sparse, we mask the zero elements and only conduct the softmax operation on the nonzero elements.

The matrix Head_i is calculated by Eq. (2.2a). Within that equation, the attention weight matrix W^m of the highlighted head Head_i^H can be calculated by Eq. (3.2) or Eq. (3.4), while that of the normal head Head_i^N can be calculated by Eq. (2.2b). In Eq. (3.5), results of all the heads will be concatenated and then projected through a feed-forward layer, whose parameter matrix is $W^o \in \mathbb{R}^{hd_v \times d_{model}}$.

3.4 Datasets

We train and evaluate our model on an MDS dataset named Multi-News [38] and an SDS dataset named PubMed [23] to verify the effectiveness of our proposed methods on two summarization tasks (MDS and SDS) and datasets from two domains (news articles and academic literature).

In the Multi-News dataset [38], each example contains multiple news articles about the same event collected from diverse sources and a summary written by professional editors from newser.com. Cohan et al. [23] collected biomedical papers and built an SDS dataset named PubMed. Scientific papers are usually long documents. Their abstracts can be used as ground truth summaries. We find that the original PubMed dataset fails to separate abstracts from body sections in some examples, so we remove these abstracts from body sections in these examples.

The statistical information of these two datasets is shown in Table 3.1. Since Multi-News is an MDS dataset, the input document's length is calculated on the concatenation of all input documents in each example. We find that input documents in the PubMed dataset are notably longer than those of the Multi-News dataset. Additionally, academic literature's format and content organization are quite different from news articles, so we need to adopt different data preprocessing operations on these two datasets.

Table 3.1: Statistical information of two summarization datasets we used, "Pairs" denotes the number of examples, and "Words" denotes the average number of words in input documents and ground truth summaries.

Dataset	**Pairs**	**Words (Doc)**	**Words (Summary)**
Multi-News	56K	2,103	264
PubMed	133K	3,016	203

Table 3.2: Some common sections' contributions to abstracts in the PubMed dataset. 'Examples' denotes the percentage of examples containing each section. The average recalls of unigram, bigram, and longest common subsequence (LCS) are calculated by comparing each section with the abstract.

Sections	**Examples**	**Recall**		
		Unigram	**Bigram**	**LCS**
Introduction	76.3%	0.515	0.205	0.405
Discussion	69.6%	0.678	0.502	0.531
Result	56.4%	0.503	0.225	0.387
Conclusion	55.2%	0.304	0.123	0.272
Methods	54.0%	0.576	0.202	0.449
Case Report	28.8%	0.558	0.216	0.448
Analysis	21.6%	0.255	0.069	0.213

3.5 Experiments

3.5.1 Data preprocessing

To prepare the text data for training and evaluating the summarization model, we need to remove irrelevant content, filter out some outliers, change the format and length of text content, and split the dataset into training, validation, and test subsets. We lowercase all tokens in two datasets and perform sentence and word tokenization using NLTK [10]. More specific operations should depend on the nature of the dataset and the requirements of the summarization model.

For the Multi-News dataset, we follow the settings of data preparation in [38] and only keep examples with 2-10 input documents per summary. We take the first 500/S tokens from each document for the examples with S documents. If some input documents are shorter than 500/S, we follow [38] and iteratively adjust the quota for each document until reaching the 500-token limit. Then these truncated documents within one example are concatenated into a single document. We follow [38] to split the dataset into training (80%), validation (10%), and test (10%) sets.

For the PubMed dataset, we follow the settings in [23] and first filter out the outliers that are excessively long or too short. We also remove examples that do not contain the abstract. Figures and tables are removed, and we only preserve the text content. Since academic papers usually contain multiple sections and each section contributes differently to the abstract, we need strategies to preprocess these sections differently. We discover that the sections appearing after the conclusion section, like acknowledgments, conflict of interest, and sponsorship, usually do not contribute to the content of the abstract, so we remove these sections. Besides, some examples in the original PubMed dataset mix these abstracts' subsections with other body sections. We remove abstracts from these examples' body sections to prevent leaking target outputs into inputs of these examples.

In addition, we count section names of these papers in the PubMed dataset and find some common sections, including introduction, discussion, results, conclusion, methods, case report, and analysis. The "Examples" column in Table 3.2 summarizes the percentages of examples containing these common sections.[6] To calculate the average recalls of unigram, bigram, and longest common subsequence (LCS), we compare each of these common sections with the abstract, and we only count the examples whose inputs include that section.

Existing neural models' time or space complexity is usually highly correlated with the input sequence length. There is usually a limitation on the input sequence length to ensure the efficiency of the neural summarization model. Since the concatenation of these common sections can be excessively long, we still need to truncate them. We first count the number of common sections included in each paper. If one paper contains S common sections, we truncate each common section to 1000/S tokens. When some sections are shorter than 1000/S tokens, the excess quota will be equally distributed to other common sections. When the total length of these common sections is less than 1000, we equally assign the remaining quota to other body sections.

These truncated sections within one example are concatenated into a single document as the input. When increasing input length from 1000 to 2000 tokens, we do not find significant performance improvement, while training the neural models with a larger input size is more time-consuming. Following the settings in [23], we split the dataset into training (90%), validation (5%), and test (5%) sets.

3.5.2 Experimental setting

We adopt a 4-layer encoder and a 4-layer decoder to build the KPAT model. Each layer has eight attention heads. Both word embedding size and hidden size are set as

[6]We employ string matching on the section name to judge if each section belongs to a common section.

512. The maximum size of the vocabulary is set as 50000 by default. We also use label smoothing [126] with the smoothing factor 0.1 and dropout [123] with probability 0.2. The optimizer is Adam [60] with learning rate 2, β_1=0.9, and β_2=0.998. We also adopt the learning rate warmup over the first 8,000 steps and decay as in [131]. During decoding, we use beam search with a beam size of 5. The trigram blocking is used to reduce repetitions. We implement our model with OpenNMT-py [61]. All the models are trained on one NVIDIA QUADRO RTX 8000 GPU.

3.5.3 Baselines

We compare our proposed KPAT model with the following summarization methods. These methods can be roughly divided into two categories: extractive summarization methods and abstractive summarization methods. These models' details are shown in Table 3.11.

Extractive summarization methods

LexRank and TextRank[7] [37, 84] are two graph-based ranking methods that can be used for extractive summarization. They first build a sentence similarity graph and adopt the idea of PageRank [12] to score sentences. These sentences are sorted in descending order of their scores. Then top-ranked sentences are selected to form a summary.

Tf-idf [110] scores of words within a sentence can be summed to measure the sentence's importance. An extractive summarization method [22] is built based on this idea.

[7]We utilize the implementation of the LexRank model from https://pypi.org/project/lexrank/ and that of the TextRank model from https://radimrehurek.com/gensim_3.8.3/summarization/summariser.html

BertExt [78] stacks inter-sentence transformer layers on top of the pre-trained BERT model to capture document-level features. We follow settings in [78] and fine-tune the BERT model and inter-sentence transformer layers jointly on the training sets we used.

Abstractive summarization methods

PG and PG-MMR are models based on the pointer-generator network [63]. The pointer-generator network [116] allows both copying words from the input text and generating words from a vocabulary. Besides, it utilizes the coverage mechanism to discourage repetition in the generated text.

Hi-MAP [38] expands the PG network into a hierarchical network and calculates each sentence's Maximal Marginal Relevance (MMR) score. The attention distribution of each token within one sentence is multiplied by the MMR score of that sentence.

DAA [23] extends the pointer-generator network with discourse-aware attention. It consists of a hierarchical encoder modeling the discourse structure of each input document and an attentive discourse-aware decoder.

CopyTransformer [38, 41] adds the copy mechanism [116] to a 4-layer transformer model. Our KPAT model's decoder part adopts the decoder from this model.

SAGCopy [141] modifies the copy mechanism by adding words' centrality scores to the linearly transformed hidden state when calculating the copy distribution.

BertAbs [78] adopts the pre-trained BERT model as the encoder and randomly initializes a decoder comprising six transformer layers. We adopt the settings in [78] and fine-tune the model on the training sets we used.

3.5.4 Evaluation metrics

We use the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) F_1 scores [67] as the automatic evaluation metrics. Specifically, we report overlaps of unigrams (R-1), bigrams (R-2), and skip-bigrams with unigrams (R-SU) between system-generated summaries and reference summaries provided by summarization datasets.

ROUGE-N (R-N) is a statistic on n-gram co-occurring in both a candidate summary and a set of reference summaries. N is the number of words in the n-gram.

$$\text{R-N}_\text{r} = \frac{\sum_{S \in \text{ref}} \sum_{\text{gram}_\text{N} \in S} \text{Count}_\text{m}(\text{gram}_\text{N})}{\sum_{S \in \text{ref}} \sum_{\text{gram}_\text{N} \in S} \text{Count}(\text{gram}_\text{N})} \tag{3.6a}$$

$$\text{R-N}_\text{p} = \frac{\sum_{S \in \text{ref}} \sum_{\text{gram}_\text{N} \in S} \text{Count}_\text{m}(\text{gram}_\text{N})}{\sum_{S \in \text{cand}} \sum_{\text{gram}_\text{N} \in S} \text{Count}(\text{gram}_\text{N})} \tag{3.6b}$$

$$\text{R-N}_{\text{F}_1} = \frac{2 \times \text{R-N}_\text{p} \times \text{R-N}_\text{r}}{\text{R-N}_\text{p} + \text{R-N}_\text{r}} \tag{3.6c}$$

In Eq. (3.6), S represents the sentence in summaries. $\text{Count}_\text{m}(\text{gram}_\text{N})$ is the maximum number of n-grams co-occurring in both a candidate summary and a set of reference summaries. R-N_r, R-N_p, and R-N_{F_1} represent the recall, precision, and F_1 score of ROUGE-N. We employ the F_1 scores of ROUGE-1 (R-1) and ROUGE-2 (R-2) for assessing generated summaries' informativeness [65, 77].

ROUGE-S (R-S) is a co-occurrence statistic on the skip-bigram. In a sentence, each skip-bigram is an ordered pair of words allowing for arbitrary gaps between them. Given a sentence $\text{sent}_i = [w_1, w_2, ..., w_n]$ comprising multiple words in a candidate summary, a pair of words within the sentence (w_{j_1}, w_{j_2}) is a skip-bigram if $j_1 < j_2$. ROUGE-S does not require consecutive matching, but it is still sensitive to word order [67]. It counts all in-order matching word pairs and can be computed as follows:

$$\text{R-S}_\text{r} = \frac{\text{SKIP}(X, Y)}{C(m, 2)} \tag{3.7a}$$

$$\text{R-S}_\text{p} = \frac{\text{SKIP}(X, Y)}{C(n, 2)} \tag{3.7b}$$

$$\text{R-S}_{\text{F}_1} = \frac{2 \times \text{R-S}_\text{r} \times \text{R-S}_\text{p}}{\text{R-S}_\text{r} + \text{R-S}_\text{p}} \tag{3.7c}$$

$$\text{SKIP}(X, Y) = \sum_{S \in X} \sum_{\text{s-gram}_i \in S} \text{Count}_\text{m}(\text{s-gram}_i) \tag{3.7d}$$

In Eq. (3.7), s-gram_i is a skip-bigram. m is the length of the reference summary X. n represents that of the generated candidate summary Y. $C(m, 2)$ and $C(n, 2)$ are numbers of skip-bigrams in X and Y. Besides, $\text{SKIP}(X, Y)$ is the number of matched skip-bigrams between X and Y. R-S_r, R-S_p, and R-S_{F_1} represent the recall, precision, and F_1 score of ROUGE-S.

However, ROUGE-S does not consider that some generated sentences may not include skip-bigrams. ROUGE-SU extends the ROUGE-S by adding unigram as an additional counting unit. It can be implemented by adding a marker at the beginning of the candidate and reference sentences [67].

$$\text{ROUGE-SU}(X, Y) = \text{ROUGE-S}(X^+, Y^+) \tag{3.8a}$$

$$\text{SKIP}(X^+, Y^+) = \text{SKIP}(X, Y) + \text{Uni-CNT} \tag{3.8b}$$

$$\text{Uni-CNT} = \sum_{S \in X} \sum_{\text{Unigram}_i \in S} \text{Count}_\text{m}(\text{Unigram}_i) \tag{3.8c}$$

In Eq. (3.8), X^+ and Y^+ denote the reference and candidate summary added a start token. Uni-CNT is the maximum number of unigrams co-occurring in both the candidate summary and a set of reference summaries.

3.6 Results and discussion

In this section, we present and analyze our experimental results. To compare the quality of summaries generated by our KPAT model and various advanced baselines, we conduct automatic and human evaluations and analyze evaluation results in subsection 3.6.1 and 3.6.2. Besides, we conduct more experiments to analyze the impact of each part of our model on the summarization performance. We compare the effects of adopting the highlighting attention in different numbers of heads and layers in the encoder of our KPAT model in subsection 3.6.3. We also compare the impacts of introducing different numbers of key phrases from different key phrase extractors into the KPAT model in subsection 3.6.4. To validate the effectiveness of each component in our proposed KPAT model, we adopt ablation studies and report results in subsection 3.6.5.

3.6.1 Automatic evaluation results

In the automatic evaluation, we employ the ROUGE scores [67] of generated summaries on test sets as the evaluation metrics and compare the summaries generated by our KPAT model with those of baseline models.

Automatic evaluation results of LexRank, TextRank, PG, PG-MMR, Hi-MAP, and CopyTransformer on the Multi-News test set follow Fabbri et al. [38]. For the PubMed dataset, we train and evaluate these models since we choose a different truncation strategy compared with the original scheme in [23] and remove abstracts from body sections in some examples that fail to separate them. In addition to the baseline models used in [23, 38], we add two additional extractive baselines and two abstractive baselines introduced in subsection 3.5.3. The tf-idf-based extractive method [22] is adopted as a baseline to compare with introducing the tf-idf score into our abstractive model. The BERT-based extractive summarization method named BertExt and

Table 3.3: Automatic evaluation results on the Multi-News test set.

Method	**R-1**	**R-2**	**R-SU**
LexRank	38.27	12.70	13.20
TextRank	38.44	13.10	13.50
tf-idf	38.68	12.09	13.54
BertExt	44.27	15.09	17.44
PG	41.85	12.91	16.46
PG-MMR	40.55	12.36	15.87
Hi-MAP	43.47	14.89	17.41
BertAbs	42.21	15.14	16.33
SAGCopy	43.98	15.21	17.65
CopyTransformer	43.57	14.03	17.37
KPAT (Weighted)	**45.30**	**15.96**	**18.62**
KPAT (Additive)	44.37	15.55	17.77

the abstractive summarization method named BertAbs in [78] are fine-tuned on our training sets and evaluated on our test sets. The SAGCopy [141] is also trained and evaluated on these two datasets we used.

We report the F_1 scores of ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-SU (R-SU) in Table 3.3 and 3.4. "KPAT (Weighted)" denotes the KPAT model equipped with the weighted highlighting attention, and "KPAT (Additive)" represents the KPAT model equipped with the additive highlighting attention. For the Multi-News dataset, we report the results of the KPAT model based on the top-20 key phrases extracted by the PositionRank. For the PubMed dataset, we report the results of the KPAT model based on the top-10 key phrases extracted by the PositionRank.

Our proposed model significantly outperforms these baseline models on all metrics.

Table 3.4: Automatic evaluation results on the PubMed test set.

Method	**R-1**	**R-2**	**R-SU**
LexRank	35.78	14.75	11.35
TextRank	36.41	14.97	11.90
tf-idf	33.67	9.18	10.74
BertExt	37.72	13.95	12.48
PG	38.37	13.59	14.72
DAA	38.95	15.41	15.63
BertAbs	39.29	15.59	15.84
SAGCopy	38.66	15.24	15.35
CopyTransformer	38.81	14.99	15.39
KPAT (Weighted)	**40.04**	**15.82**	**16.24**
KPAT (Additive)	39.67	15.61	15.94

These results prove the effectiveness of the highlighting mechanism on two summarization tasks (MDS and SDS) and datasets from two domains (news articles and academic literature). Besides, the weighted highlighting attention performs better than additive highlighting attention. According to Eq. (3.2) and (3.3), the weighted highlighting attention can take advantage of the exponential operation in the softmax function to amplify the influence of key phrases' importance scores. Consequently, the highlighted heads adopting the weighted highlighting attention can enlarge the contributions of tokens within the same key phrase to the contextual representation of each token in that phrase.

Table 3.5: Human evaluation results on the test sets of Multi-News and PubMed, "Win" represents the generated summary of our KPAT model is better than that of the CopyTransformer in one aspect. "Tie" denotes two summaries are comparable in one aspect.

	Win	**Lose**	**Tie**	**Kappa**
Multi-News dataset				
Informativeness	44.5%	21.0%	34.5%	0.656
Fluency	31.5%	29.0%	39.5%	0.647
Non-Redundancy	28.0%	23.5%	48.5%	0.614
PubMed dataset				
Informativeness	43.0%	19.5%	37.5%	0.659
Fluency	27.0%	25.0%	48.0%	0.622
Non-Redundancy	23.5%	19.0%	57.5%	0.631

3.6.2 Human evaluation results

We perform the human evaluation to compare the quality of summaries generated by our KPAT model and the CopyTransformer model. This human evaluation mainly focuses on three metrics: informativeness (the coverage of information from input documents), fluency (content organization and grammatical correctness), and non-redundancy (less repetitive information). We randomly select 50 samples from the test sets of Multi-News and PubMed, respectively. We invite four annotators to compare summaries generated by two models. These summaries are presented anonymously. Besides, we assess annotators' agreements by Fleiss' kappa [39].

Human evaluation results in Table 3.5 suggest that our proposed model significantly outperforms the CopyTransformer in terms of informativeness and is comparative in

Table 3.6: Evaluation results of highlighting different numbers of heads and layers on the Multi-News test set.

KPAT (Weighted)	**R-1**	**R-2**	**R-SU**
1/4 Heads 1/2 Layers	**45.30**	**15.96**	**18.62**
1/2 Heads 1/2 Layers	44.61	15.60	18.16
All Heads 1/2 Layers	44.42	15.36	17.92
1/4 Heads All Layers	44.67	15.54	18.11
1/2 Heads All Layers	44.58	15.43	18.02
All Heads All Layers	44.35	15.23	17.90

terms of fluency and non-redundancy on these two datasets we used.

3.6.3 Impact of the multi-head highlighting attention

We compare the effects of adopting the weighted highlighting attention in different numbers of heads and layers in the encoder of our proposed model. In this experiment, we adopt the weighted highlighting attention mechanism on each highlighted head of our encoder. Table 3.6 summarizes results on the test set of Multi-News. It shows that adopting the weighted highlighting attention in a quarter of the heads and half of the layers achieves the best performance. We discover that adopting highlighting attention in a subset of heads surpasses adopting it in all heads. Applying the multi-head highlighting attention in all the encoder layers is also not optimal.

Multi-head attention in the transformer model [131] is designed for jointly attending to information from different representation sub-spaces. Voita et al. [132] find that heads in the transformer model trained on the neural machine translation dataset have specialized functions and focus on different types of information, including the

Table 3.7: Adopting different settings of key phrase selection on the Multi-News test set.

Key phrase extractor	**R-1**	**R-2**	**R-SU**
tf-idf (top-10)	44.56	15.63	18.00
tf-idf (top-20)	44.84	15.80	18.21
TopicRank (top-10)	44.53	15.29	17.97
TopicRank (top-20)	45.24	15.93	18.56
PositionRank (top-10)	44.70	15.73	18.12
PositionRank (top-20)	**45.30**	**15.96**	**18.62**

adjacent tokens, syntactic relations, and rare words. Adopting the highlighting attention in all heads and layers is not conducive to encoding other types of useful information and leads to performance degradation.

3.6.4 Impact of the key phrase extraction

Since our KPAT model relies on extracted key phrases to construct highlighting metrics, the precision and coverage of introduced key phrases are crucial. In this subsection, we compare the impacts of introducing different numbers of key phrases from different key phrase extractors into our proposed KPAT model.

There are usually no labels of key phrases in existing summarization datasets, so we only focus on unsupervised extraction methods. We adopt the tf-idf-based extractor and two graph-based ranking methods: TopicRank [11] and PositionRank [40], to score the importance of phrases and extract key phrases from input documents. Based on their extraction results, we build highlighting matrices as a part of the input of our summarization model.

Table 3.8: Adopting different settings of key phrase selection on the PubMed test set.

Key phrase extractor	**R-1**	**R-2**	**R-SU**
tf-idf (top-10)	39.88	15.70	16.06
tf-idf (top-20)	39.43	15.55	15.96
TopicRank (top-10)	39.48	15.73	15.91
TopicRank (top-20)	39.28	15.58	15.85
PositionRank (top-10)	**40.04**	**15.82**	**16.24**
PositionRank (top-20)	39.64	15.70	15.99

Evaluation results in Table 3.7 and Table 3.8 suggest that selecting the top-10 key phrases performs well on the PubMed dataset. Considering PubMed is an SDS dataset, ten key phrases can be enough for a single input document. For the MDS dataset Multi-News, multiple news articles in each example usually contain more diverse phrases, including names of events, persons, locations, and organizations. Ten phrases are not enough to cover them, so we decide to extract the top-20 phrases from the input text in each example of Multi-News. We discover that selecting the top-20 key phrases performs better on the Multi-News dataset.

When it comes to the impact of different key phrase extractors, introducing key phrases extracted by the PositionRank algorithm [40] achieves the best results on the two summarization datasets we used. PositionRank assigns larger probabilities to words found early or frequently in a given document. It can meet the phenomenon that key phrases usually appear near the beginning of a document or appear frequently, which is common in news articles and academic literature [40].

For further analysis, we need to assess the performance of used key phrase extractors. Unfortunately, there are usually no labels for key phrases in existing summarization

Table 3.9: Evaluation results on our labeled test sets for key phrase extraction on the Multi-News and PubMed. "P%" is the precision. "R%" is the recall. "Exact" denotes the exact match. "Contain" represents one gold phrase that is contained in one predicted phrase. "Cover" is the number of predicted phrases contained in target summaries.

	P% Exact	**R% Exact**	**P% Contain**	**R% Contain**	**Num Cover**
Multi-News (top-20)					
tf-idf	23.9	19.8	25.0	20.6	2.95
TopicRank	39.4	38.2	45.2	43.7	3.38
PositionRank	45.6	43.5	54.2	51.6	3.72
PubMed (top-10)					
tf-idf	23.6	23.9	35.6	33.2	2.83
TopicRank	22.4	22.6	34.4	29.2	2.20
PositionRank	30.5	30.0	52.9	37.3	3.21

datasets. Therefore, we randomly selected 50 examples from training sets of the Multi-News and PubMed, respectively. Then we invite annotators to label key phrases (bigrams and trigrams) in input texts of these selected examples.

We evaluate three key phrase extractors on our labeled test sets. Considering that gold phrases labeled by humans can be part of the phrases in input texts, we not only calculate the precision and recall based on exact matching but also count predicted phrases containing a gold phrase. Because our target is to completely encode key phrases in input texts, we do not accept predicted phrases contained in gold phrases. Additionally, we count the number of predicted phrases contained in target summaries.

Table 3.10: Ablation study on the Multi-News and PubMed datasets. "w/o block linear" denotes removing the block-wise linear transformation on the block diagonal highlighting matrix, "w/o highlighting attention" is replacing the highlighting attention with the original self-attention, and "w/o self-attention" represents replacing self-attention weight matrices with highlighting matrices.

	R-1	**R-2**	**R-SU**
Multi-News dataset			
KPAT model	**45.30**	**15.96**	**18.62**
w/o block linear	44.62	15.57	18.06
w/o highlighting attention	43.57	14.03	17.37
w/o self-attention	42.82	14.66	16.71
PubMed dataset			
KPAT model	**40.04**	**15.82**	**16.24**
w/o block linear	39.68	15.62	15.95
w/o highlighting attention	38.81	14.99	15.39
w/o self-attention	37.63	14.87	15.14

Table 3.9 shows these key phrase extractors' performance. PositionRank performs the best on two datasets. TopicRank performs better than tf-idf on the Multi-News, while tf-idf performs better on the Pubmed dataset. Compared with the summarization results of our KPAT model in Table 3.7 and 3.8, we discover that a better key phrase extractor can bring performance gains to our KPAT model since our model relies on extracted phrases to construct highlighting metrics. This positive correlation also reveals our highlighting mechanism's effectiveness in introducing key phrases' information.

Table 3.11: Details of summarization models.

Model	Type	Architecture	Params	Enc/Dec Layers	Input Len
PG	LSTM	Enc-Dec	42.8M	2	-
BertAbs	Transformer	Enc-Dec	180.6M	12	1,024
CopyTransformer	Transformer	Enc-Dec	81.5M	4	1,024
KPAT	Transformer	Enc-Dec	81.5M	4	1,024

3.6.5 Ablation study

We conduct ablation studies to validate the effectiveness of each component in our proposed model. Table 3.10 summarizes ablation studies' results on the Multi-News and PubMed datasets. These results confirm that incorporating the highlighting attention and the block-wise linear transformation on the block diagonal highlighting matrix can benefit both single-document summarization and multi-document summarization. In addition, we also tried directly replacing the self-attention weight matrices with the highlighting matrices in a quarter of the heads and half of the layers. The performance degradation reveals that combining attention weights with phrases' importance scores outperforms simply replacing the self-attention mechanism.

3.7 Chapter Summary

In this chapter, we propose the key phrase aware transformer (KPAT), a lightweight model achieving great performance on multiple abstractive summarization tasks. This work focuses on enhancing the transformer encoder to completely encode the key phrases in input documents. We present the highlighting mechanism incorporating the prior knowledge of key phrases when calculating attention weights for tokens within key phrases. The results of our comparative experiments verify the effectiveness of the highlighting mechanism.

Chapter 4

From Unimodal to Multimodal: Long Text and Multi-Table Summarization

4.1 Introduction

Report documents, like financial reports, investigative reports, and technical reports, are essential information sources. These report documents usually contain large amounts of textual and tabular content and provide rich knowledge about companies, industries, technologies, etc. Each report's salient information can be scattered in long text and multiple tables in different sections, which makes it difficult for non-specialized readers to efficiently read these report documents. A high-quality summary of each report document can help readers quickly browse key information. Automatic document summarization techniques can be utilized to produce reports' summaries. Users can flexibly adjust the input document and immediately get a summary from the automatic summarization system. Our target is to let the computer generate an informative, fluent, and non-redundant summary for the long text and

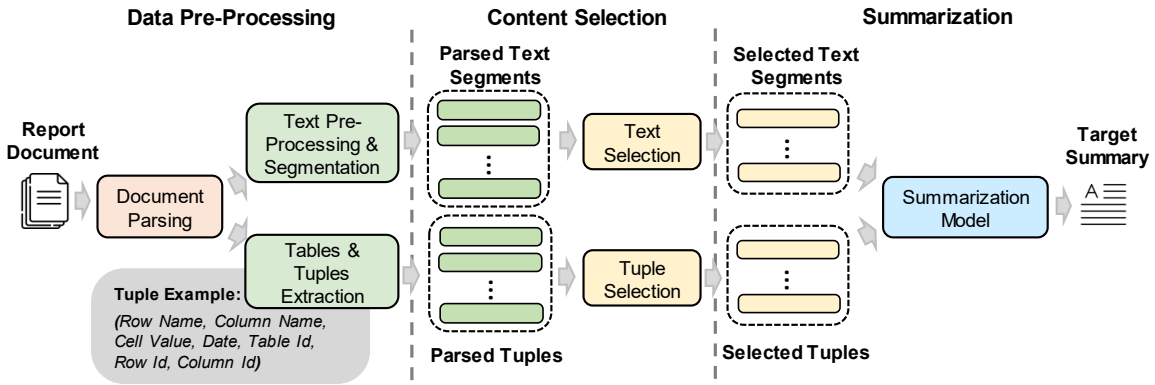


Figure 4.1: An overview of our solution for long text and multi-table summarization.

multiple tables in each report document. To achieve this target, we need to deal with some challenging issues: 1) the scarcity of available datasets, 2) identifying the salient information scattered in a large amount of input content, 3) incorporating different types of content when generating summaries, and 4) models' efficiency in processing long inputs and outputs.

Previous document summarization datasets usually focus on text. Non-textual content is usually regarded as noises and filtered out. When target summaries only focus on narratives and qualitative descriptions, removing non-textual content has little effect since the document's text already contains most of the required information. When it comes to report documents, like financial reports, their summaries should cover both the narrative content and quantitative descriptions of critical metrics recorded in tables, which are essential for readers' analysis and decision-making [115]. Existing datasets cannot meet the requirements of summarizing long text and multiple tables in each report document.

To deal with the scarcity of available datasets, we propose FINDSum, the first large-scale dataset for long text and multi-table summarization[1]. FINDSum has two subsets

[1]FINDSum dataset is available for download online at: https://github.com/StevenLau6/FINDSum

named FINDSum-ROO and FINDSum-Liquidity for summarizing companies' results of operations and liquidity. Inputs of each example in FINDSum include tens of thousands of words and dozens of tables from a report document. Table 4.1 shows that FINDSum's target summaries usually contain more numerical values than previous datasets. Meanwhile, most numerical values in target summaries cannot be found in the corresponding input text. Only focusing on text is not enough to summarize these financial reports.

We propose a solution for long text and multi-table summarization to cope with the other three issues. As shown in Fig. 4.1, our solution has three main steps: data pre-processing, content selection, and summarization. To efficiently identify the scattered key information, we add the content selection step as a rough selection over the long inputs, and then the summarization step conducts a finer selection. The content selection step aims to compress long inputs while maximizing the recall of salient content in long text and dozens of tables. Specifically, we adopt the Maximum Marginal Recall Gain (MMRG) method to select salient text segments. As for the tabular content, we transform each table cell into a tuple and regard the salient tuple selection as a binary classification problem. The summarization step should jointly consider different types of inputs. To incorporate text and tabular content, we present four types of summarization methods: generate-then-combine (GC), combine-then-generate (CG), generate-template-then-fill (GTF), and generate-combine-then-generate (GCG).

The complexity of the transformer's self-attention mechanism scales quadratically with the input length [131], which limits transformer-based models' efficiency. Thus, we employ content selection methods and sparse attention mechanisms to reduce the complexity and enable fine-tuning large pre-trained models over long inputs on an off-the-shelf GPU. Besides, existing autoregressive models still have difficulty in generating long sequences [54, 105]. We employ a divide-and-conquer approach to generate summary segments in parallel and then merge them as the final summary.

Table 4.1: Statistical information of summarization datasets. "Pairs" is the number of examples. "Words" and "Sents" denote the average number of words and sentences in input text or target summary. "Num" is the average number of numerical values in target summaries, and "Cov Num" is the ratio of the target summary's numerical values found in the input text. "Cov." and "Dens." are the extractive fragment's coverage and density [44].

Dataset	**Pairs**	**Words (Doc)**	**Sents (Doc)**	**Words (Sum)**	**Sents (Sum)**	**Num (Sum)**	**% Cov Num**	**Cov.**	**Dens.**
CNN/DM	312k	810.6	39.8	56.2	3.7	0.6	78.7	0.9	3.8
PubMed	133k	3049.0	87.5	202.4	6.8	3.3	68.2	0.8	5.8
arXiv	216k	6029.9	205.7	272.7	9.6	0.7	53.9	0.9	3.8
ROO	21k	45,566.0	1250.5	660.7	16.3	**24.3**	**26.3**	0.9	9.7
Liquidity	21k	45,566.0	1250.5	1,057.6	26.7	**32.3**	**41.2**	0.9	9.6

We benchmark advanced extractive and abstractive summarizers as baselines on our FINDSum dataset. To compare their performance, we conduct automatic evaluation and human evaluation. In addition to the commonly used ROUGE scores [67], we propose a set of evaluation metrics to assess the usage of numerical information in produced summaries. Experimental results show that our methods can outperform competitive baselines.

We also conduct extensive comparative experiments and a case study to compare and analyze the influence of model components and configurations on summarization results. We find the input sequence length, content selection methods, divide-and-conquer method, sparse attention mechanism, and pre-trained model can greatly affect summarization results. Experimental results also verify the effectiveness of our methods in content selection and summarization.

Our contribution is fourfold:

- We build FINDSum, the first large-scale dataset for long text and multi-table summarization.
- We present and compare four types of methods incorporating text and tables into summary generation.
- We propose evaluation metrics to assess the usage of numerical information in generated summaries.
- We find vital components and configurations of models that improve summarization results.

4.2 FINDSum Dataset

Financial report document summarization (FINDSum) is the first large-scale dataset for long text and multi-table summarization. This section introduces our data collection and pre-processing procedures and describes FINDSum's two subsets. We conduct descriptive statistics and in-depth analysis on FINDSum and compare it with other datasets.

4.2.1 Data Collection and Pre-Processing

Form 10-K is the annual report that comprehensively describes a company's financial performance in the prior fiscal year [115]. We collected thousands of companies' last ten 10-K forms' HTML files from the Electronic Data Gathering, Analysis, and Retrieval (EDGAR) system[2]. The U.S. Securities and Exchange Commission (SEC) makes companies' 10-K forms publicly available through the EDGAR system. The

[2]www.sec.gov/edgar/searchedgar/companysearch.html

SEC stipulates the 10-K form's format and required content. It usually has four parts and sixteen items [114]. The item "Management's Discussion and Analysis of Financial Condition and Results of Operations" (MD&A) contains the management's summary of the company's results of operations and liquidity [89]. FINDSum uses the text in MD&A's two sections: "results of operations" and "liquidity and capital resources" as target summaries and the remaining content of each report document as the input.

After collecting tens of thousands of 10-K forms' HTML files, we parse them and split text and tables. To keep tables' positional information and align tables and text, we add a special token containing each table's index to concatenate the text before and after the table. Extracted text and tables are stored in separate files. Text and tabular data require different pre-processing procedures, considering their different natures. Our text pre-processing procedures include: removing noises (e.g., cover pages and special characters composing a style) and dividing text in different parts of 10-K form into text segments. To pre-process tabular data, we extract table content (e.g., names of rows and columns, cell content), remove noises in table content, and transform each cell into a tuple: (row name, column name, cell value, date, table index, row index, column index). The cell value in the tuple concatenates the original cell value and the rounding result with an ampersand. Besides, we remove duplicate samples and outliers with too-short input text, truncate too-long input text, split the training (80%), validation (10%), and test (10%) sets. Considering that the same company's annual reports in different years usually have duplicate content, we split these three sets by company to minimize their overlaps.

4.2.2 Dataset Description

FINDSum dataset is built on collected report documents. It has two subsets: FINDSum-ROO and FINDSum-Liquidity.

Table 4.2: The proportion of novel n-grams in target summaries.

Dataset	**% of novel n-grams in target summary**			
	unigrams	**bigrams**	**trigrams**	**4-grams**
CNN/DM	19.50	56.88	74.41	82.83
PubMed	18.38	49.97	69.21	78.42
arXiv	15.04	48.21	71.66	83.26
FINDSum-ROO	17.79	50.59	72.13	81.66
FINDSum-Liquidity	26.45	59.63	80.43	88.48

FINDSum-ROO is the subset focusing on each company's results of operations (ROO). In the ROO section of MD&A, the company's management usually compares and explains critical items of revenue and expense in the current and prior period [114]. This section's text can be regarded as the target summary written by experts. Table 4.1 exhibits that the average number of numerical values in FINDSum-ROO's target summaries is dozens of times larger than that of previous datasets. However, nearly three-quarters of these numerical values cannot be found in these reports' remaining text. A lot of critical numerical information is only recorded in tables. Therefore, we use the remaining parts' text and all the tables in each report as inputs for each example.

FINDSum-Liquidity focuses on summarizing each company's liquidity and capital resources. The "liquidity and capital resources" section in MD&A mainly analyzes the company's ability to generate and obtain cash [89]. This section's text can be used as the target summary. Most of the numerical values in target summaries are not included in the remaining parts' text. FINDSum-Liquidity's inputs include the remaining text and all the tables in each report.

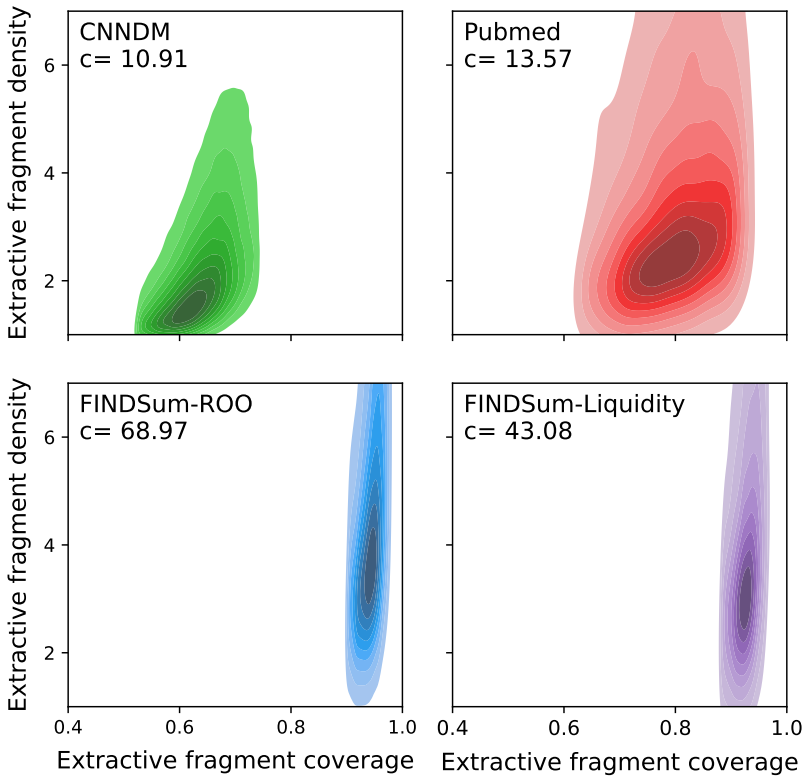


Figure 4.2: Distributions of extractive fragments' density and coverage.

4.2.3 Dataset Analysis

We conduct statistics and analysis on FINDSum's two subsets. Table 4.1 shows that both the input documents and target summaries of these two subsets are much longer than those of previous summarization datasets. These two subsets' target summaries contain much more numerical information, while most of them cannot be found in the input text.

To measure how abstractive FINDSum's target summaries are, we count the percentage of summaries' novel n-grams not appearing in inputs. Table 4.2 shows that FINDSum-Liquidity's target summaries have more novel n-grams and are more abstractive. The abstractiveness of FINDSum-ROO's target summaries is similar to that of other datasets.

We also adopt three measures defined by Grusky et al. [44] to assess the extractive nature of summarization datasets. Given a document $D = [d_1, d_2, ..., d_n]$ consisting of a sequence of tokens d_i and its summary $S = [s_1, s_2, ..., s_m]$, extractive fragments $F(D, S)$ is the set of shared token sequences in D and S. In Eq. (4.1a), extractive fragment coverage measures the percentage of summary words that are part of an extractive fragment from the input document. Eq. (4.1b) calculates the extractive fragment density assessing the average length of the extractive fragment to which each summary word belongs. Besides, the compression ratio is the word ratio between the articles and their summaries, as shown in Eq. (4.1c). We report the extractive fragment coverage and density in Table 4.1. Two measures' distributions are visualized using kernel density estimation in Fig. 4.2. FINDSum's density is higher than those of previous datasets. The variability along the y-axis (density) suggests the varying writing styles in its target summaries.

$$\text{COVERAGE}(D, S) = \frac{1}{|S|} \sum_{f \in F(D,S)} |f| \tag{4.1a}$$

$$\text{DENSITY}(D, S) = \frac{1}{|S|} \sum_{f \in F(D,S)} |f|^2 \tag{4.1b}$$

$$\text{COMPRESSION}(D, S) = \frac{|D|}{|S|} \tag{4.1c}$$

4.3 Method

Summarizing long text and multiple tables has several challenging issues: identifying the salient information from a large amount of input content, incorporating the text and tabular content into the summary generation, and efficiently processing long input and output sequences. This section presents our solution to the above issues.

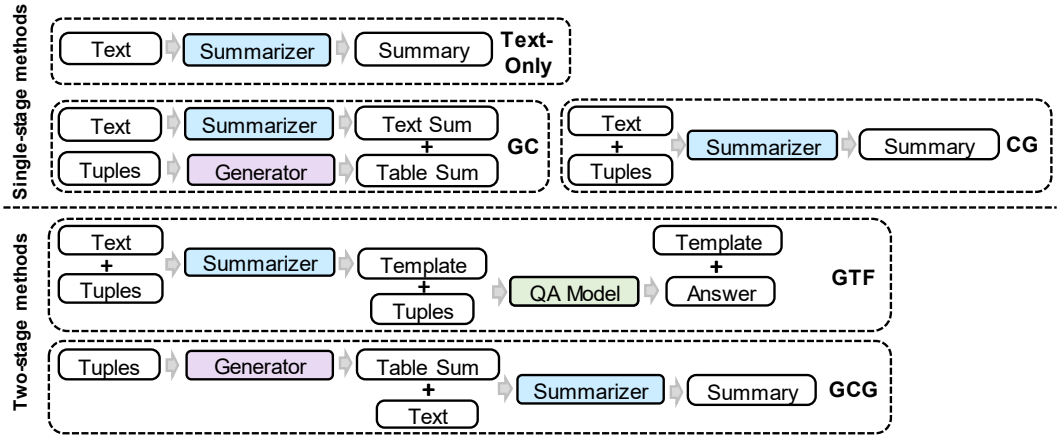


Figure 4.3: An overview of our summarization methods.

4.3.1 Textual and Tabular Content Selection

Fig. 4.1 shows the three main steps of our solution: data pre-processing, content selection, and summarization. After the pre-processing, we get dozens of text segments and thousands of tuples from dozens of tables in each report. It is challenging to accurately identify the scattered salient content. We add the content selection step as a rough selection to compress long inputs while maximizing the recall of salient content that should be preserved in summaries. Then compressed inputs are fed into the summarizer for further selection. Content selection methods' output lengths should not exceed pre-specified lengths, as neural summarization models' complexity can scale with input sequence length.

We employ separate methods to select salient content from textual and tabular data considering their different natures. To select salient text segments, we adopt a method named Maximum Marginal Recall Gain (MMRG) on our training set. Specifically, MMRG keeps adding the text segment bringing the maximum gain of n-gram's recall into the combination of selected segments till reaching the length limit. Finally, we can get selected salient segments' indexes and choose text segments with the same indexes for samples in our test set. Algorithm 1 is MMRG's pseudocode. We also

follow Liu et al. [70] to try some extractive summarizers, like Textrank [84] and Lexrank [37], for salient text selection. Experimental results in subsection 4.5.2 show that MMRG performs the best, so we use it in our experiments.

Algorithm 1 Maximum Marginal Recall Gain (MMRG)

Input: Input m examples $I \leftarrow [e_1, ..., e_m]$, each example e_i contains n parts for selection $e_i \leftarrow [p_i^1, ..., p_i^n]$, the list of target item $T \leftarrow [t_1, ..., t_m]$, and the maximum number of selected parts n' ($n' \ll n$)

Output: The list of selected parts' id $S \leftarrow [j, ..., k]$ and the selected inputs $I' \leftarrow [e'_1, ..., e'_m]$, in which each example e'_i has selected parts $e'_i \leftarrow [p_i^j, ..., p_i^k]$ ($|e'_i| = |S| \leq n'$)

$S \leftarrow [\,]$;
$e'_1, ..., e'_m \leftarrow$ "", ..., "";
$I' \leftarrow [e'_1, ..., e'_m]$;
while $|S| < n'$ **do**
  //SelectPart finds the part $p^{j_{select}}$ bringing the largest average recall gain across all examples
  $j_{select} = \text{SelectPart}(I, I', T, S)$;
  if $\text{j}_{\text{select}} > 0$ **then**
    $S \leftarrow S \cup [j_{select}]$;
    while $i \leq m$ **do**
      $I'[i] \leftarrow \text{Concat}(I'[i], p_i^{j_{select}})$;
    end while
  end if
end while

As for those thousands of tuples extracted from tables, we model the salient tuple selection as a binary classification problem. Based on the FINDSum dataset, we annotate a tuple selection dataset for training and evaluating different classifiers (e.g., logistic regression, support vector machine, AdaBoost [50], XGBoost [15], and Multi-

layer Perceptron)[3]. We also utilize various features, including positional features (e.g., indexes of the row, column, table, and section, together with their normalized values) and text features (e.g., word embedding of row and column names). Considering the content selection step focuses more on the recall of salient content, we sort these tuples by their positive probability predicted by the trained classifier and use the top-n tuples' recall to evaluate these classifiers. Tables 4.11 and 4.12 show that the XGBoost and MLP models equipped with positional features and Glove embedding [99] outperform others. We adopt them for tuple selection and compare their impact on the produced summaries in subsection 4.5.2. We follow the setting in [71] to flatten the selected tuples into a linearized sequence.

4.3.2 Generating Summary for Textual and Tabular Data

To incorporate text and tabular data into summary generation, we present four types of methods: Generate-then-Combine (GC), Combine-then-Generate (CG), Generate-Template-then-Fill (GTF), and Generate-Combine-then-Generate (GCG). We show their structures in Fig. 4.3.

GC method makes two assumptions: 1) The summary of long text and multiple tables can be divided into text summary and table summary. 2) Summary generation can be divided into two parallel processes generating these two parts of summary. It assigns the maximum output lengths for the text summary and table summary, generates these two summaries separately, and concatenates them to form the final summary. GC has obvious limitations: 1) It cannot merge the information from text and tables when generating each summary sentence. 2) The pre-defined length assignment is not flexible enough to adapt to diverse examples.

CG is an end-to-end method generating a summary for both text and table content.

[3]We use XGBoost's implementation from xgboost.readthedocs.io/ en/stable/ and other classifiers from scikit-learn

It first concatenates the selected text segments and tuples with a special symbol and then feeds them into a sequence-to-sequence summarizer. The summarizer needs to learn both text-to-text and tuple-to-text generation. When generating summaries, it should jointly consider these two types of input content.

GTF method is inspired by how humans write quantitative descriptions of report documents. The CG and GC methods use similar processes to generate qualitative and quantitative descriptions, while the way people write quantitative descriptions differs from the way they write qualitative descriptions. Specifically, people usually decide which metrics to describe and then read tables to find numerical values to fill in the quantitative descriptions. When writing qualitative descriptions, they mainly refer to text content. GTF method has two stages: template generation and template filling, which mimic how humans write quantitative descriptions. The template generation stage generates all the words and the special token [num] as the placeholder for numerical values from tables. We regard the template filling as a question answering (QA) task. We use each template sentence containing the placeholder as a question and the linearized sequence of selected tuples as the context. We train the QA model to find the numerical value from table content as an answer to replace the placeholder. Table 4.16 shows that Bigbird's large model performs the best on the ROO subset. Longformer's large model performs the best on the Liquidity subset. We use them for the template filling in GTF.

GCG is another two-stage method. It employs a tuple-to-text generator to produce input tuples' text descriptions, concatenates the input text with the tuples' descriptions, and feeds them into the summarizer. Compared with the CG, GCG simplifies the requirement on the summarizer to focus on summarizing text, but the extra tuple-to-text generation process can lose some tuples' information. We annotate a tuple-to-text generation dataset based on our FINDSum dataset for training and evaluating various generators. Table 4.18 indicates that the BART-large outperforms other baselines, so we use it as the tuple-to-text generator in GCG.

4.3.3 Processing Long Inputs and Outputs

Input documents in our FINDSum-ROO and FINDSum-Liquidity subsets contain tens of thousands of words. The average length of target summaries in FINDSum-Liquidity exceeds 1,000 words. Long inputs and outputs bring some problems: 1) The transformer model's self-attention mechanism [131] scales quadratically with the input sequence length. It is prohibitively expensive for long input [20] and precludes the usage of large pre-trained models with limited computational resources. 2) Existing autoregressive abstractive summarization methods still have difficulty in generating long text in terms of efficiency and quality [54, 105]. To deal with the first problem, we employ sparse attention mechanisms [6, 147] in our summarization models' encoders. The content selection step in our solution also reduces the length of input sequences. To handle the second problem, we follow a divide-and-conquer method [42] and decompose the long summary generation problem into multiple sub-problems of summary segment generation. These summary segments can be generated in parallel and merged as a final summary. To minimize output summaries' redundancy, we add a constraint that the MMRG in the content selection step should not select the same combination of input text segments for generating different summary segments.

4.4 Experiments

4.4.1 Baselines

In our experiments, we adopt advanced extractive and abstractive summarization models as baselines. These models' details are shown in Table 4.20.

LexRank and TextRank [37, 84] are two graph-based ranking methods that can be used for unsupervised extractive summarization.

BART [64] is a denoising autoencoder built with a sequence-to-sequence model pre-

trained to reconstruct the original input text from the corrupted text.

PEGASUS [149] is a transformer-based model pre-trained with the Gap Sentences Generation (GSG) and Masked Language Model (MLM) objectives.

LongT5 [45] extends the original T5 encoder [103] with a global-local attention mechanism to handle long inputs.

BigBird-PEGASUS [147] adopts the BigBird encoder with sparse attention mechanisms and the PEGASUS decoder.

Longformer-Encoder-Decoder (LED) [6] follows BART's architecture and adopts sparse attention in its encoder.

4.4.2 Experimental Setting

The vocabulary's maximum size is set as 50,265 for summarization models, while the tuple-to-text generators use 32,128 as default. When fine-tuning these pre-trained models, we use the learning rate of $5e^{-5}$ and adopt the learning rate warmup and decay. The optimizer is Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We use dropout with a probability of 0.1. In the generation process, we use beam search with a beam size of 5. Trigram blocking is used to reduce repetitions. We adopt the implementations of BART, PEGASUS, T5, BigBird, and LED from HuggingFace's Transformers [138]. All the models are trained on one NVIDIA RTX 8000 GPU.

4.4.3 Evaluation Metrics

We propose a set of evaluation metrics to assess the usage of numerical information in produced summaries. This is necessary for long text and multi-table summarization. We use D, S, and H to denote the input document, human-written target summary, and the summarizer's output summary. D_n, S_n, and H_n are sets of numbers contained

Table 4.3: Automatic evaluation results on test sets of FINDSum-Liquidity.

Type	Method	R-1	R-2	R-L	NP	NC	NS
Only Text	LexRank	40.67	10.61	16.28	12.58	14.50	13.47
	TextRank	41.71	10.90	16.54	13.37	13.02	13.19
	BART	52.37	17.91	19.59	21.18	22.78	21.95
	PEGASUS	52.57	18.46	19.75	16.98	22.74	19.44
	LongT5	44.89	14.61	17.39	13.74	17.00	15.20
	LED	53.52	18.91	19.75	18.68	22.56	20.44
	BigBird-PEGASUS	53.42	19.39	20.07	17.16	22.44	19.45
Single Stage	GC-LED	52.30	20.09	19.61	15.13	**44.47**	22.58
	GC-BigBird	51.61	20.00	19.86	14.76	44.21	22.13
	CG-LED	54.12	20.26	20.46	21.86	35.14	**26.95**
	CG-BigBird	53.82	20.15	20.39	20.98	34.29	26.03
Two Stage	GTF-LED	53.88	19.82	20.13	21.37	31.76	25.55
	GTF-BigBird	53.66	19.56	19.97	**21.96**	30.52	25.54
	GCG-LED	**54.55**	20.36	20.41	21.15	34.52	26.23
	GCG-BigBird	53.90	**20.47**	**20.59**	20.67	36.43	26.38

in them. $|D_n|$, $|S_n|$, and $|H_n|$ denote the sizes of these number sets. For a produced summary H, we first extract the number set H_n from it.[4] Then $M(H_n, S_n)$ counts numbers appearing in both the produced summary H and the target summary S. $M(D_n, S_n)$ counts numbers appearing in both the input document D and the target summary S.

We mainly consider three metrics: Number Precision (NP), Number Coverage (NC), and Number Selection (NS). Calculated by Eq. (4.2), NP is the ratio of numbers in the

[4]We do not count numbers in a word, like COVID-19.

Table 4.4: Automatic evaluation results on test sets of FINDSum-ROO.

Type	Method	R-1	R-2	R-L	NP	NC	NS
Only Text	LexRank	34.43	7.73	14.92	14.77	9.73	11.73
	TextRank	35.93	7.74	15.08	14.68	10.96	12.55
	BART	49.00	16.88	19.14	14.38	23.72	17.91
	PEGASUS	51.92	19.31	21.47	10.90	21.89	14.55
	LongT5	43.26	11.84	17.83	8.75	10.37	9.49
	LED	53.06	20.33	22.28	14.25	22.99	17.59
	BigBird-PEGASUS	53.08	20.85	20.94	13.15	23.82	16.95
Single Stage	GC-LED	53.19	21.97	22.84	12.83	**41.54**	19.60
	GC-BigBird	53.13	22.03	23.11	12.49	41.30	19.18
	CG-LED	54.24	22.08	23.10	16.41	33.89	22.11
	CG-BigBird	**54.40**	**22.48**	**23.21**	**16.46**	35.84	**22.56**
Two Stage	GTF-LED	53.60	21.61	22.89	15.49	29.06	20.21
	GTF-BigBird	54.07	21.93	22.85	15.27	29.99	20.24
	GCG-LED	54.32	21.92	23.03	16.03	32.54	21.48
	GCG-BigBird	54.12	22.11	23.02	15.33	32.82	20.90

produced summary that also appears in the target summary. It measures how well the produced summary matches the target summary in terms of contained numbers. NC measures how well the produced summary covers the numbers appearing in both the target summary and the input document. Some of the numbers in the target summary cannot be directly found in the inputs (including textual and tabular data) and need numerical reasoning. Some of them may be lost when preparing the summarization model's inputs, which can limit the produced summary's number recall computed by Eq. (4.3a). To evaluate the summarization model's coverage capability, we divide the produced summary's number recall by the input document's number recall in Eq.

Table 4.5: GC methods' evaluation results on test sets of FINDSum-Liquidity. "Text/Tuple" denotes the assigned length ratio of text summary and table summary in each combined summary.

Text/ Tuple	**Method**	**R-1**	**R-2**	**R-L**	**NP**	**NC**	**NS**
1:1	GC-LED	52.30	20.09	19.61	15.13	**44.47**	**22.58**
	GC-BigBird	51.61	20.00	**19.86**	14.76	44.21	22.13
2:1	GC-LED	52.28	18.37	19.12	**16.63**	22.45	19.11
	GC-BigBird	52.99	**20.18**	19.81	14.43	35.62	20.54
3:1	GC-LED	52.57	18.47	19.21	16.13	22.24	18.70
	GC-BigBird	**53.33**	20.15	19.81	14.58	32.30	20.09

Table 4.6: GC methods' evaluation results on test sets of FINDSum-ROO. "Text/Tuple" denotes the assigned length ratio of text summary and table summary in each combined summary.

Text/ Tuple	**Method**	**R-1**	**R-2**	**R-L**	**NP**	**NC**	**NS**
1:1	GC-LED	53.19	21.97	22.84	12.83	**41.54**	19.60
	GC-BigBird	53.13	22.03	**23.11**	12.49	41.30	19.18
2:1	GC-LED	53.56	21.95	22.78	13.45	36.54	19.66
	GC-BigBird	53.51	22.02	22.69	12.82	38.74	19.26
3:1	GC-LED	**53.66**	21.88	22.48	**13.62**	36.21	**19.79**
	GC-BigBird	53.59	**22.07**	22.73	13.18	35.84	19.27

(4.3b). NS calculates the harmonic mean of NP and NC in Eq. (4.4) and reflects the quality of number selection in the produced summary.

Table 4.7: Human evaluation results on FINDSum-ROO. "Win" represents the generated summary of our method is better than that of BigBird-PEGASUS.

Method	Metric	Win	Lose	Tie	Kappa
CG-BigBird	Informativeness	44.2%	20.8%	35.0%	0.626
	Fluency	26.7%	25.8%	47.5%	0.616
	Non-Redundancy	35.0%	23.3%	41.7%	0.632
GTF-BigBird	Informativeness	42.5%	23.3%	34.2%	0.631
	Fluency	25.0%	26.7%	48.3%	0.648
	Non-Redundancy	34.2%	21.7%	44.2%	0.636
GCG-BigBird	Informativeness	43.3%	20.8%	35.8%	0.653
	Fluency	27.5%	24.2%	48.3%	0.613
	Non-Redundancy	33.3%	21.7%	45.0%	0.644

Table 4.8: Human evaluation results on FINDSum-Liquidity. "Win" represents the generated summary of our method is better than that of BigBird-PEGASUS.

Method	Metric	Win	Lose	Tie	Kappa
CG-BigBird	Informativeness	40.8%	20.8%	38.3%	0.620
	Fluency	25.0%	24.2%	50.8%	0.615
	Non-Redundancy	31.7%	22.5%	45.8%	0.626
GTF-BigBird	Informativeness	40.0%	22.5%	37.5%	0.649
	Fluency	24.2%	23.3%	52.5%	0.637
	Non-Redundancy	30.8%	24.2%	45.0%	0.629
GCG-BigBird	Informativeness	41.7%	21.6%	36.7%	0.655
	Fluency	25.8%	25.0%	49.2%	0.611
	Non-Redundancy	32.5%	23.3%	44.2%	0.638

Table 4.9: Evaluation results of input text selection methods on FINDSum-ROO. R-1 denotes the recall of unigram, and R-AVG is the average recall of unigram, bigram, trigram, and 5-gram.

Method	**Segment 1**		**Segment 2**	
	R-1	**R-AVG**	**R-1**	**R-AVG**
LexRank	56.01	22.14	53.96	20.72
TextRank	58.38	22.94	56.25	21.53
MMRG	**63.38**	**28.01**	**61.68**	**27.85**

$$\mathrm{NP}(H_n, S_n) = \frac{M(H_n, S_n)}{|H_n|} \tag{4.2}$$

$$\mathrm{NR}(H_n, S_n) = \frac{M(H_n, S_n)}{|S_n|} \tag{4.3a}$$

$$\mathrm{NC}(D_n, H_n, S_n) = \frac{\mathrm{NR}(H_n, S_n) * |S_n|}{M(D_n, S_n)} \tag{4.3b}$$

$$\mathrm{NS}(D_n, H_n, S_n) = \frac{2 * \mathrm{NP} * \mathrm{NC}}{\mathrm{NP} + \mathrm{NC}} \tag{4.4}$$

4.5 Results and Discussion

This section presents our experimental results and analysis. We conduct automatic and human evaluations to compare the quality of summaries produced by different models. We also conduct extensive comparative experiments to compare and analyze the influence of different components and configurations of summarization models. Finally, a case study compares and analyses different models' output summaries.

Table 4.10: Evaluation results of input text selection methods on FINDSum-Liquidity. R-1 denotes the recall of unigram, and R-AVG is the average recall of unigram, bigram, trigram, and 5-gram.

Method	Segment 1		Segment 2		Segment 3	
	R-1	**R-AVG**	**R-1**	**R-AVG**	**R-1**	**R-AVG**
LexRank	49.71	18.59	48.92	17.97	46.45	17.00
TextRank	55.18	20.94	54.02	20.40	51.72	19.49
MMRG	**58.61**	**24.28**	**56.69**	**23.09**	**53.94**	**21.62**

4.5.1 Summarization Results

In the automatic evaluation, we calculate the ROUGE F_1 scores [67], including the overlaps of unigrams (R-1), bigrams (R-2), and longest common subsequence (R-L)[5], and NP, NC, and NS scores. We employ a divide-and-conquer approach to generate summary segments in parallel and then merge them as the final summary. Tables 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.14, and 4.15 report the final merged summaries' scores. In Tables 4.3 and 4.4, all the abstractive methods are built on large pre-trained models. Limited by the GPU memory size, the input text length of models using full attention mechanism is 1024, while that of models with sparse attention mechanisms is 2048. The GC, CG, GTF, and GCG methods in Tables 4.3 and 4.4 receive selected tuples' linearized sequences of length 1024. Tables 4.3 and 4.4 show that these abstractive summarizers based on pre-trained models outperform unsupervised extractive summarizers. Compared with other baselines based on full attention, LED [147] and BigBird-PEGASUS [6] equipped with sparse attention can model longer context and achieve higher ROUGE scores. Longer inputs can cover more scattered salient content, which benefits output summaries' informativeness.

Our CG, GTF, and GCG methods outperform these text-only baselines on FIND-

[5]github.com/falcondai/pyrouge/

Table 4.11: Evaluation results of salient tuple selection on the Liquidity subset. "Pos" denotes positional features. "Glove" is the Glove embedding of row and column names. "ACC" and "Recall" are the accuracy and recall of the selected top-n tuples.

Method	**Features**	**Top-100**		**Top-200**		**Top-400**	
		ACC	**Rec**	**ACC**	**Rec**	**ACC**	**Rec**
LR	Pos	94.53	40.36	89.32	61.95	78.64	73.01
	Pos+Glove	94.64	52.96	89.36	66.84	78.70	79.69
SVM	Pos	94.55	43.19	89.34	64.27	78.63	72.24
	Pos+Glove	94.64	53.73	89.36	66.58	78.69	79.43
Adaboost	Pos	94.61	50.13	89.35	65.04	78.68	78.15
	Pos+Glove	94.69	58.87	89.42	73.78	78.74	85.35
XGBoost	Pos	94.61	49.61	89.38	69.15	78.70	79.95
	Pos+Glove	**94.74**	**65.30**	**89.46**	78.15	**78.77**	**88.43**
MLP	Pos	94.61	50.13	89.37	67.87	78.69	78.41
	Pos+Glove	**94.74**	**65.30**	**89.46**	**78.66**	78.76	87.40

Sum's two subsets. Incorporating tabular information is conducive to improving the NP, NC, NS, and ROUGE scores. GCG methods perform better on FINDSum-Liquidity, while CG methods perform better on FINDSum-ROO. Table 4.1 shows that target summaries in the FINDSum-ROO subset have a larger ratio of numerical information not found in the input text and rely more on tables. The table content passes one generation process in CG methods but needs to pass through two stages in GTF and GCG methods. The extra stage can lose some required tabular information and accumulate more errors. In FINDSum-Liquidity, a larger ratio of the numerical values can be found in the input text, and the loss of tabular information in the extra stage has less effect. Table 4.18 depicts that these tuple-to-text generators perform better on the Liquidity subset, which also contributes to the GCG methods' perfor-

Table 4.12: Evaluation results of salient tuple selection on the ROO subset. "Pos" denotes positional features. "Glove" is the Glove embedding of row and column names. "ACC" and "Recall" are the accuracy and recall of the selected top-n tuples.

Method	**Features**	**Top-100**		**Top-200**		**Top-400**	
		ACC	**Rec**	**ACC**	**Rec**	**ACC**	**Rec**
LR	Pos	94.54	41.53	89.27	56.08	78.60	68.78
	Pos+Glove	94.56	43.39	89.31	60.58	78.64	73.28
SVM	Pos	94.55	42.86	89.28	57.14	78.62	70.63
	Pos+Glove	94.56	43.65	89.31	60.58	78.65	74.34
Adaboost	Pos	94.57	45.24	89.31	60.05	78.62	70.90
	Pos+Glove	94.56	43.12	89.30	58.99	78.65	75.40
XGBoost	Pos	94.59	47.62	89.32	62.17	78.65	75.13
	Pos+Glove	94.63	52.65	**89.36**	**67.20**	**78.71**	**82.28**
MLP	Pos	94.60	48.41	89.32	61.90	78.65	74.60
	Pos+Glove	**94.64**	**53.97**	89.34	64.55	78.67	77.25

mance on the FINDSum-Liquidity. GTF methods' performance is mainly limited by the template generation process, which needs to decide whether to copy the numerical values appearing in the input text or the values in input tables and put placeholders in the exact positions. Meanwhile, better template filling methods can also benefit produced summaries' quality.

The GC methods do not perform well, which is due to GC's limitations mentioned in subsection 4.3.2. In Tables 4.3 and 4.4, the GC methods' evaluation results represent summaries combining text and table summaries of the same length. Although they can achieve high NC scores, their NP and ROUGE scores are unsatisfactory. The result reflects that it is not appropriate to treat long text and multi-table summarization

Table 4.13: Impact of text content selection methods on summarization results.

Summarizer	Text Select	R1/R2/RL
FINDSum-Liquidity		
LED-large	MMRG	**53.52/18.91/19.75**
	Textrank	51.76/17.35/19.02
	Lexrank	51.68/17.29/19.00
BigBird-PEGASUS	MMRG	**53.42/19.39/20.07**
	Textrank	51.09/17.29/19.20
	Lexrank	51.00/17.16/19.15
FINDSum-ROO		
LED-large	MMRG	**53.06/20.33/22.28**
	Textrank	48.75/16.07/19.78
	Lexrank	48.73/16.10/19.78
BigBird-PEGASUS	MMRG	**53.08/20.85/20.94**
	Textrank	49.59/17.30/20.67
	Lexrank	49.78/17.48/20.66

as two independent processes. Tables 4.5 and 4.6 show that the pre-defined length assignment can affect the combined summaries' quality. It is not flexible enough to adapt to diverse examples.

Our human evaluation compares different models' output summaries in terms of informativeness (the coverage of information from input documents), fluency (content organization and grammatical correctness), and non-redundancy (less repetitive information). We randomly selected 30 samples from the test sets of the FINDSum-ROO and FINDSum-Liquidity, respectively. Four annotators are required to compare two models' output summaries presented anonymously. We also assess their agreements by Fleiss' kappa [39]. Tables 4.7 and 4.8 exhibit that CG-BigBird, GTF-BigBird,

Table 4.14: Effect of input sequence length on generated results for FINDSum-Liquidity. "Input Len" denotes the length of input text and flattened tuples.

Input Len	Method	R-1	R-2	R-L	NP	NC	NS
2K+1K	LED-base	52.91	18.41	19.58	19.97	24.07	21.83
	CG-LED-base	53.73	19.68	20.26	21.76	34.39	26.65
	GTF-LED-base	53.89	19.39	19.88	20.77	30.73	24.79
	GCG-LED-base	54.01	19.94	20.26	20.29	35.32	25.77
4K+2K	LED-base	53.58	19.29	20.01	19.68	24.63	21.88
	CG-LED-base	54.42	20.40	20.44	22.32	38.10	28.15
	GTF-LED-base	54.02	19.83	20.10	21.75	30.70	25.46
	GCG-LED-base	54.14	20.11	20.32	21.23	36.78	26.92
8K+4K	LED-base	54.04	19.88	20.31	20.40	26.35	23.00
	CG-LED-base	**54.82**	**20.95**	**20.78**	**24.49**	**41.36**	**30.76**
	GTF-LED-base	54.61	20.67	20.55	23.05	32.27	26.89
	GCG-LED-base	54.60	20.71	20.57	22.53	38.26	28.36

and GCG-BigBird significantly outperform the BigBird-PEGASUS only using input text in terms of informativeness and are comparable in terms of fluency and non-redundancy. It verifies that incorporating text and tabular data into summary generation can benefit output summaries' informativeness.

The following subsections discuss our extensive comparative experiments comparing the performance of different model components (e.g., content selection methods, divide-and-conquer method, sparse attention mechanisms, template filling methods in GTF, and tuple-to-text generators in GCG). We also analyze their influence on summarization results.

Table 4.15: Effect of input sequence length on generated results for FINDSum-ROO. "Input Len" denotes the length of input text and flattened tuples.

Input Len	Method	R-1	R-2	R-L	NP	NC	NS
2K+1K	LED-base	52.96	20.52	22.14	14.60	24.67	18.34
	CG-LED-base	54.38	22.13	23.10	17.21	34.94	23.06
	GTF-LED-base	53.76	21.54	22.81	14.69	29.07	19.52
	GCG-LED-base	53.91	21.69	22.83	15.90	30.23	20.84
4K+2K	LED-base	53.68	21.64	22.95	15.66	26.10	19.58
	CG-LED-base	54.51	22.63	23.42	17.55	35.06	23.39
	GTF-LED-base	54.16	22.31	23.27	16.11	30.37	21.05
	GCG-LED-base	**54.53**	**22.63**	23.45	16.07	32.71	21.55
8K+4K	LED-base	53.39	21.53	22.97	15.91	26.94	20.01
	CG-LED-base	54.43	22.59	**23.46**	**18.24**	**35.69**	**24.14**
	GTF-LED-base	53.92	21.96	23.11	16.61	30.09	21.40
	GCG-LED-base	54.09	22.23	23.21	15.98	32.58	21.44

4.5.2 Discussion on Content Selection Methods

As introduced in subsection 4.3.1, the content selection step filters out the non-prominent content and retains the salient content as summarizers' inputs. Different methods are employed to select salient content from text and tabular data, considering their different natures. We conduct a series of experiments to compare the performance of different content selection methods and their impact on summarization results.

To select salient text content, we compare the statistics-based MMRG method with some extractive summarization methods, like TextRank and Lexrank. We use the

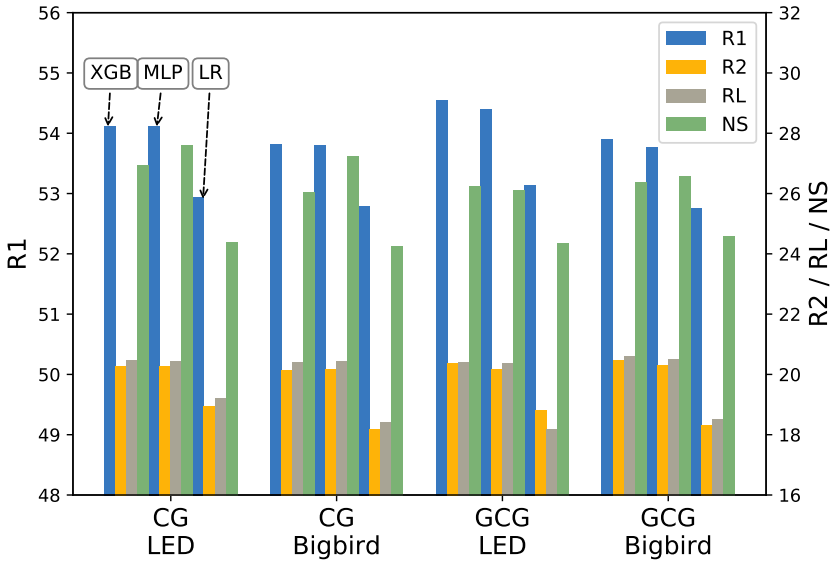


Figure 4.4: Impact of tuple selection methods on FINDSum-Liquidity. Each summarization method using outputs of XGBoost, MLP, and LR has three parts of scores.

recall of n-grams to evaluate these methods' performance in selecting the salient text of the same length. Tables 4.10 and 4.9 indicate that MMRG outperforms these extractive summarizers. We also compare their impact on the ROUGE scores of produced summaries. Table 4.13 depicts that the better text content selection method benefits the quality of produced summaries.

As for those thousands of tuples extracted from tables, we model the salient tuple selection as a binary classification task. We annotate a tuple selection dataset based on the FINDSum dataset. Salient tuples (positive samples) are usually sparse in these report documents. To deal with the class imbalance problem, we perform undersampling over negative samples to ensure the ratio of positive and negative samples is 1:10 in the training set. We train and evaluate different classification methods, including the logistic regression (LR), support vector machine (SVM), AdaBoost, XGBoost, and Multi-layer Perceptron (MLP), on our annotated tuple selection dataset. We use the accuracy and recall of salient tuples to evaluate these classifiers. Tables 4.11 and 4.12 show that adding word embeddings of row and column names as features

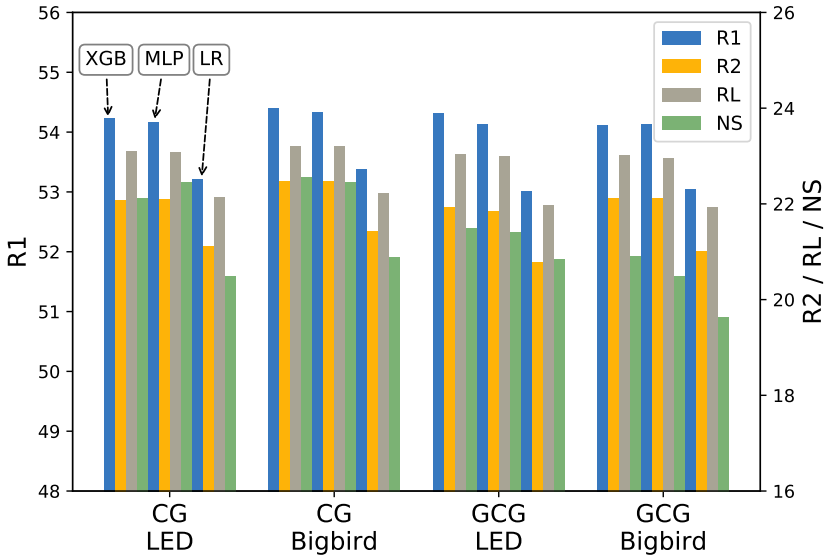


Figure 4.5: Impact of tuple selection methods on FINDSum-ROO. Each summarization method using outputs of XGBoost, MLP, and LR has three parts of scores.

significantly improves the recall of all classifiers and facilitates finding salient table content. Besides, XGBoost and MLP models equipped with positional features and Glove embedding [99] outperform other classifiers. We compare three tuple selection methods' impact on the quality of produced summaries. Fig. 4.4 and Fig. 4.5 depict that summarization models receiving the outputs of XGBoost and MLP outperform summarization models using the LR model's outputs. Better tuple selection methods benefit the quality of produced summaries. This verifies the effectiveness of our tuple selection methods.

4.5.3 Discussion on Input Length of Summarization Model

How to set the input sequence length of the summarization model is also an important issue. We conduct a series of experiments to analyze the input sequence length's impact on the performance of summarization models. Considering the constraint of GPU memory size, we use the base model of LED as the backbone. We compare the

performance of text-only, CG, GTF, and GCG methods over the inputs of different lengths. Tables 4.14 and 4.15 show that these methods built on the base model with longer inputs can surpass these methods based on the large model shown in Tables 4.3 and 4.4. When the GPU memory size is limited, using the base model with longer inputs may be better. However, increasing input length does not always bring performance gain. As shown in Fig. 4.6 and Fig. 4.7, all of these summarization models improve when the input length is doubled. When the input length increases from double to quadruple, models' performance on the Liquidity subset improves, while some models' ROUGE scores on the ROO subset decrease. Longer inputs can cover more salient information, which benefits generating informative summaries. Meanwhile, longer inputs can also introduce more non-prominent content, making it more difficult for summarization models to identify salient content. When adjusting the input length, we should find a balance between covering salient information and reducing non-prominent content to meet the requirements of various outputs.

4.5.4 Discussion on the Divide-and-Conquer Method

Existing autoregressive abstractive summarization methods still have difficulty in generating long text in terms of efficiency and quality [54, 105]. We adopt a divide-and-conquer (DC) method [42], which decomposes the long summary generation problem into multiple sub-problems of summary segment generation. These summary segments can be generated in parallel and merged as a final summary. We conduct experiments comparing the performance of these summarization models with and without the divide-and-conquer method. Fig. 4.6 and Fig. 4.7 show that DC can bring additional performance gains even when the context length grows. When the GPU memory size limits the model's context length, DC can help the model produce better summaries with relatively short inputs. It reveals the effectiveness of the divide-and-conquer method. To reduce the redundancy in the merged summary, we add constraints to MMRG to avoid providing the same inputs for summarizers and

Table 4.16: Evaluation results of template filling. EM denotes Exact Match.

Dataset	Type	Method	EM
ROO	Gen	BART-base	71.71
		BART-large	75.40
		TAPEX-base	74.75
		TAPEX-large	76.82
	Ext	Bigbird-base	76.75
		Bigbird-large	**80.72**
		Longformer-base	77.88
		Longformer-large	80.30
Liquidity	Gen	BART-base	73.13
		BART-large	70.96
		TAPEX-base	75.49
		TAPEX-large	74.59
	Ext	Bigbird-base	77.59
		Bigbird-large	78.12
		Longformer-base	78.31
		Longformer-large	**79.99**

use trigram blocking in the summary generation process, as discussed in subsections 4.3.3 and 4.4.2. Besides, we train a separate model to generate each summary segment. Different segments of target summaries can supervise separate summarization models to focus on different content.

Table 4.17: Impact of template filling methods. TG and TF denote the template generation and template filling methods. LF is the Longformer model.

TG	TF	R1/R2/RL	NP/NC/NS
FINDSum-ROO			
BigBird-PEGASUS	Bigbird-base	53.90/21.81/22.76	14.16/27.7/18.74
	Bigbird-large	**54.07**/**21.93**/22.85	15.27/**29.99**/**20.24**
LED	LF-base	53.51/21.50/22.78	14.08/28.48/18.84
	LF-large	53.60/21.61/**22.89**	**15.49**/29.06/20.21
FINDSum-Liquidity			
BigBird-PEGASUS	Bigbird-base	53.52/19.45/19.86	21.14/30.16/24.86
	Bigbird-large	53.66/19.56/19.97	**21.96**/30.52/25.54
LED	LF-base	53.75/19.71/20.03	20.64/31.04/24.79
	LF-large	**53.88**/**19.82**/**20.13**	21.37/**31.76**/**25.55**

4.5.5 Discussion on Template Filling Methods

Template filling is the second stage in GTF methods introduced in subsection 4.3.2. We model the template filling process as a question answering (QA) task. We use each template sentence containing the placeholder as a question and the linearized sequence of selected tuples as the context. We annotate a question answering dataset based on FINDSum and employ extractive or generative QA models to find numerical values from table content as answers to replace placeholders in questions. Considering the requirements of template filling, we use exact match (EM) to evaluate template filling methods. Table 4.16 shows that the Bigbird-large outperforms other baselines on the ROO subset, while the Longformer-large performs the best on the Liquidity subset. Besides, these extractive methods perform better than generative methods. We find that generative methods still suffer from hallucinations and can generate inaccurate numerical values. Compared with the backbone BART model [64], pre-

Table 4.18: Evaluation results of tuple-to-text generation.

Dataset	Method	R-1	R-2	R-L	BLEU
ROO	T5-base	45.45	24.77	28.84	12.20
	T5-large	45.81	24.64	29.04	12.87
	BART-base	42.08	20.45	25.86	10.57
	BART-large	**47.21**	**25.63**	**31.08**	**13.14**
Liquidity	T5-base	48.90	**28.34**	31.98	15.44
	T5-large	49.03	28.05	32.02	15.86
	BART-base	45.71	24.75	29.28	13.66
	BART-large	**49.78**	28.24	**32.59**	**16.05**

training on table-related tasks brings performance gain to the TAPEX model [71]. Table 4.17 compares the impact of different template filling methods on the quality of generated summaries when using the same template generation method. As shown in Table 4.16, the large models of Bigbird and Longformer perform better than their base models in template filling. These better template filling methods benefit the NP, NC, and NS scores of produced summaries. Presenting accurate numerical values is essential for financial reports' summaries. Template filling methods have less impact on ROUGE scores because there are many fewer numerical values than words in target summaries.

4.5.6 Discussion on Tuple-to-Text Generation Methods

As introduced in subsection 4.3.2, tuple-to-text generation is the first step in GCG methods. We annotate a tuple-to-text generation dataset based on our FINDSum dataset for training and evaluating various generators. These tuple-to-text generators

Table 4.19: N-gram recall of tuple-to-text generation results on test sets of FINDSum-ROO and FINDSum-Liquidity.

	Top-N	R-1	R-2	R-3	R-5	R-AVG
ROO						
XGB	100	36.32	16.55	7.45	1.47	15.45
	200	42.73	20.67	9.78	2.01	18.80
	400	49.25	**24.88**	**12.19**	**2.59**	22.23
MLP	100	36.75	16.69	7.51	1.49	15.61
	200	42.98	20.69	9.74	1.97	18.85
	400	**49.38**	24.86	12.16	2.57	**22.24**
Liquidity						
XGB	100	33.47	15.27	7.12	1.69	14.39
	200	40.11	18.99	9.26	2.13	17.62
	400	46.84	22.78	11.51	2.67	20.95
MLP	100	33.65	15.39	7.17	1.68	14.47
	200	40.30	19.10	9.31	2.15	17.72
	400	**47.06**	**22.87**	**11.55**	**2.70**	**21.05**

are evaluated by the ROUGE [67] and BLEU scores[6] [95]. Table 4.18 depicts the performance of different tuple-to-text generators on ROO and Liquidity subsets. The large model of BART [64] performs the best on these two subsets, so we use it as the tuple-to-text generator in GCG. Table 4.19 depicts that both the tuple selection methods in the content selection step and the number of input tuples can affect tuple-to-text generators' performance.

[6]www.nltk.org/api/nltk.translate.bleu_score.html. We report the cumulative 4-gram BLEU score.

Table 4.20: Details of summarization models.

Model	Architecture	Params	Enc/Dec Layers	Heads	d_{model}	d_{ff}	Input Len
Bigbird_{base}	Enc	127.5M	12	12	768	3,072	4,096
Bigbird_{large}	Enc	359.1M	24	16	1,024	4,096	4,096
Longformer_{base}	Enc	148.7M	12	12	768	3,072	4,098
$\text{Longformer}_{large}$	Enc	434.6M	24	16	1,024	4,096	4,098
BART_{base}	Enc-Dec	139.4M	6	12	768	3,072	1,024
BART_{large}	Enc-Dec	406.3M	12	16	1,024	4,096	1,024
PEGASUS_{large}	Enc-Dec	570.8M	16	16	1,024	4,096	1,024
LED_{base}	Enc-Dec	161.8M	6	12	768	3,072	16,384
$\text{LED}_{\text{large}}$	Enc-Dec	459.8M	12	16	1,024	4,096	16,384
BigBird-PEGASUS	Enc-Dec	577.1M	16	16	1,024	4,096	4,096
T5_{base}	Enc-Dec	222.9M	12	12	768	3,072	512
T5_{large}	Enc-Dec	737.7M	24	16	1,024	4,096	512
$\text{LongT5}_{\text{large}}$	Enc-Dec	783.2M	24	16	1,024	2,816	4,096
TAPEX_{base}	Enc-Dec	139.4M	6	12	768	3,072	1,024
TAPEX_{large}	Enc-Dec	406.3M	12	16	1,024	4,096	1,024

4.5.7 Case Study

We conduct a case study to compare and analyze summaries generated by different models. Fig. 4.8 has two parts. Its left part shows some fragments of input text and tables from one example in the FINDSum-Liquidity. The right part presents fragments in the target summary and different models' output summaries. When comparing these summaries, we find that our GCG, CG, and GTF methods can generate quantitative descriptions of some critical items in tables. The text-only baseline BigBird-PEGASUS focuses more on narratives in the input text. Without tabular data as evidence, most of the numerical values generated by the BigBird-PEGASUS are inaccurate. It reflects the importance of incorporating tables when

summarizing report documents.

GCG method's input is the concatenation of input text and generated text descriptions of selected tabular data, which differs from the CG method. The summary generated by GCG focuses more on descriptions of tables. Unlike the GCG method, the CG method needs to handle text-to-text and tuple-to-text generation simultaneously, which is quite challenging. The generated summary reflects that the CG method can find a balance for its focus on text and table content. The accuracy of its tuple-to-text generation needs further improvements. As for the GTF method, it enumerates many critical items in its generated summary, but it does not mention these items' values. As discussed in subsection 4.5.1, the GTF method's performance is mainly limited by the template generation process. If the generated template does not add or add placeholders in the wrong positions, the template filling step cannot produce quantitative descriptions correctly.

Some items mentioned in the target summary need numerical reasoning over tabular data. For example, the item "changes in our operating assets and liabilities" has many components. Although its value is not shown in the table, we can calculate it by adding up all its components. Some items like "non-cash charges" do not appear in inputs. To handle these more complex cases, the summarization model needs more knowledge about the relationships among all these items and better numerical reasoning ability.

4.6 Chapter Summary

In this chapter, we propose a new task named long text and multi-table summarization, which generalizes the long document summarization from unimodal (text) summarization to multimodal. Previous document summarization datasets and methods are usually restricted to summarize the text content and exclude tables and figures

from the input. In financial report documents, the key information can be distributed across both textual and non-textual content. The absence of tabular data can restrict the informativeness of generated summaries, particularly when summaries necessitate the quantitative descriptions of vital metrics within tables. Existing summarization methods and datasets fail to meet the demands of summarizing extensive textual and tabular content within financial reports. To deal with the scarcity of available datasets, we propose FINDSum, the first large-scale dataset for long text and multi-table summarization. Besides, we present four types of summarization methods to jointly consider the text and table content when summarizing reports. Additionally, we propose a set of evaluation metrics assessing the utilization of numerical information within the generated summaries.

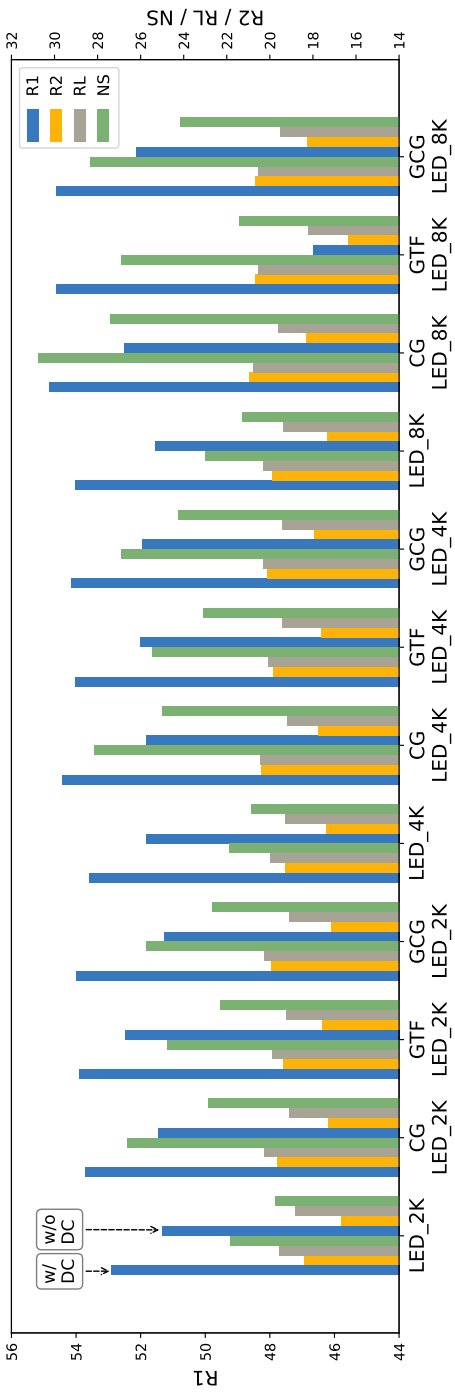


Figure 4.6: Impact of input length and Divide-and-Conquer (DC) on FINDSum-Liquidity. Each summarizer has two parts of scores denoting w/ and w/o DC.

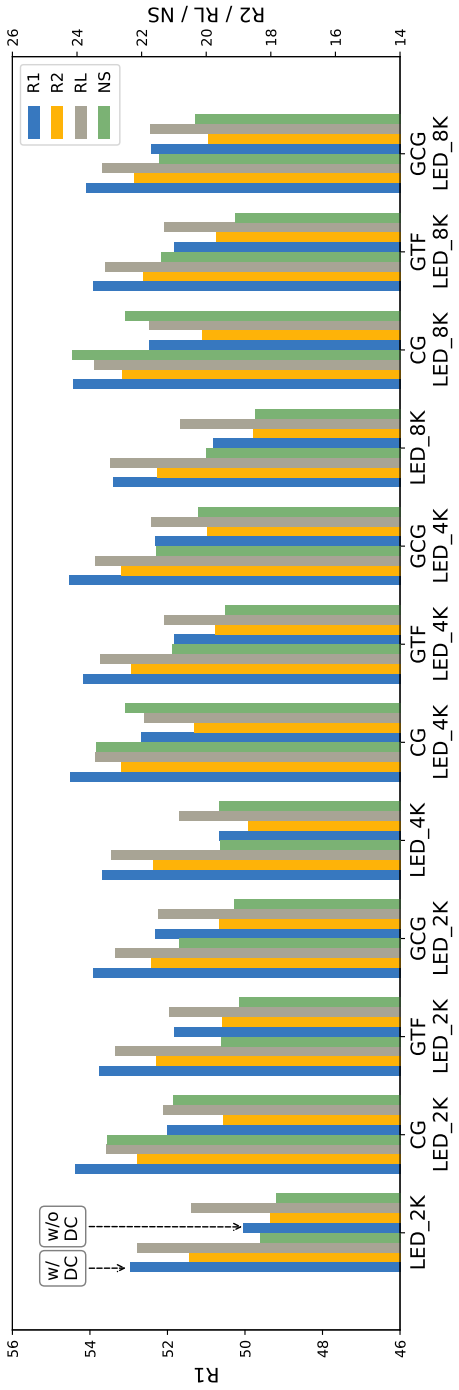


Figure 4.7: Impact of input length and Divide-and-Conquer (DC) on FINDSum-ROO. Each summarizer has two parts of scores denoting w/ and w/o DC.

Input

Table 1: Statements of Cash Flows (In thousands)

	2019
Net loss	(94,433)
Adjustments to reconcile net loss to net cash used in operating activities:	
Depreciation and amortization	4,745
Net amortization of premiums and discounts on marketable securities	(1,349)
Stock-based compensation	5,299
Non-cash operating lease expense	6,382
Changes in operating assets and liabilities:	
…	…
Deferred revenue	(4,297)
…	…
Purchase of marketable securities	(80,979)
Maturities of marketable securities	112,993
Purchase of property and equipment	(16,173)
…	…

Table 2: (In thousands)

	2019
Cash used in operating activities	(85,011)
Cash provided by (used in) investing activities	15,841
…	…

…

Input Text:

…
we have funded our operations to date primarily from private placements of our convertible preferred stock , the net proceeds from our initial public offering … we have incurred net losses each year since inception. our net losses were $ 94.4 million, $ 64.8 million and $ 41.4 million for the years ended december 31, 2019, 2018 and 2017, respectively. as of december 31, 2019, we had an accumulated deficit of $ 221.0 million … since our inception, we have incurred significant losses and negative cash flows from operations . we have an accumulated deficit of $ 221.0 million through december 31, 2019. we expect to incur substantial additional losses in the future as we conduct and expand our research and development activities. …

Output Summary

Target Summary: cash used in operating activities was $ 85.0 million , which consisted of a net loss of $ 94.4 million , adjusted by non-cash charges of $ 15.1 million and cash used due to changes in our operating assets and liabilities of $ 5.7 million. the non-cash charges consisted primarily of depreciation and amortization expense of $ 3.4 million , stock-based compensation of $ 5.3 million, and non-cash operating lease expense of $ 6.4 million. … during the year ended december 31 , 2019 , cash provided by investing activities was $ 15.8 million , which consisted of $ 113.0 million in proceeds from the maturity of marketable securities , offset by $ 81.0 million of purchases of marketable securities and $ 16.2 million of capital expenditures to purchase property and equipment. …

GCG-Bigbird: operating activities net cash used in operating activities was $ 85.0 million for the year ended december 31, 2019, primarily resulting from our net loss of $ 94.4 million, which was partially offset by non-cash charges of $ 50.0 million for depreciation and amortization, $ 0.7 million for stock-based compensation, and $ 6.4 million for operating lease expense. … net cash provided by investing activities was $ 15.8 million during the periods presented, primarily due to maturities of marketable securities of $ 64.3 million and purchases of property and equipment of $ 5.7 million offset by purchases of available-for-sale marketable securities totaling $ 80.0 million …

CG-Bigbird: since our inception, we have incurred significant losses and negative cash flows from operations. we have an accumulated deficit of $ 221.0 million through december 31, 2019. we expect to incur substantial additional losses in the future as we conduct and expand our research and development activities and invest in our manufacturing facility. … operating activities net cash used in operating activities was $ 81.0 million for the year ended december 31, 2019, and consisted primarily of a net loss of $ 94.4 million, partially offset by non-cash charges of $ 55.0 million and net cash provided by changes in our operating assets and liabilities of $ 15.8 million. the net loss was primarily due to an increase in the net deferred revenue of $ 4.3 million due to the timing of payments …
$ 5.3 million in stock-based compensation expense and $ 6.4 million in operating lease expense … net cash from investing activities for 2019 was $ 16.4 million and consisted primarily of maturities and purchases of marketable securities of $ 48.7 million …

GTF-Bigbird: since our inception, we have incurred significant losses and negative cash flows from operations. we have funded our operations to date primarily from private placements of our convertible preferred stock, the net proceeds from our initial public offering … net cash used in operating activities for the year ended december 31, 2019 was $ 85 million and consisted primarily of our net loss of $ 94.4 million, partially offset by non-cash charges for stock-based compensation, depreciation and amortization, operating lease expense, net amortization of premiums and discounts on marketable securities and net changes in our operating assets and liabilities. … our investing activities consist primarily of purchases and maturities of marketable securities, purchases and sales of property and equipment, and purchases of intangible assets. …

BigBird-PEGASUS: since our inception, we have incurred significant losses and negative cash flows from operations. we have funded our operations to date primarily from private placements of our convertible preferred stock, the net proceeds from our initial public offering, … net cash used in operating activities was $ 64.6 million, primarily resulting from our net loss of $ 92.2 million and changes in our operating assets and liabilities, partially offset by non-cash charges totaling $ 19.9 million. … net cash from investing activities was $ (20.0) million for the years ended december 30, 2019 and 2018, respectively. during the year, we purchased marketable securities totaling $ 20.0 million and purchased property and equipment, net of cash acquired, of $ 0.2 million and $ 0.1 million respectively, which were offset by maturities of marketable securities of $ 25.0 million …

Figure 4.8: The input content and output summaries of an example from the FINDSum-Liquidity. In these output summaries, the underlined content comes from row names or cell values of input tables or input text fragments. The summary sentences marked with dotted lines below are mainly derived from the input text, while those marked with solid lines below mainly come from the input tables.

Chapter 5

From Single Document to Multiple Documents: Generating a Structured Summary of Numerous Academic Papers

5.1 Introduction

The number of published academic papers has been growing rapidly [2]. It becomes more and more difficult for researchers to read through the numerous papers on the research topics they are interested in. A summary of papers on a research topic can help researchers quickly browse key information in these papers. As a type of human-written summary, the survey paper can review numerous papers on each research topic and guide people to learn the topic. However, writing a survey paper needs a lot of time and effort, making it challenging to cover the latest papers and all the research topics. The multi-document summarization (MDS) techniques [38, 70, 74] can be utilized to automatically produce summaries as a supplement to human-

written summaries. To cover the latest papers and more research topics at a low cost, people can flexibly adjust the input papers and let the summarization methods produce summaries for these papers. Our target is to generate a comprehensive, well-organized, and non-redundant summary for numerous papers on the same research topic. To achieve this target, there are some challenging issues, including the scarcity of available data, the organization of diverse content from different sources, and the summarization models' efficiency in processing long texts.

Although there have been some MDS datasets [38, 81], most of them focus on producing short and structureless summaries covering less than ten input documents, which cannot meet the real needs of reviewing numerous papers on one research topic. To deal with the scarcity of available data, we propose BigSurvey, the first large-scale dataset for numerous academic papers summarization. It contains more than seven thousand survey papers and their 434 thousand reference papers' abstracts. Considering copyright issues, we collect these reference papers' abstracts as input documents for MDS. These abstracts can be regarded as summaries written by their authors, which include these reference papers' salient information.

These input abstracts usually have content on multiple aspects, including the background, method, objective, and results. It is challenging for a summary to organize and present the diverse content from dozens of input documents. Compared with the structureless summary, the structured summary contains multiple sections summarizing particular aspects of input content and is found easier to read and more welcomed by readers [48, 49]. To balance comprehensiveness and brevity, we built two subsets of the BigSurvey to produce two-level summaries. The BigSurvey-MDS focuses on producing comprehensive summaries, while the BigSurvey-Abs is built to produce more concise summaries of these summaries in BigSurvey-MDS.

We make two assumptions for the structured summary of multiple papers on the same topic. 1) The research topic's descriptions on one aspect constitute a subset of the union of related papers' content on this aspect (e.g., the research topic's back-

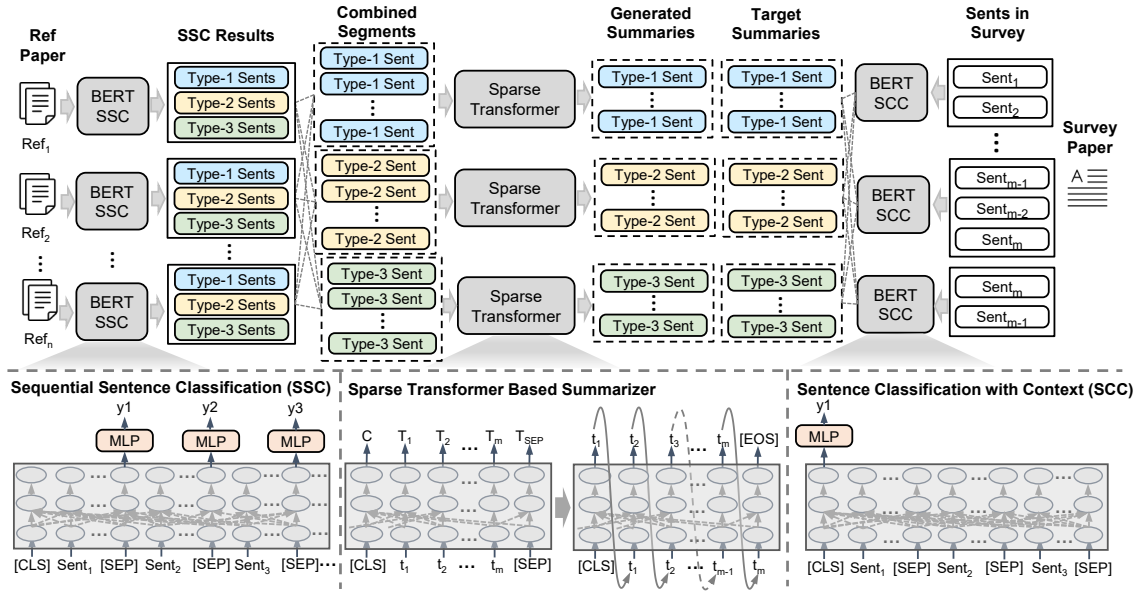


Figure 5.1: An overview of our CAST method.

ground should be part of all the related papers' backgrounds). 2) Each section of the structured summary focuses more on the salient content in one subset mentioned in 1) (e.g., the summary's background section focuses on the salient content in all the reference papers' backgrounds). Based on these assumptions, we propose category-based alignment (CA) to align each section of the structured summary with a set of input sentences classified as the same type.

As shown in Table 5.1, the average sum of input documents' word numbers is close to twelve thousand in each example of the BigSurvey dataset. The much longer inputs can introduce more noises, and the salient content can be more scattered, which makes it more difficult to capture and encode the salient content. Long input sequences can also reduce the efficiency of summarization models since existing neural models' time or space complexity is usually highly correlated with the input sequence length. To deal with the above problems, we propose a method named category-based alignment and sparse transformer (CAST). As shown in Fig. 5.1, we use the BERT-based sequential sentence classification (SSC) method and the sentence classification

with context (SCC) method to classify input and output sentences. Then, we use the category-based alignment to align the sets of input and target output sentences classified as the same type and compose examples for training summarization models. We adopt the transformer with the sparse attention mechanism for abstractive summarization. The sparse attention supports the encoder in modeling longer input sequences with limited GPU memory. Our BigSurvey dataset and CAST method enable fine-tuning a large pre-trained model to generate structured summaries covering dozens of input documents on an off-the-shelf GPU.

We benchmark advanced extractive and abstractive summarization methods as baselines on our BigSurvey dataset. To compare their performance, we conduct automatic evaluation and human evaluation. Experimental results show that our proposed CAST method outperforms these baseline models, and adding the category-based alignment can bring extra performance gains for various summarization methods.

Our contribution is threefold:

- We build BigSurvey, the first large-scale dataset for numerous academic papers summarization.

- We propose a method named category-based alignment and sparse transformer (CAST) to summarize numerous academic papers on each research topic.

- We benchmark various summarization methods on our dataset and find that adding the category-based alignment can bring extra performance gains for various methods.

Table 5.1: Comparison of our BigSurvey dataset to other summarization datasets. "Pairs" denotes the number of examples. "Words" and "Sents" indicate the average number of words and sentences in input text or target summary. "Doc Num" represents the average number of input documents in each example. "Cov." is the extractive fragment coverage, "Dens." is the extractive fragment density, and "Comp." is the compression ratio of target summaries.

Dataset	**Pairs**	**Words (Doc)**	**Sents (Doc)**	**Words (Sum)**	**Sents (Sum)**	**Doc Num**	**Cov.**	**Dens.**	**Comp.**
Multi-News	56.2k	2,103.5	82.7	263.7	10.0	2.8	0.69	3.1	6.3
Multi-XScience	40.5k	778.1	23.7	116.4	4.9	4.4	0.60	1.1	5.6
PubMed	133.2k	3,049.0	87.5	202.4	6.8	1	0.79	4.3	13.6
ArXiv	215.9k	6,029.9	205.7	272.7	9.6	1	0.87	3.8	39.8
BigSurvey-MDS	4.4k	11,893.1	450.1	1,051.7	38.8	**76.3**	0.81	1.5	11.3
BigSurvey-Abs	7.1k	12,174.5	463.8	170.1	6.4	1	0.83	3.5	71.6

5.2 BigSurvey Dataset

In this section, we first present our data sources and procedures of data collection and pre-processing. And then, we introduce our BigSurvey dataset[1]. We also conduct descriptive statistics and in-depth analysis of our dataset and compare them with other commonly used document summarization datasets.

[1]Our dataset: https://github.com/StevenLau6/BigSurvey

5.2.1 Data Collection and Pre-processing

We collect more than seven thousand survey papers from arXiv.org [2], download their PDF files by their DOIs, and parse these files with a tool named science-parse[3]. We can extract the bibliography information (e.g., reference papers' titles and authors) from parsing results. Based on these survey papers' bibliography information, we collect their reference papers' abstracts from Microsoft Academic Service (MAS) [121] and Semantic Scholar [1]. We collected more than 434 thousand reference papers in total.

In the pre-processing stage, we first filter out invalid samples from collected data. Specifically, downloaded files that are duplicated or cannot be parsed properly (e.g., some PDF files are scanned or incomplete) are removed. We also filter out outliers with too-short parsed texts in the survey papers or very few collected reference papers. For each selected survey paper, we remove noises (e.g., the copyright information before the first section and special symbols used to compose a style), extract the abstract and introduction sections from these survey papers, and truncate their reference papers' abstracts. We lowercase these texts and use NLTK [10] to split sentences and words. After that, we split the training (80%), validation (10%), and test (10%) sets.

5.2.2 Dataset Description

BigSurvey is a large-scale dataset containing two-level target summaries for dozens of academic papers on the same topic. The long summary aims to comprehensively cover the reference papers' salient content in different aspects, while the much shorter summary is more concise and can be regarded as the summary of the long summary.

[2]These survey papers' metadata are collected from a June 2021 dump (https://www.kaggle.com/datasets/Cornell-University/arxiv)

[3]https://github.com/allenai/science-parse

For these two-level summaries, we build two subsets: BigSurvey-MDS and BigSurvey-Abs. Their statistical information is shown in Table 5.1. We will introduce their definitions and properties separately.

BigSurvey-MDS. This subset focuses on producing comprehensive summaries covering numerous academic papers on one research topic. Each example in the BigSurvey-MDS corresponds to one survey paper from arXiv.org. These survey papers usually have tens or hundreds of reference papers. Considering copyright issues, BigSurvey-MDS does not include these reference papers' body sections and uses their abstracts as input documents. These abstracts can be regarded as summaries written by their authors, which include these papers' salient information. For each survey paper, we collect at most two hundred reference papers' abstracts and truncate each of them to no more than two hundred words. These truncated abstracts are used as input documents of the BigSurvey-MDS.

The survey paper's introduction section usually introduces a research topic's background, method, and other aspects. We split the content of the survey paper's introduction into three sections (the background, method, and other) and use them to compose the structured summary as the target in each example of the BigSurvey-MDS. The content about the objective, result, and others are merged into the section named others because we observe that these types of content appear less frequently than the background and method in the survey papers' introduction section. To prepare these three sections in the target summary, we first collect the introduction section from a survey paper. If there is no introduction section, we extract the survey paper's first 1,024 words after the abstract part. Then, we classify sentences in the introduction section and concatenate the sentences classified as the same type to form the three sections in the target summary. We filter out the examples with too short input sequences or target summaries. As shown in Table 5.1, BigSurvey-MDS's average input length, average output length, and average number of input documents

Table 5.2: Coverage and density distributions of the BigSurvey.

Dataset	% of novel n-grams in target summary			
	unigrams	bigrams	trigrams	4-grams
Multi-News	17.76	57.10	75.71	82.30
Multi-XScience	42.33	81.75	94.57	97.62
PubMed	18.38	49.97	69.21	78.42
ArXiv	15.04	48.21	71.66	83.26
BigSurvey-MDS	37.39	76.46	93.87	98.04
BigSurvey-Abs	19.85	53.97	74.15	82.22

are much larger than previous MDS datasets.

BigSurvey-Abs. The body text of a survey paper can be regarded as a comprehensive and long summary of its reference papers. Meanwhile, the survey paper's abstract is a short summary of its body text. The subset named BigSurvey-Abs uses these survey papers' abstracts as target summaries, which aims to produce more concise summaries of these survey papers' body text. Considering the constraints of GPU memory, we truncate these survey papers in our experiments. Specifically, we follow the settings in [147, 149] to use the first 1,024 words as the input for transformer-based models without sparse attention and use the first 3,072 words as the input for transformer-based models with sparse attention. In this case, the input documents of the BigSurvey-Abs highly overlap with the target summaries in the BigSurvey-MDS. Therefore, the short summary in the BigSurvey-Abs can be regarded as the summary of the long summary in the BigSurvey-MDS. Besides, the average input and output lengths are similar to previous academic literature summarization datasets. Previous text summarization methods should be able to adapt to the BigSurvey-Abs dataset.

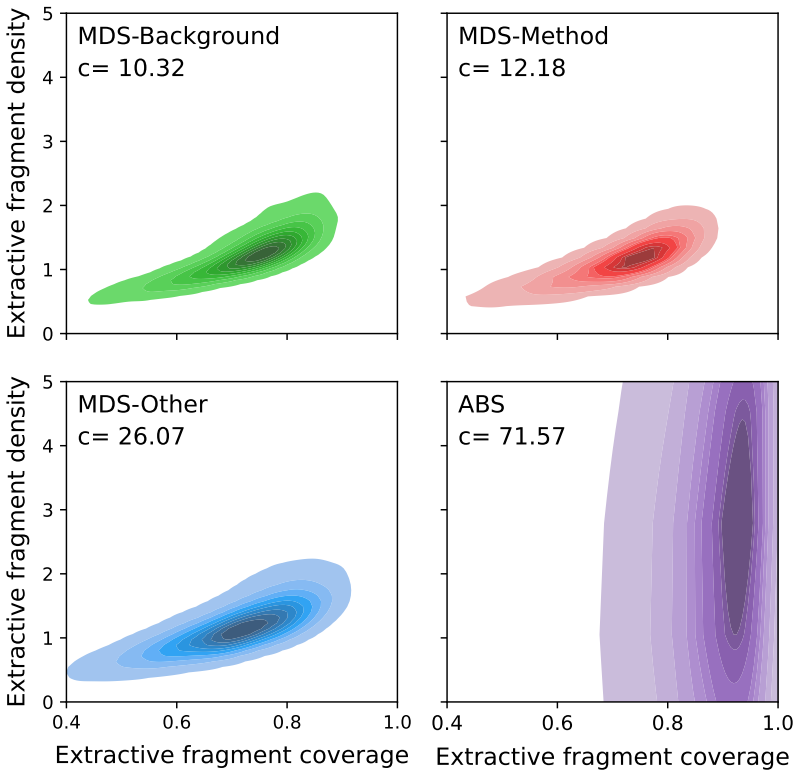


Figure 5.2: Coverage and density distributions of the BigSurvey.

5.2.3 Diversity Analysis of Dataset

To measure how abstractive our target summaries are, we report the percentage of target summaries' novel n-grams, which do not appear in input documents. Table 5.2 reflects that the abstractiveness of the BigSurvey-MDS subset is similar to that of the Multi-XScience. The BigSurvey-Abs subset's abstractiveness is lower than that of the BigSurvey-MDS and Multi-XScience and is similar to other existing datasets.

Besides, we assess the extractive nature of BigSurvey's subsets by three measures defined by Grusky et al. [44], including the extractive fragment coverage, extractive fragment density, and compression ratio. They can be calculated with Eq. 4.1. The extractive fragment coverage measures the percentage of words in the summary that are part of an extractive fragment from the input document. The extractive fragment density assesses the average length of the extractive fragment where each word

in the target summary belongs. The compression ratio is the word ratio between the input documents and their target summaries. The results of these three measures are visualized using kernel density estimation. Fig. 5.2 shows that three summary sections in the BigSurvey-MDS subset have similar distributions in coverage and density. Their densities are low, and their coverages vary in a relatively large range. The BigSurvey-Abs subset varies largely along the y-axis (extractive fragment density), which suggests varying writing styles of target summaries.

5.3 Method

In this chapter, we propose a solution named category-based alignment and sparse transformer (CAST) to summarize numerous academic papers on one research topic. CAST contains three main components: the BERT-based sentence classification with context (SCC) model, the sequential sentence classification (SSC) model, and the transformer-based abstractive summarization model with sparse attention.

Each section of the structured summary usually focuses on a specific aspect of the content from input documents. To prepare each summary section's content, we classify sentences in the extracted introduction section of a survey paper and merge sentences classified as the same type. We design a method named sentence classification with context (SCC) to classify these sentences. Given a sentence and the sentences before and after it, we concatenate them as the input for the sentence classification model based on a pre-trained model (e.g., BERT [32] or RoBERTa [79]). We train the SCC model on the labeled sentences from the CSABST dataset [24], in which each sentence is annotated as one of 5 categories: background, objective, method, result, and other.

To deal with the above problem, we use category-based alignment (CA) to align each summary section with input sentences classified as the same type. The aligned input text and target summary compose the example for model training. CA can be

Table 5.3: Automatic evaluation results of each summary segment on the BigSurvey-MDS test set.

Method	**Background**	**Method**	**Other**
	R1/R2/RL	**R1/R2/RL**	**R1/R2/RL**
LexRank+CA	31.33/5.92/13.93	28.85/4.65/13.07	23.61/6.08/13.04
TextRank+CA	31.20/5.79/13.91	28.80/4.38/12.94	24.41/6.42/13.81
BART	31.96/5.73/14.96	28.61/4.97/14.32	23.87/5.74/13.50
BART+CA	33.05/6.21/15.40	29.22/5.22/14.57	25.39/6.58/14.44
PEGASUS	33.51/6.74/15.67	27.47/4.93/14.17	25.20/6.55/14.02
PEGASUS+CA	33.93/6.80/15.67	29.76/5.74/15.05	26.32/7.34/15.34
BigBird-PEGASUS	34.31/6.78/15.54	29.46/5.47/14.43	26.07/6.66/14.24
LED	34.11/6.84/15.78	26.15/4.59/13.47	25.26/6.34/13.74
CAST-BigBird	34.56/6.96/15.55	30.83/5.95/15.03	26.90/7.47/15.45
CAST-LED	**36.55/8.82/16.87**	**31.72/6.94/15.61**	**27.16/8.10/15.53**

regarded as a content selection operation based on sentence classification, supporting the summarization model to focus on specific aspects of input documents.

Considering that different sections of the structured summary can be written in different ways, we train multiple models to produce separate sections in the target summary. To prepare the pairs of input and output for model training, aligning all summary sections with the same input (one-to-many) is straightforward. Merged from dozens of reference papers' abstracts, the input text of each example in the BigSurvey-MDS usually contains multiple aspects of content. For a summary section focusing on a specific aspect, other aspects of input content can be regarded as noises. Using the same input for producing different summary sections can make the produced sections mix different aspects of content.

Table 5.4: Automatic evaluation results of combined summary on the BigSurvey-MDS test set.

Method	**R1/R2/RL**
LexRank	35.85/8.59/14.22
LexRank+CA	37.92/8.56/14.63
TextRank	36.35/8.49/14.24
TextRank+CA	38.22/8.45/14.70
BART	37.64/8.45/15.69
BART+CA	40.21/9.38/16.06
PEGASUS	38.91/9.00/16.20
PEGASUS+CA	41.09/9.96/16.76
BigBird-PEGASUS	41.29/9.84/16.37
LED	39.79/9.42/16.05
CAST-BigBird	42.10/10.24/16.71
CAST-LED	**43.13/11.64/17.35**

Classifying sentences from reference papers' abstracts can be defined as a sequential sentence classification (SSC) problem. We follow the setting in [24] and train a BERT-based SSC model on the datasets named CSABST [24]. We first use the SSC model to classify the sentences in each reference paper's abstract and then merge the sentences classified as the same type. Our evaluation results show that the SSC model can outperform the SCC model in abstract sentence classification. Considering the target summary in BigSurvey-MDS is usually much longer than the samples in the CSABST and the max length limit of the BERT model, it is not appropriate to use the SSC model trained on the CSABST to classify the target summary's sentences. Therefore, we utilize the SSC model to classify input sentences and the SCC model to classify sentences in the target summary.

The original transformer model's encoder adopts the self-attention mechanism scaling quadratically with the number of tokens in input sequences [131]. It is prohibitively expensive for the long input sequence [20] and precludes fine-tuning large pre-trained models with limited computational resources. Some transformer models' variants adopt sparse attention mechanisms to reduce the complexity. For example, BigBird [147] and Longformer [6] combine three different types of attention mechanisms and scale linearly with sequence length. Considering the constraint of GPU memory, our CAST model employs the pre-trained encoder with sparse attention to encode longer input texts. Our CAST model has two versions: the CAST-BigBird employs the BigBird [147] as the encoder, and the CAST-LED's encoder is from the Longformer [6].

5.4 Experiments

5.4.1 Baselines

In our experiments, we compare various extractive and abstractive summarization models on our BigSurvey dataset. These models' details are shown in Table 5.8.

LexRank and TextRank. Two unsupervised extractive summarizers are built on graph-based ranking methods [37, 84].

CopyTransformer. Gehrmann et al. [41] add the copy mechanism [116] to the transformer model for abstractive summarization.

BART. Lewis et al. [64] build a sequence-to-sequence denoising autoencoder that is pre-trained to reconstruct the original input text from the corrupted text.

PEGASUS. [149] pre-train a transformer-based model with the Gap Sentences Generation (GSG) and Masked Language Model (MLM) objectives.

BigBird-PEGASUS. Zaheer et al. [147] combine the BigBird encoder with the decoder from the PEGASUS model.

Longformer-Encoder-Decoder (LED).LED [6] is built on BART and adopts the local and global attention mechanisms in the encoder part, while its decoder part still utilizes the original self-attention mechanism.

We fine-tuned large models of these pre-trained summarizers on BigSurvey's training set.

5.4.2 Experimental Setting

The vocabulary's maximum size is set as 50,265 for these abstractive summarization models, while the BERT-based classifiers use 30,522 as default. We use dropout with the probability 0.1. The optimizer is Adam with β_1=0.9 and β_2=0.999. Summarization models use learning rate of $5e^{-5}$, while the classifiers use $2e^{-5}$. We also adopt the learning rate warmup and decay. During decoding, we use beam search with a beam size of 5. Trigram blocking is used to reduce repetitions. We adopt the implementations of PEGASUS, BigBird, and LED from HuggingFace's Transformers [138]. The BART's implementation is from the fairseq [93]. All the models are trained on one NVIDIA RTX8000.

5.4.3 Results and Discussion

In our experiments, we train and evaluate various summarization models on the BigSurvey-MDS and BigSurvey-Abs. We divide the BigSurvey-MDS into three subsets and train three models producing separate sections in the target summary. In this section, we report and analyze our experimental results.

To compare the quality of summaries produced by these models, we conduct automatic evaluation and report the ROUGE F_1 scores [67], including the overlap of

Table 5.5: Automatic evaluation results on the BigSurvey-Abs.

Method	**R-1**	**R-2**	**R-L**
LexRank	30.93	8.53	15.54
TextRank	32.21	8.79	15.96
CopyTransformer	30.59	5.80	16.76
BART	35.28	9.71	17.89
PEGASUS	37.47	11.08	19.25
LED	38.57	11.52	19.36
BigBird-PEGASUS	**39.75**	**12.60**	**20.11**

Table 5.6: Human evaluation results on the test set of BigSurvey-MDS. "Win" denotes that the generated summary of our CAST-LED is better than that of the original LED model in one aspect. "Tie" represents that two summaries are comparable in one aspect.

	Win	**Lose**	**Tie**	**Kappa**
Informativeness	39.5%	25.0%	35.5%	0.659
Fluency	28.5%	27.5%	44.0%	0.631
Non-Redundancy	33.0%	25.5%	41.5%	0.623

unigrams (R-1), bigrams (R-2), and longest common subsequence (R-L). We report ROUGE scores of produced three summary sections and the combined summaries for the BigSurvey-MDS in Tables 5.3 and 5.4. It shows that these abstractive summarization models can outperform these extractive models on the BigSurvey-MDS. Replacing the encoder's self-attention mechanism with sparse attention mechanisms can enable us to train transformer-based models on longer input texts with limited GPU memory. The BigBird-PEGASUS and the LED outperform other transformer-based models without sparse attention, which reveals that introducing longer input

Table 5.7: Ablation study on the test set of BigSurvey-MDS. We report the ROUGE scores of combined summaries. "w/o sparse attn" denotes using the original self-attention in the encoder. "w/o CA" represents removing the category-based alignment.

	R-1	**R-2**	**R-L**
CAST-LED	**43.13**	**11.64**	**17.35**
w/o sparse attn	40.21	9.38	16.06
w/o CA	39.79	9.42	16.05
w/o CA + LED$_{base\text{-}8192}$	39.38	9.78	16.30

Table 5.8: Details of summarization models.

Model	**Architecture**	**Params**	**Enc/Dec Layers**	**Heads**	$\mathbf{d_{model}}$	$\mathbf{d_{ff}}$	**Input Len**
CopyTransformer	Enc-Dec	81.5M	4	8	512	2,048	512
BART$_{large}$	Enc-Dec	406.3M	12	16	1,024	4,096	1,024
PEGASUS$_{large}$	Enc-Dec	570.8M	16	16	1,024	4,096	1,024
LED$_{\text{large}}$	Enc-Dec	459.8M	12	16	1,024	4,096	16,384
BigBird-PEGASUS	Enc-Dec	577.1M	16	16	1,024	4,096	4,096

text can benefit the quality of generated summaries.

The concatenation of input reference papers' abstracts usually contains multiple aspects of content. It requires the summarization method to have a strong capability of content selection to produce a summary section precisely covering a specified aspect. We compare the effects of applying different ways of alignment (one-to-many or category-based alignment) on various summarization models. When using the one-to-many alignment, we observe that the produced summary sections often mix multiple aspects of content and have overlapping content in different sections. It reveals that these summarization models still have difficulties in content selection,

although they have supervision from target summaries. Tables 5.3 and 5.4 show that introducing CA can bring extra performance gains for various summarization models. It reflects the effectiveness of CA and the need to enhance summarization models' capabilities of content selection. Combining the CA and the transformer model with the sparse attention mechanism, CAST-LED outperforms other baseline models on the BigSurvey-MDS.

Table 5.5 shows the evaluation results on the BigSurvey-Abs. These transformer-based abstractive summarization models with sparse attention mechanisms also outperform other baselines. It reveals that modeling longer input text is also important for summarizing survey papers in the BigSurvey-Abs. Besides, the pre-trained sequence-to-sequence models outperform the model trained from scratch.

In addition to automatic evaluation, we performed a human evaluation to compare two summarization models' generated summaries in terms of their informativeness (the coverage of input documents' content), fluency (content organization and grammatical correctness), and non-redundancy (fewer repetitions). We randomly selected 50 samples from the test set of the BigSurvey-MDS. Four annotators are required to compare two models' generated summaries that are presented anonymously. We also assess their agreements by Fleiss' kappa [39]. Human evaluation results in Table 5.6 exhibit that our CAST-LED method outperforms the original LED model in terms of informativeness and non-redundancy.

We also conduct the ablation study to validate the effectiveness of individual components in our method. Table 5.7 shows that using the original self-attention to replace the sparse attention mechanism in the encoder part or removing the category-based alignment can lead to performance degradation. Besides, increasing the input sequence length cannot replace the CA. The longer inputs can introduce more noise, and it is still difficult for summarization models to select the salient content on the specific aspect without CA. The results verify the effectiveness of the sparse attention mechanism and CA.

5.5 Chapter Summary

In this chapter, we study how to summarize numerous academic papers about the same topic into a structured summary. Existing multi-document summarization (MDS) work usually focuses on producing an unstructured summary that encompasses only a limited number of input documents. Meanwhile, previous structured summarization work focuses on summarizing each document into a multi-aspect summary. Existing methods and datasets fail to fulfill the demands of summarizing numerous academic literature. We propose BigSurvey, the first large-scale dataset for generating comprehensive summaries of numerous academic papers on each topic. Besides, we propose the category-based alignment and sparse transformer (CAST) to effectively arrange the diverse content from a large number of input documents while simultaneously ensuring efficiency when processing long inputs.

Chapter 6

From High-Resource to Low-Resource: Low-Resource Court Judgment Summarization for Common Law Systems

6.1 Introduction

Common law systems rely on case precedents, which encompass not only judicial decisions within a particular jurisdiction but also decisions from all jurisdictions throughout the common law world [33]. Judges in common law jurisdictions need to find similar precedents (prior cases) resolved in the past and refer to the rationale employed in previous decisions [9]. Court judgment documents typically contain long text that comprehensively discusses each case and provides detailed explanations of judges' decisions. Reading previous cases' judgment documents is crucial for legal practitioners in common law jurisdictions. There exist massive reported cases in common law jurisdictions, and the number of cases is still increasing [33], which

makes it difficult for legal practitioners to read through abundant cases' judgment documents.

High-quality summaries of judgment documents can facilitate readers to quickly browse key information. Nevertheless, employing legal experts to write summaries costs a lot and has limited coverage of new or atypical cases' judgments [58]. Alternatively, we can leverage automatic text summarization technology to generate summaries for court judgments. Our objective is to enable computers to generate high-quality court judgment summaries that are informative, coherent, and concise. To accomplish this objective, we must address several problems: 1) the lack of datasets, 2) training supervised summarization models with very limited labeled data, 3) identifying the salient content dispersed within the long judgment document, and 4) improving the efficiency of summarization models and training methods to process long input documents and summaries.

There are very few court judgment summarization datasets. Some focus on civil law jurisdictions [28, 43], while others concentrate on judgments from specific common law jurisdictions [5, 120]. Considering judges need to compare similar precedents across all common law jurisdictions [33], summaries of court judgments from multiple common law jurisdictions are helpful for comparing and analyzing similar precedents efficiently. The formats and content of court judgment documents vary across different jurisdictions. A summarization method that exhibits proficiency in processing judgment documents from one jurisdiction may not perform well when applied to judgments from other jurisdictions. Current summarization datasets are insufficient to satisfy the demands of summarizing court judgment documents across multiple common law jurisdictions and comprehensively evaluating judgment summarization methods. In order to address the lack of datasets, we introduce CLSum, the first large-scale dataset for summarizing multi-jurisdictional common law court judgment documents. [1] CLSum has four subsets for court judgments from Canada, Australia,

[1]The CLSum dataset will be made publicly available upon publication.

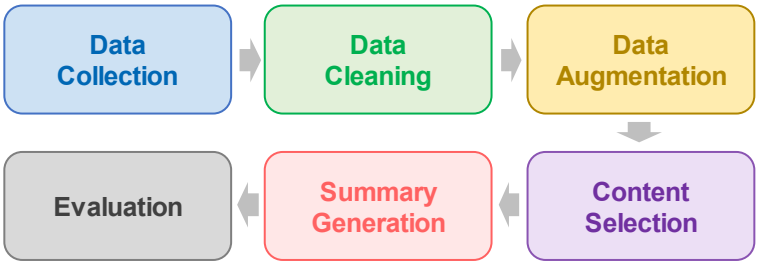


Figure 6.1: Our workflow of Court Judgment Summarization.

the United Kingdom, and Hong Kong SAR.

We propose a solution for low-resource court judgment summarization to address the remaining three problems. Fig. 6.1 depicts six key steps in our solution, including data collection, data cleaning, data augmentation, content selection, summary generation, and evaluation.

Similar to other domain-specific tasks, court judgment summarization usually suffers from the shortage of labeled data. Court websites in most jurisdictions publish only a small number of judgment summaries of typical cases. Under this low resource condition, summarization models' few-shot and zero-shot performance are important. We carry out extensive comparative experiments to analyze the effect of the training set size on summarization models' performance. In addition to selecting models with good few-shot and zero-shot performance, expanding the dataset is another way to improve summarization performance. To expand our CLSum's training sets and reduce overfitting, we adopt the large language model (LLM) for data augmentation. Meanwhile, we introduce legal knowledge into the prompts to constrain the LLM to properly use legal concepts when synthesizing sample text in the data augmentation process.

Another challenging issue is identifying and integrating the salient information scattered in long judgment documents. Accordingly, our solution has a two-stage summarization framework to deal with this issue from coarse to fine. The first stage, named

salient content selection, can be regarded as a rough selection. The recall of essential content that should be retained in summaries is maximized during the compression of long inputs. Subsequently, the condensed inputs are passed to summarization models for fine-grained content selection and integration. Compressing long inputs also improves the efficiency of summarization models. Further improving the efficiency requires modifying the models and training methods to reduce complexity.

The complexity of the self-attention mechanism in the transformer model [131] exhibits a quadratic increase with the input length. It can take up a lot of GPU memory and limit transformer-based models' efficiency. The model complexity can be reduced by substituting the original self-attention mechanism with sparse attention mechanisms. Given the constraints of GPU memory size, models equipped with sparse attention mechanisms can be pre-trained on longer text sequences and fine-tuned to generate the full summary directly. For models pre-trained on the shorter input sequences, we adopt a divide-and-conquer-based training strategy for generating summary segments, followed by merging them to form the final summary. To further reduce the consumption of GPU memory and fine-tune large language models (LLMs) on off-the-shelf GPUs, we adopt some memory-efficient training techniques, like gradient accumulation, gradient checkpointing [2], parameter quantization [29, 139], memory-efficient optimizer [30, 104], and adding parameter-efficient adapters [31, 55].

We adopt various summarization methods as baselines and evaluate them on our CLSum dataset. For performance comparison, we carry out automatic evaluation and human evaluation. Apart from the widely used Recall-Oriented Understudy for Gisting Evaluation (ROUGE) scores [67] and the model-based BARTScore [145], we design a legal knowledge enhanced evaluation metric named LTScore, which is based on foundation models fine-tuned on the legal corpus. Legal texts usually contain more legal terms compared to texts in the general domain. These terms should be used

[2]github.com/cybertronai/gradient-checkpointing

accurately in court judgments and their respective summaries. Therefore, LTScore assigns greater weights to legal terms in judgment summaries to better assess the accurate usage of such legal terms.

The contribution of this work is threefold:

- We present CLSum, the first large-scale dataset for summarizing common law court judgment documents from multiple jurisdictions.
- We are the first to employ large language models for data augmentation, summary generation, and evaluation in court judgment summarization.
- We design a legal knowledge enhanced evaluation metric named LTScore to evaluate generated legal text.

6.2 CLSum Dataset

Common law court judgment summarization (CLSum) is a large-scale summarization dataset covering court judgments from four common law jurisdictions, including Canada, Australia, the United Kingdom, and Hong Kong SAR. This section first presents our procedures for collecting and pre-processing data. Subsequently, we describe four subsets in CLSum. Furthermore, we carry out statistics on CLSum and perform a comparative analysis with other datasets.

6.2.1 Collecting and Pre-processing Data

Court judgment documents usually comprehensively discuss each case and explain judges' decisions. Electronic files of judgments are usually publicly available online. Judiciaries usually publish judgment summaries of typical cases to the public. We collected court judgment documents together with their summaries from court websites

Table 6.1: Summarization datasets' statistical information. "Samples" is the sample number in the dataset. "Doc" and "Sum" stand for the input document and target summary. "Sents" and "Words" represent the mean number of sentences and words. "Dens." and "Cov." are the density and coverage of extractive fragments.

Dataset	**Samples**	**Sents (Doc)**	**Words (Doc)**	**Sents (Sum)**	**Words (Sum)**	**Dens.**	**Cov.**
CNN/DM	312,085	39.8	810.6	3.7	56.2	3.8	0.9
PubMed	133,215	87.5	3049.0	6.8	202.4	5.8	0.8
arXiv	215,913	205.7	6029.9	9.6	272.7	3.8	0.9
CLSum-CA	192	1,168	38,403	38	748	0.8	0.5
CLSum-HK	233	395	11,911	46	1,169	9.7	0.9
CLSum-UK	793	458	16,143	41	1,241	2.4	0.7
CLSum-AUS	1,019	630	20,485	19	592	1.4	0.6

in four jurisdictions.

After collecting thousands of court judgments' HTML or PDF files, we parse these files and extract their content. Then, we conduct a series of data pre-processing operations, including eliminating noises, eliminating replicated examples and outliers with excessively short content, and splitting three sets for training, validation, and test.

6.2.2 Description of the CLSum's Subsets

We collected multi-jurisdictional common law court judgment documents to build the CLSum dataset. CLSum consists of four subsets, namely CLSum-CA, CLSum-HK, CLSum-UK, and CLSum-AUS.

CLSum-CA is collected from the website of the Supreme Court of Canada (SCC)[3]. We collect the case briefs and corresponding judgment documents from 2018 to 2023. CLSum-CA is the subset with the smallest number of samples.

CLSum-HK is collected from the legal reference system[4]. It covers typical cases from multilevel courts, including the Coroner's Court, Magistrates' Court, District Court (DC), High Court (HC), and Court of Final Appeal (CFA) in Hong Kong. We collect these cases' judgment documents and their press summaries from 2012 to 2023.

CLSum-UK is the subset focusing on the United Kingdom Supreme Court's judgment documents[5]. We collect British judgment documents and their press summaries from 2009 to 2023.

CLSum-AUS is collected from the High Court of Australia's website[6]. We collect Australian judgment documents and their summaries from 2005 to 2023. CLSum-AUS is the subset with the most samples.

6.2.3 Dataset Analysis

We conduct statistics on CLSum's four subsets and perform a comparative analysis with other datasets. As shown in Table 6.1, these four subsets' input documents and target summaries are much longer in comparison to existing datasets. The formats and content of court judgments in these four subsets are different. CLSum-HK and CLSum-CA have a few samples. CLSum-UK and CLSum-AUS have more samples. Among these four subsets, CLSum-CA and CLSum-AUS have longer input documents but shorter target summaries.

[3]www.scc-csc.ca/case-dossier/cb/index-eng.aspx

[4]legalref.judiciary.hk/lrs/common/contactus/contactus.jsp

[5]www.supremecourt.uk/decided-cases/

[6]www.hcourt.gov.au/publications/judgment-summaries/2023-judgment-summaries

Table 6.2: The percentage of target summaries' new n-grams.

Dataset	**unigrams**	**bigrams**	**trigrams**	**4-grams**
arXiv	15.04	48.21	71.66	83.26
PubMed	18.38	49.97	69.21	78.42
CNN/DM	19.50	56.88	74.41	82.83
CLSum-CA	**21.65**	**58.00**	**80.90**	**90.09**
CLSum-HK	13.48	38.86	57.53	69.06
CLSum-UK	15.00	36.25	53.71	64.29
CLSum-AUS	11.65	37.69	58.01	70.53

In order to quantify the abstraction level of CLSum's target summaries, Table 6.2 counts the ratio of target summaries' n-grams that do not appear in the inputs. Target summaries of CLSum-CA exhibit a greater number of new n-grams and a higher abstraction level. The abstraction level of target summaries in CLSum-HK, CLSum-UK, and CLSum-AUS is comparatively lower than that in other datasets.

Additionally, we utilize two measures [44], including the coverage and density of extractive fragments, to evaluate summarization datasets' extractive nature. As shown in Table 6.1, CLSum-HK's coverage is similar to previous summarization datasets but is higher than that of other subsets in CLSum. Among these four subsets, CLSum-CA and CLSum-AUS have smaller coverage and density of extractive fragments. Fig. 6.2 depicts the visualization of distributions of two measures using kernel density estimation. CLSum-HK and CLSum-UK subsets have high variability in density, which suggests their target summaries are written in varying styles. Furthermore, the compression ratio is calculated by dividing the word count of a document by that of its corresponding summary.

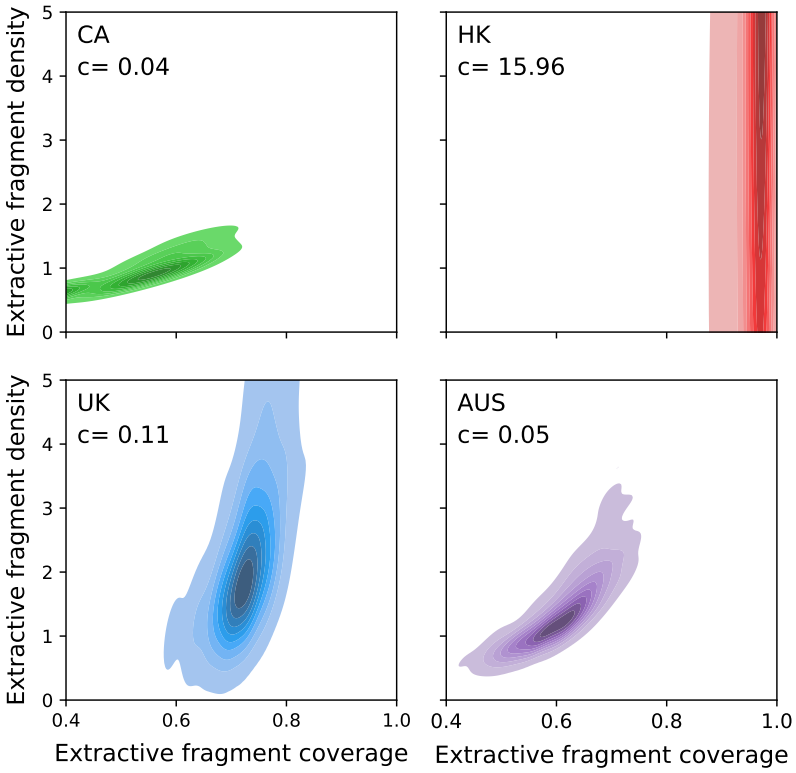


Figure 6.2: Distributions of extractive fragment coverage and extractive fragment density. "c" denotes the compression ratio.

6.3 Method

Summarizing court judgment documents under low resource conditions has several problems, including: training supervised models with extremely limited labeled data, identifying the salient content dispersed within the long judgment document, and improving the efficiency of summarization models and training methods to process long input documents and summaries. This section presents our solution to address the aforementioned problems.

Fig. 6.1 depicts that our solution consists of six key steps: data collection, data cleaning, data augmentation, content selection, summary generation, and evaluation. Our data collection and cleaning procedures are introduced in subsection 6.2.1. After

the first two steps, we conduct data augmentation to expand the training sets and reduce overfitting to the limited training samples. Then, the content selection and summary generation steps complete the selection and integration of key information from rough to fine. Our evaluation step comprises both automatic evaluation and human evaluation for assessing the summaries generated by various summarization models. To the best of our knowledge, this is the first court judgment summarization work adopting LLM in data augmentation, summary generation, and evaluation. This section will introduce our methods for training supervised summarization models with very limited labeled data, identifying salient content scattered in long judgment documents, and improving the efficiency of the summarization model and its training process.

6.3.1 Mitigating the Impact of Insufficient Labeled Samples

In most jurisdictions, courts only release a limited number of judgment summaries for typical cases. The limited size of the labeled training set usually hinders the performance of supervised models trained from scratch. It is essential to guarantee the summarization model's performance when generalizing to cases not seen during training. Labeling large-scale datasets can cost a lot, while unlabeled data can be easily collected from the Internet. Researchers pre-train foundation models with self-supervised tasks on massive unlabeled data to learn better text representations. These foundation models can provide good initialization and reduce the amount of labeled training samples required for downstream tasks. By fine-tuning on the downstream task, these foundation models often outperform models trained on the same task from scratch. To mitigate the impact of insufficient labeled samples from the summarization model perspective, we evaluate diverse foundation models' few-shot and zero-shot performance on our CLSum dataset and select the best performing model.

Meanwhile, we also study mitigating the impact of insufficient labeled samples from the data perspective. We propose knowledge-constrained rephrasing, an LLM-based data augmentation method constrained by legal knowledge. We also compare it with different data augmentation methods like back translation and rephrasing. These data augmentation methods can expand the training sets and reduce overfitting to the limited training samples. The back translation is a commonly used data augmentation method. It first translates the text into another language (e.g., from English to German) and then translates it back to the original language (e.g., from German to English) [117]. The rephrasing method employs large language models to rephrase each sentence in judgment documents and target summaries [26]. Legal texts usually contain more legal terms compared to texts in the general domain. These terms must be used correctly in court judgments and their summaries. Therefore, we propose knowledge-constrained rephrasing, which introduces legal knowledge into the prompts of LLMs to constrain the synthesized sentences to correctly use legal concepts in the data augmentation process. These generated sentences are merged as new data samples. We supplement the synthetic data into the training sets to alleviate the impact of insufficient labeled data.

6.3.2 Salient Content Identification and Integration

Judgment documents in our CLSum datasets usually contain tens of thousands of words, as depicted in Table 6.1. The salient content is dispersed within different parts of these long judgment documents. Nonetheless, foundation models are commonly pre-trained on text sequences that have a predetermined maximum length. When abstractive summarization models cannot accept the entire document as input, compressing the input length while preserving key information is important. Apart from simply truncating the document, there exist more efficient content selection methods.

We conduct two-stage operations to identify and integrate the salient content from

Table 6.3: Evaluation results of content selection methods. "R_1" is the unigram recall, and "R_{avg}" represents the mean value of the recalls of unigram, bigram, trigram, and 5-gram. "Lead" represents the truncation method.

Method	CLSum-CA		CLSum-HK		CLSum-UK		CLSum-AUS	
	$\mathbf{R_1}$	$\mathbf{R_{avg}}$	$\mathbf{R_1}$	$\mathbf{R_{avg}}$	$\mathbf{R_1}$	$\mathbf{R_{avg}}$	$\mathbf{R_1}$	$\mathbf{R_{avg}}$
Lead	68.41	30.46	85.67	52.29	**83.81**	**54.47**	80.05	46.95
LexRank	69.10	30.61	85.61	52.38	83.15	53.84	85.83	50.91
TextRank	**69.88**	**30.88**	**85.68**	**52.45**	83.06	53.70	**85.90**	**50.91**

coarse to fine. The first stage, named salient content selection, can be regarded as a rough selection. The recall of essential content that should be retained in summaries is maximized during the compression of long inputs. Subsequently, the condensed inputs are passed to summarization models for fine-grained content selection and integration.

The step of content selection is designed to preserve the maximum key information when compressing the input to a fixed length. We compare different methods' performance and mainly focus on their average recall of n-grams. Table 6.3 shows these methods' evaluation results on the training set of CLSum. We choose the most effective method for content selection in our subsequent experiments. Specifically, we adopt TextRank for CLSum-CA, CLSum-HK, and CLSum-AUS and use truncation for CLSum-UK.

6.3.3 Improving the Efficiency of Models and Training Methods

Most real-world applications not only face low data resources but also have the constraint of low computing resources. Especially when available computing resources are

limited, how to improve the efficiency of model training and inference is an important issue. Many summarization methods require large computing resources when processing long documents, which limits their applications. In transformer-based models [131], the self-attention mechanism's complexity exhibits a quadratic increase with the input length. It can take up a lot of GPU memory and limit models' efficiency. Moreover, the limited GPU memory size poses constraints on transformer-based models' capability to model longer context. To improve summarization models' efficiency, we employ sparse attention mechanisms [6, 27, 45]. Summarization models employing sparse attention mechanisms have the capability to model longer contexts with the same size of GPU memory.

In addition to efficient models, efficient training methods can also expedite the training process. We adopt some memory-efficient training methods, like gradient accumulation, gradient checkpointing [7], parameter quantization [29, 139], memory-efficient optimizer [30, 104], and adding parameter-efficient adapters [31, 55]. For those models pre-trained on the shorter input sequences, we adopt a divide-and-conquer-based training strategy for generating summary segments, followed by merging them to form the final summary. Additionally, our two-stage summarization schema also reduces the context length that neural summarization models need to model, thus reducing the associated GPU memory consumption. These efficient training methods enable us to fine-tune LLMs on lengthy inputs using one off-the-shelf GPU.

6.4 Experiments

6.4.1 Baselines

We employ various summarization methods as baselines and evaluate them on the CLSum dataset. These models' details are shown in Table 6.11.

[7] github.com/cybertronai/gradient-checkpointing

TextRank and LexRank [37, 84] are graph-based ranking methods that are widely employed in unsupervised extractive summarization.

Longformer-Encoder-Decoder (LED) [6] is built on the architecture of the BART model [64] and employs sparse attention mechanisms to replace the original self-attention mechanism within its encoder.

Legal-LED [8] is the LED model fine-tuned on the litigation releases of U.S. Securities and Exchange Commission (SEC)[9].

LongT5 [45] replaces the self-attention mechanism with a global-local attention mechanism in the encoder part of T5 model [103] to model longer inputs.

LLaMA [129] is a collection of LLMs with parameter sizes ranging from 7B to 65B, which are trained on publicly available data.

Vicuna [17] is a set of LLaMA models fine-tuned with user-shared ChatGPT conversation data.

GPT-3.5-turbo[10] is the model employed in the ChatGPT. Its fine-tuning process employs Reinforcement Learning from Human Feedback (RLHF) on the GPT-3.5 model [92].

6.4.2 Experimental Setting

For LED-based models (LED and Legal-LED), the vocabulary size is set as 50,265, whereas LLaMA-based models (Vicuna and LLaMA) and LongT5 model utilize a default vocabulary size of 32,000 and 32,128, respectively. When fine-tuning the LED-based model, we set the learning rate to $5e^{-5}$. The LLaMA-based models and LongT5 model use $2e^{-5}$ and $1e^{-3}$, respectively. We utilize the warmup and decay of

[8] github.com/nsi319/Legal-Summarizer

[9] www.sec.gov/litigation/litreleases.htm

[10] We adopt the GPT-3.5-turbo-0301 API from Azure Cloud

the learning rate for all these models. As for the optimizer, we use Adam [60] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for LED-based models and Adafactor [119] for T5-based models. When fine-tuning LLaMA and Vicuna models, we use 4-bit NormalFloat (NF4), QLoRA, and 32-bit paged AdamW optimizer [31] to save GPU memory. Different foundation models are pre-trained on texts of different lengths. In the fine-tuning stage, we predefine the maximum input length for each model to match its input length during the pre-training. Given the constraints of GPU memory size, models equipped with sparse attention mechanisms (e.g., LongT5, LED, Legal-LED) can be pre-trained on longer text sequences. Their maximum input length is 16,384. We fine-tuned them to generate the full summary directly. For models pre-trained on the shorter input sequences (e.g., LLaMA and Vicuna), we utilize the divide-and-conquer-based training strategy for generating summary segments, followed by combining them to form the final summary. We employ the beam search, whose beam size is five. We utilize the implementations of LongT5, LED, Legal-LED, and Vicuna from HuggingFace's Transformers [138] and LLaMA's implementation from Touvron et al. [129]. We fine-tune these models using one GPU named Nvidia RTX8000.

6.4.3 Evaluation Metrics

We carry out automatic evaluation and human evaluation for assessing the summaries generated by various models. The automatic evaluation metrics we used can be further divided into statistics-based evaluation metrics (e.g., ROUGE) and model-based evaluation metrics (e.g., BARTScore). We not only employ commonly used evaluation metrics but also propose novel evaluation metrics.

We present F_1 scores of ROUGE [67] in our experimental results. Specifically, we measure overlaps of unigrams (R-1), bigrams (R-2), and the longest common subsequence (R-L) between output summaries and target summaries.

$$\text{BARTScore} = \sum_{t=1}^{m} \omega_t \log p\left(\mathbf{y}_t \mid \mathbf{y}_{<t}, \mathbf{x}, \theta\right) \tag{6.1}$$

BARTScore [145] is a model-based evaluation metric assessing the quality of generated text by formulating it as a text generation task. Built on the pre-trained BART model [64], BARTScore calculates the log probability of one text sequence y when given another text sequence x. In Eq. 6.1, θ represents the given pre-trained BART model's parameters. Lewis et al. [64] set equal weight ω_t for each token.

Compared with general documents, legal documents usually contain many specialized expressions and domain knowledge. Compared with the BART model trained on the general domain corpus, the models trained on legal instruments have a better command of these specialized expressions and domain knowledge. To evaluate the generated legal text, we design an evaluation metric named legal text score (LTScore). LTScore employs foundation models (e.g., LED and Vicuna) fine-tuned on our legal corpus to predict the log probability of each text sequence.

Legal texts usually contain more legal terms compared to texts in the general domain. These terms must be used accurately in court judgments and their summaries. Therefore, LTScore assigns greater weight to legal terms in judgment summaries to better evaluate whether these legal terms are used correctly. LTScore can be calculated according to the following formulas.

$$\text{LTScore}_P = \sum_{t=1}^{m} \omega_t \log p\left(\text{cand}_t \mid \text{cand}_{<t}, \text{ref}, \theta\right) \tag{6.2a}$$

$$\text{LTScore}_R = \sum_{t=1}^{m} \omega_t \log p\left(\text{ref}_t \mid \text{ref}_{<t}, \text{cand}, \theta\right) \tag{6.2b}$$

$$\text{LTScore}_{\text{F}_1} = \frac{2 \times \text{LTScore}_P \times \text{LTScore}_R}{\text{LTScore}_P + \text{LTScore}_R} \tag{6.2c}$$

$$\omega_t = \begin{cases} 1, & \text{if token}_t \notin g_i \\ 1 + e^{\omega_{g_i}}, & \text{if token}_t \in g_i \end{cases} \tag{6.2d}$$

$$\omega_{g_i} = \frac{\text{Score}(g_i) - \text{Score}(G)_{\min}}{\text{Score}(G)_{\max} - \text{Score}(G)_{\min}} \qquad g_i \in G \tag{6.2e}$$

The θ represents the given legal foundation model's parameters. Eq. 6.2a and Eq. 6.2b calculate the precision and recall of LTScore. In Eq. 6.2c, the F_1 score of LTScore is the arithmetic average of its recall and precision. We adopt the LED and Vicuna model [6] fine-tuned on our CLSum dataset to calculate $\log p$ in Eq. 6.2a and 6.2b. For each sample, we select the phrases appearing in the candidate sequence or reference sequence from the glossary of legal terms[11]. We rank these selected phrases according to their importance scores and then select the set of phrases G with top-100 importance scores $\text{Score}(G)$. In our experiments, we employ these phrases' tf-idf scores as their importance scores $\text{Score}(G)$. The g_i is the i-th phrase in the selected top-100 phrases set G. Eq. 6.2e calculates the Min-Max normalized importance score of g_i as ω_{g_i}. Eq. 6.2d calculates the weight ω_t of the t-th token token_t in the candidate sequence or reference sequence. If the t-th token token_t is a part of the phrase g_i, we add the exponential weight of phrase g_i to the t-th token's weight w_t.

LTScore enhances the adaptation to legal texts from two aspects: not only by injecting legal knowledge through fine-tuning the base model on the legal dataset but also by adjusting token weights to emphasize the precise use of legal terms.

[11] www.glossary.doj.gov.hk/

Table 6.4: Automatic evaluation results on test sets of CLSum-CA. "N examples" denotes using N examples when fine-tuning models.

Method	CLSum-CA			
	0 examples	**10 examples**	**50 examples**	**100 examples**
	R1 / R2 / RL	**R1 / R2 / RL**	**R1 / R2 / RL**	**R1 / R2 / RL**
LexRank	31.87/9.54/13.24	-	-	-
TextRank	32.03/9.36/13.57	-	-	-
GPT3.5	**50.01**/**18.58**/**20.62**	-	-	-
$LLaMA_{7B}$	39.88/11.90/15.99	46.41/15.95/19.00	45.61/15.80/20.08	47.91/18.10/20.74
$LLaMA_{13B}$	40.59/12.63/16.19	44.08/16.80/18.30	45.44/15.70/20.22	48.09/17.00/20.45
$Vicuna_{7B}$	47.32/16.42/20.00	47.94/16.78/20.77	47.02/17.00/21.64	47.62/17.36/22.05
$Vicuna_{13B}$	47.69/17.17/20.29	48.36/**17.50**/20.32	49.78/19.29/**22.72**	50.66/19.22/**22.68**
LongT5	23.29/6.08/10.23	48.97/12.79/18.10	52.77/19.23/21.15	55.85/19.98/21.48
LED_{Base}	23.63/6.83/11.58	51.41/16.76/20.75	55.10/19.39/21.36	54.57/19.63/21.32
Legal-LED	37.10/6.97/16.66	51.94/16.65/20.80	54.63/19.10/21.43	56.04/20.33/21.73
LED_{Large}	23.87/6.80/10.79	**54.37**/17.29/**20.85**	**56.27**/**19.52**/21.54	**57.23**/**21.15**/22.65

Table 6.5: Automatic evaluation results on test sets of CLSum-HK. "N examples" denotes using N examples when fine-tuning models.

Method	CLSum-HK			
	0 examples	**10 examples**	**50 examples**	**100 examples**
	R1 / R2 / RL	**R1 / R2 / RL**	**R1 / R2 / RL**	**R1 / R2 / RL**
LexRank	49.66/23.58/21.41	-	-	-
TextRank	51.50/24.36/23.65	-	-	-
GPT3.5	**54.28**/24.04/**25.13**	-	-	-
$LLaMA_{7B}$	47.60/18.22/20.91	53.15/23.15/24.31	50.66/22.81/25.39	51.71/23.30/26.18
$LLaMA_{13B}$	48.21/18.90/20.88	52.75/22.95/25.03	51.98/22.76/25.22	52.21/23.99/26.06
$Vicuna_{7B}$	53.01/23.20/23.94	**55.58**/**25.57**/**26.26**	54.84/25.26/26.50	55.01/25.26/26.42
$Vicuna_{13B}$	53.08/**24.45**/24.91	54.14/24.83/26.15	**56.04**/**26.99**/**27.67**	55.07/26.18/**26.78**
LongT5	46.73/16.63/19.32	51.35/19.38/21.39	55.36/24.81/23.50	56.29/**26.67**/24.85
LED_{Base}	47.26/18.16/19.85	53.03/21.36/21.89	53.62/23.52/22.65	55.56/25.47/23.04
Legal-LED	39.43/9.79/17.88	53.26/22.05/22.54	54.23/24.45/23.58	56.10/25.50/23.74
LED_{Large}	47.03/17.27/19.97	53.96/22.52/22.42	53.61/22.93/22.23	**56.43**/26.49/24.92

6.5 Results and Discussion

In this section, we exhibit our experimental results, and then we analyze and discuss these results. We carry out automatic evaluation and human evaluation for assessing the summaries generated by various summarization models. Furthermore, we carry out comprehensive comparative experiments to find essential model components and settings that are capable of improving summarization performance.

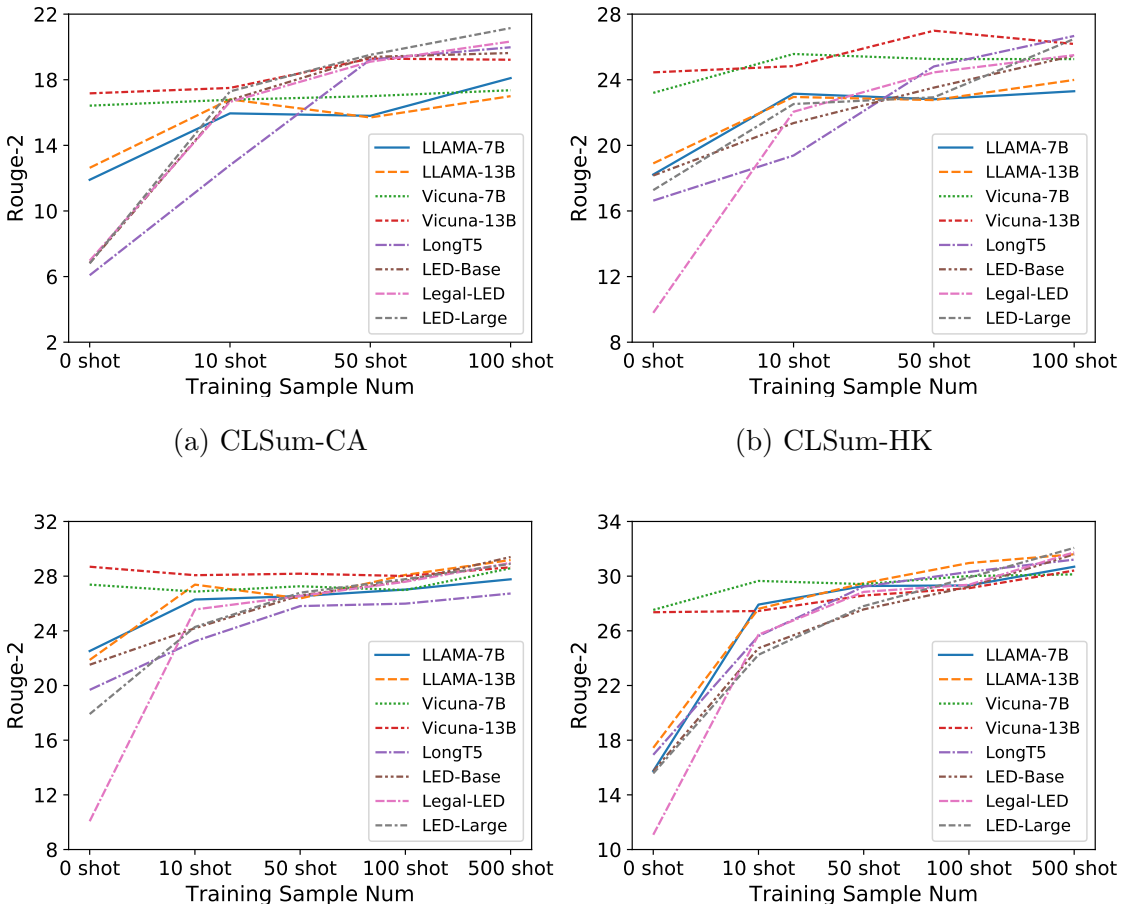
6.5.1 Summarization Results

We employ multiple metrics to assess the quality of the output summaries in the automatic evaluation. Specifically, we adopt the F_1 score of ROUGE [67][12], and some model-based metrics, including BARTScore$_{F1}$ and our proposed LTScore$_{F1}$. We fine-tune summarization models on training sets of increasing size (from zero examples to hundreds of examples). Tables 6.4, 6.5, 6.6, and 6.7 report ROUGE scores of final summaries.

Under the zero-shot setting, LLMs (GPT-3.5-turbo, LLaMA, and Vicuna) are competitive on all subsets of our CLSum dataset. The zero-shot performance of some pre-trained sequence-to-sequence models with hundreds of millions of parameters (LongT5, LED, and Legal-LED) is not as good as that of unsupervised extractive methods (LexRank and TextRank). As for the few-shot setting, even fine-tuning on only a few examples can bring obvious performance gains for these abstractive summarization methods, which validates the necessity of fine-tuning on downstream tasks. Fig. 6.3, Fig. 6.4, Fig. 6.5, and Fig. 6.6 illustrate the impact of training set size on the ROUGE-2 scores, BARTScore, and LTScore of the generated summaries.

In our human evaluation, we compare the outputs of summarization models based on their informativeness (i.e., cover salient content of input documents), fluency

[12]github.com/bheinzerling/pyrouge



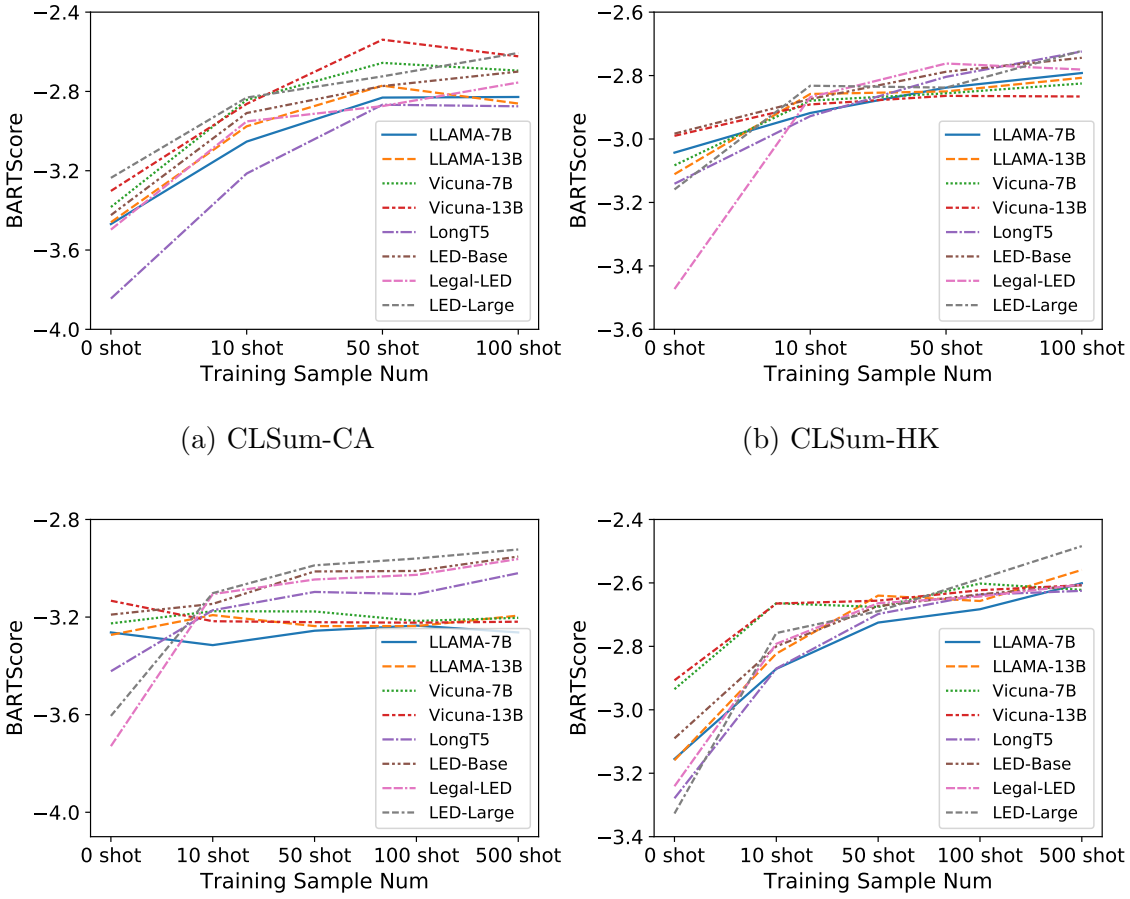
(a) CLSum-CA

(b) CLSum-HK

(c) CLSum-UK

(d) CLSum-AUS

Figure 6.3: Automatic evaluation result (ROUGE-2 Score) on CLSum.



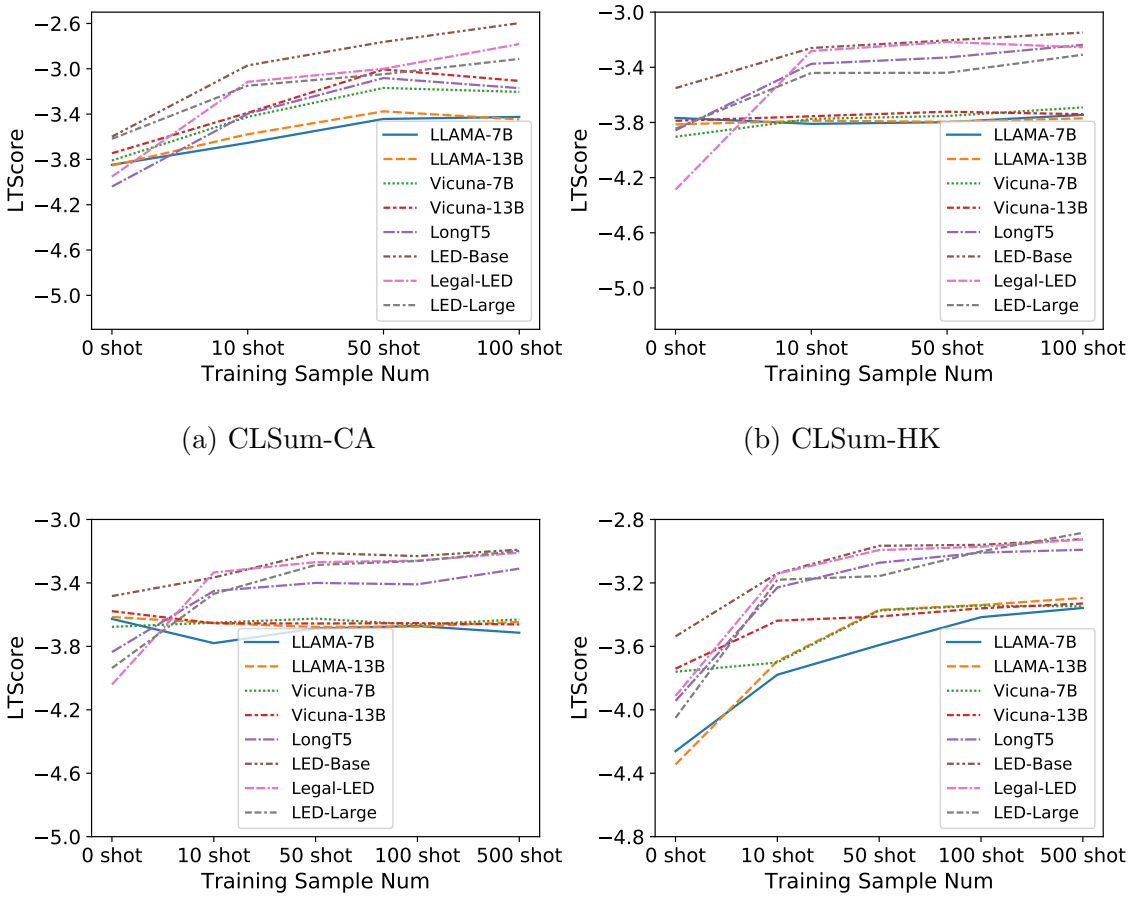
(a) CLSum-CA

(b) CLSum-HK

(c) CLSum-UK

(d) CLSum-AUS

Figure 6.4: Automatic evaluation result (BARTScore) on CLSum.



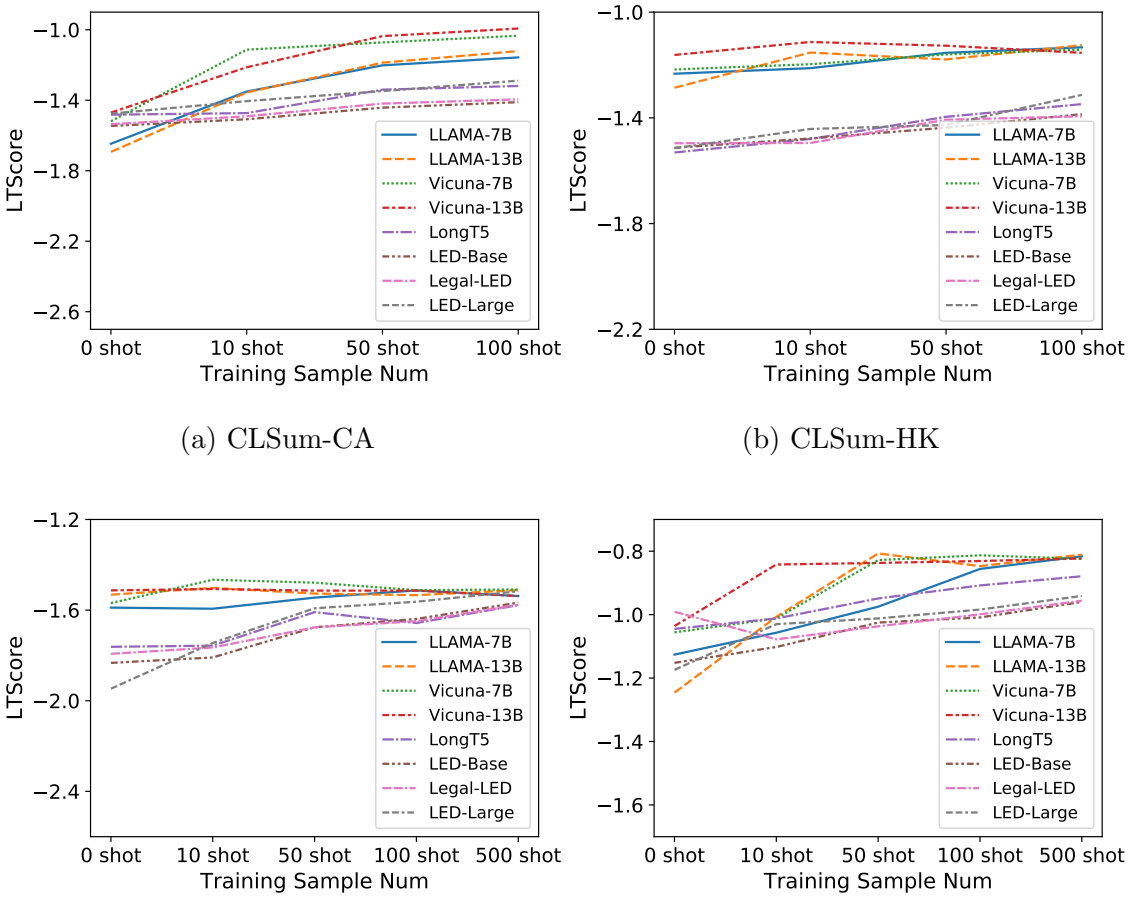
(a) CLSum-CA

(b) CLSum-HK

(c) CLSum-UK

(d) CLSum-AUS

Figure 6.5: Automatic evaluation result (LTScore-LED) on CLSum.



(a) CLSum-CA

(b) CLSum-HK

(c) CLSum-UK

(d) CLSum-AUS

Figure 6.6: Automatic evaluation result (LTScore-Vicuna) on CLSum.

Table 6.6: Automatic evaluation results on test sets of CLSum-UK. "N examples" denotes using N examples when fine-tuning models.

Method	CLSum-UK				
	0 examples	**10 examples**	**50 examples**	**100 examples**	**500 examples**
	R1 / R2 / RL	**R1 / R2 / RL**	**R1 / R2 / RL**	**R1 / R2 / RL**	**R1 / R2 / RL**
LexRank	60.28/26.86/22.84	-	-	-	-
TextRank	**60.62**/27.22/25.39	-	-	-	-
GPT3.5	57.05/25.51/24.10	-	-	-	-
LLaMA_{7B}	54.72/22.52/22.59	55.12/26.29/23.57	59.68/26.54/25.23	59.77/27.02/25.72	60.52/27.77/26.09
$\text{LLaMA}_{\text{13B}}$	52.80/21.87/22.23	60.70/27.38/25.47	59.61/26.37/25.51	60.83/**28.09**/26.26	61.26/28.54/**26.70**
$\text{Vicuna}_{\text{7B}}$	57.29/27.38/24.40	58.80/26.87/25.40	59.77/27.26/25.92	60.05/26.98/25.50	61.74/28.58/26.68
$\text{Vicuna}_{\text{13B}}$	57.86/**28.69**/**25.46**	60.00/**28.07**/**26.02**	**60.91**/**28.18**/**26.11**	60.42/28.01/**26.44**	61.05/28.65/26.46
LongT5	52.32/19.68/20.40	55.47/23.24/22.38	57.80/25.81/24.13	58.07/25.99/24.48	58.51/26.73/25.89
LED_{Base}	56.08/21.52/21.43	59.61/24.19/22.00	60.23/26.59/23.48	60.28/27.64/24.37	**62.06**/**29.40**/25.30
Legal-LED	37.46/10.07/17.05	**60.62**/25.56/23.45	60.71/26.55/23.78	**61.52**/27.59/24.49	61.62/28.97/25.67
$\text{LED}_{\text{Large}}$	49.02/17.91/20.85	59.33/24.28/22.55	60.78/26.79/24.05	61.42/27.78/24.58	61.78/28.90/26.28

(i.e., summary content is well organized and uses grammar appropriately), and non-redundancy (i.e., less repetition in output summary). We selected 30 samples at random from each CLSum subset's test set. For each sample, three annotators assess and compare the anonymously presented output summaries from two models. Additionally, we evaluate the agreement among annotators using Fleiss' kappa [39].

Table 6.8 exhibits our human evaluation results. Three models fine-tuned on each entire subset are compared here. The CLSum-CA and CLSum-HK subsets have very few samples in their training sets. On these two subsets, the Vicuna models perform worse than the LED model in terms of informativeness. Tables 6.1 and 6.2 present that target summaries' average length is shorter in the CLSum-CA subset. The shorter target summaries in the CLSum-CA comprise more new n-grams that are absent in the input and exhibit lower coverage and density of extractive fragments. Generating these more abstractive summaries can be difficult, particularly when the summarization model is fine-tuned on a very small training set. We discover that Vicuna models trained with the divide-and-conquer method on the CLSum-CA subset

Table 6.7: Automatic evaluation results on test sets of CLSum-AUS. "N examples" denotes using N examples when fine-tuning models.

Method	CLSum-AUS				
	0 examples	**10 examples**	**50 examples**	**100 examples**	**500 examples**
	R1 / R2 / RL	**R1 / R2 / RL**	**R1 / R2 / RL**	**R1 / R2 / RL**	**R1 / R2 / RL**
LexRank	53.57/24.46/24.24	-	-	-	-
TextRank	54.31/24.51/24.61	-	-	-	-
GPT3.5	54.10/25.51/25.11	-	-	-	-
LLaMA_{7B}	40.55/15.75/18.88	**58.07**/27.92/28.29	58.11/29.28/30.22	57.08/29.32/30.53	57.77/30.69/30.81
LLaMA_{13B}	43.41/17.44/19.74	57.73/27.61/27.85	57.37/**29.49**/**30.67**	59.19/**30.96**/**31.05**	59.32/31.60/**32.16**
Vicuna_{7B}	57.27/**27.53**/**27.03**	57.27/**29.65**/**29.20**	56.80/29.42/29.49	57.27/30.01/30.40	57.66/30.13/30.47
Vicuna_{13B}	**57.33**/27.36/26.75	55.03/27.45/27.27	55.70/28.58/29.52	56.41/29.12/29.95	57.73/30.41/30.95
LongT5	43.43/16.93/19.88	57.46/25.62/26.81	**60.02**/29.24/28.14	60.58/30.31/29.18	61.47/31.21/30.40
LED_{Base}	44.84/15.73/21.25	57.18/24.73/26.04	59.03/27.57/27.46	59.68/29.22/28.24	61.91/31.60/30.21
Legal-LED	42.28/11.08/19.78	57.97/25.72/26.63	59.84/28.85/28.16	60.00/29.37/28.52	61.86/31.74/30.24
$\text{LED}_{\text{Large}}$	43.64/15.57/20.40	56.73/24.25/26.41	59.29/27.81/28.42	**61.19**/29.90/29.22	**62.77**/**32.07**/31.05

generate more redundant and less informative summary content than the LED model. When there is a lack of training samples, the semantics of the generated summaries for different segments become relatively concentrated and exhibit more repeated content. Tables 6.1 and 6.2 also exhibit that the average length of target summaries is longer in the CLSum-HK subset. These longer target summaries' content is more diverse and less abstractive. When employing divide-and-conquer, semantically dispersed and longer target summaries for each segment will guide the summarization model to focus on different content when summarizing different segments. The lack of training samples in the CLSum-HK subset mainly affects informativeness and has less impact on the redundancy of generated summaries. On the CLSum-UK and CLSum-AUS subsets, the Vicuna models can outperform the LED model in informativeness, while these models are comparable regarding fluency and non-redundancy. These results verify that the training set size can affect the model acquiring the capacity to effectively summarize key information during fine-tuning, consequently influencing the informativeness of output summaries.

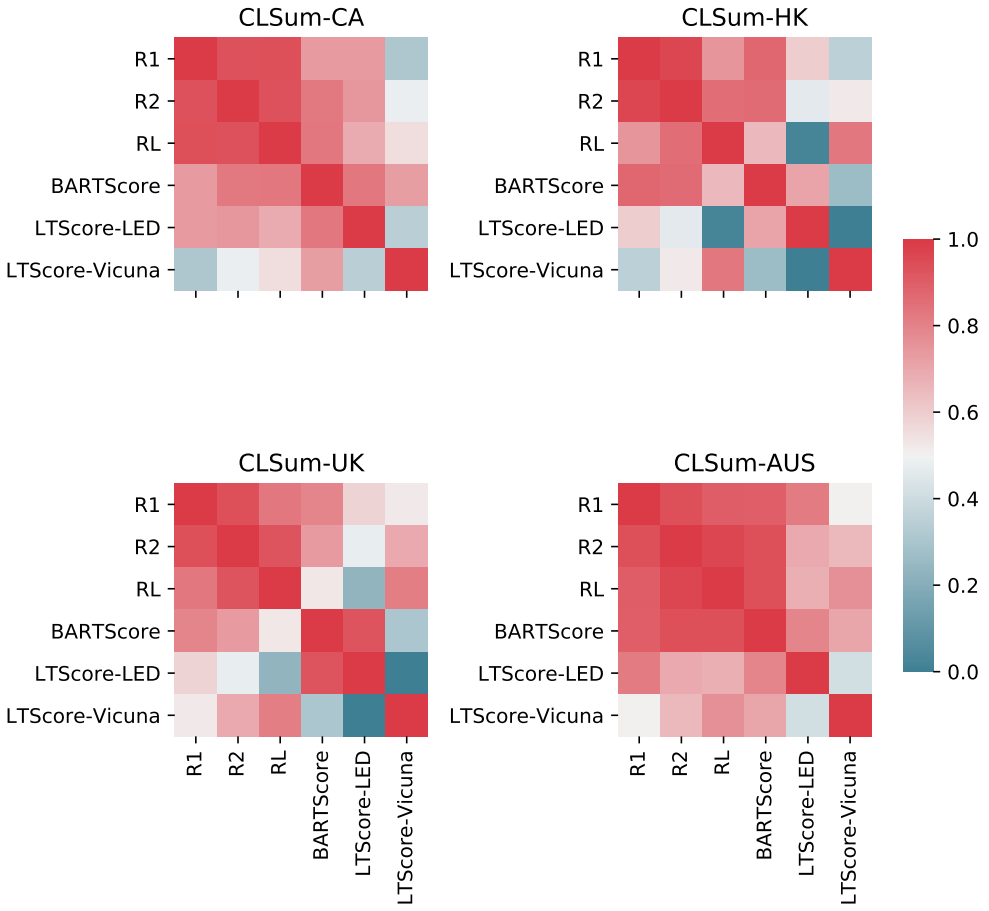


Figure 6.7: Correlation of automatic evaluation metrics.

Fig. 6.7 depicts the Pearson correlation among various evaluation metrics. The correlation among ROUGE scores on different N-grams is high. BARTScore and LTScore have a lower correlation with ROUGE scores. Introducing BARTScore and LTScore as supplements facilitates a more comprehensive evaluation of the generated results.

6.5.2 Discussion on the training set size

For models trained on the same training set, the larger models' few-shot performance is not always better than that of the smaller models. With the increase in the num-

ber of training samples, the performance of LLMs (LLaMA and Vicuna) improves slower than these smaller pre-trained sequence-to-sequence models (LongT5, LED, and Legal-LED). This may be caused by two reasons: 1) When adopting the QLoRA [31] technology, the number of trainable parameters is smaller than the entire model's parameter number. The small number of trainable parameters limits the new knowledge that the model can learn during fine-tuning; 2) Compared with the training data utilized in the pre-training stage and supervised fine-tuning (SFT) stage, the amount of labeled samples used in our fine-tuning process is relatively small, thereby having limited impact on model performance.

Finally, the performance of pre-trained models with hundreds of millions of parameters (LongT5, LED, and Legal-LED) can exceed that of a large language model with billions of parameters (LLaMA and Vicuna). Training smaller models with more labeled data can achieve comparable performance, which is critical to reducing the cost of deploying models in real-world applications.

6.5.3 Discussion on SFT and RLHF

The Vicuna models [17] fine-tuned with user-shared ChatGPT conversations largely surpass the original LLaMA models [129] in the zero-shot setting. Regarding the few-shot performance, SFT can assist LLMs in achieving commendable results by fine-tuning on a small set of labeled samples. Our experiment results verify the effectiveness of SFT.

We discover that GPT-3.5-turbo [13] fine-tuned with RLHF [92] has difficulty in generating case statements. The RLHF process trains the model to avoid generating illegal content. However, court judgment documents usually require an accurate statement of the parties' illegal facts, which are crucial bases for the judgment. RLHF used on general domains is not suitable for legal text generation. It requires a specially

[13]We adopt the GPT-3.5-turbo-0301 API from Azure Cloud

Table 6.8: Human evaluation results on CLSum dataset. "win" denotes that the current model's output summary surpasses that of $\text{LED}_{\text{Large}}$ model in one dimension.

	$\text{Vicuna}_{\text{13B}}$				$\text{Vicuna}_{\text{7B}}$			
	win	**lose**	**tie**	**kappa**	**win**	**lose**	**tie**	**kappa**
CLSum-CA								
Informativeness	23.3%	31.1%	45.6%	0.671	21.1%	26.7%	52.2%	0.618
Fluency	18.9%	21.1%	60.0%	0.623	17.8%	18.9%	63.3%	0.603
Non-Redundancy	27.8%	36.7%	35.6%	0.614	28.9%	33.3%	37.8%	0.648
CLSum-HK								
Informativeness	24.4%	28.9%	46.7%	0.635	26.7%	27.8%	45.6%	0.638
Fluency	20.0%	22.2%	57.8%	0.634	21.1%	24.4%	54.4%	0.629
Non-Redundancy	17.8%	18.9%	63.3%	0.624	18.9%	20.0%	61.1%	0.677
CLSum-UK								
Informativeness	37.8%	30.0%	32.2%	0.648	35.6%	31.1%	33.3%	0.649
Fluency	28.9%	25.6%	45.6%	0.655	26.7%	24.4%	48.9%	0.612
Non-Redundancy	17.7%	20.0%	62.2%	0.672	15.6%	16.7%	67.8%	0.636
CLSum-AUS								
Informativeness	30.0%	28.9%	41.1%	0.662	28.9%	27.8%	43.3%	0.625
Fluency	26.7%	24.4%	48.9%	0.647	23.3%	22.2%	54.4%	0.629
Non-Redundancy	25.6%	26.7%	47.8%	0.668	20.0%	22.2%	57.8%	0.615

designed RLHF process to adequately cater to the complex requirements in the legal field. Court judgment summarization models' outputs should accurately and objectively reflect the cases' facts and the court's decisions. There should be no factual or logical errors. Parties from different groups should be treated fairly.

Table 6.9: Evaluation results of summarization models trained on augmented datasets.

Method	Full Train Set	Rephrasing	Constrained Rephrasing	Back Translation
	R1 / R2 / RL	R1 / R2 / RL	R1 / R2 / RL	R1 / R2 / RL
CLSum-CA				
LLaMA-7B	47.91/18.10/20.74	46.41/16.79/21.24	52.17/19.46/22.39	39.91/12.06/21.18
LLaMA-13B	48.09/17.00/20.45	45.90/17.09/21.56	51.02/19.41/22.70	47.21/17.30/21.94
Vicuna-7B	47.62/17.36/22.05	47.92/17.47/**23.10**	52.45/19.65/22.80	43.22/14.92/21.48
Vicuna-13B	50.66/19.22/22.68	49.93/18.86/23.06	51.02/18.49/21.79	49.39/18.70/22.36
LongT5	55.85/19.98/21.48	55.01/19.88/21.73	55.31/20.18/21.84	55.62/19.93/21.70
LED-Base	54.57/19.63/21.32	53.28/19.57/21.39	54.94/20.08/22.10	52.75/19.29/20.80
Legal-LED	56.04/20.33/21.73	53.95/19.95/21.63	54.95/20.64/22.13	55.09/20.29/21.56
LED-Large	**57.23**/21.15/22.65	56.64/21.17/22.17	56.62/**21.72**/22.81	56.50/21.00/22.28
Average Improvement %		-2.14/-1.32/1.65	**2.82**/**4.82**/**3.29**	-4.62/-6.41/0.20
CLSum-HK				
LLaMA-7B	51.71/23.30/26.18	52.14/23.66/25.72	53.39/24.04/25.76	52.40/23.58/24.83
LLaMA-13B	52.21/23.99/26.06	53.15/24.76/26.24	53.53/24.67/26.83	53.06/23.84/26.06
Vicuna-7B	55.01/25.26/26.42	54.32/25.32/26.24	54.71/25.27/26.15	54.64/24.94/26.07
Vicuna-13B	55.07/26.18/26.78	55.30/25.96/26.98	56.31/26.45/**27.02**	54.87/25.14/26.17
LongT5	56.29/26.67/24.85	55.92/26.57/25.14	55.80/26.39/25.21	55.40/26.10/24.43
LED-Base	55.56/25.47/23.04	53.98/24.66/23.36	55.89/24.72/24.03	55.05/25.51/24.19
Legal-LED	56.10/25.50/23.74	55.46/25.82/24.20	56.12/25.36/24.85	55.88/26.10/24.13
LED-Large	56.43/26.49/24.92	56.69/**27.04**/25.37	**57.15**/26.44/25.52	55.75/25.85/24.41
Average Improvement %		-0.30/**0.49**/0.66	**1.06**/0.29/**1.76**	-0.27/-0.84/-0.73
CLSum-UK				
LLaMA-7B	60.68/27.65/26.04	60.83/28.29/26.68	60.47/27.14/25.90	60.37/27.37/26.05
LLaMA-13B	61.13/28.49/26.52	60.81/28.21/26.29	60.69/28.44/26.75	61.27/28.99/27.12
Vicuna-7B	61.42/29.04/26.83	61.58/29.28/27.10	61.63/29.46/27.30	61.53/28.78/26.77
Vicuna-13B	61.47/29.15/27.07	61.48/29.37/27.38	61.27/28.71/27.16	61.44/29.24/27.00
LongT5	59.62/28.08/26.64	59.76/27.81/26.22	60.27/28.62/26.86	60.54/29.23/26.93
LED-Base	62.18/28.92/25.91	**62.63**/29.52/26.58	62.41/30.81/27.41	61.61/28.49/26.02
Legal-LED	62.59/29.37/25.93	62.43/29.51/26.63	62.43/30.68/27.57	62.05/28.67/25.94
LED-Large	61.55/29.09/26.27	61.38/28.80/26.64	62.50/**31.16**/**28.17**	61.98/29.41/26.81
Average Improvement %		0.05/0.44/1.11	**0.21**/**2.24**/**2.82**	0.04/0.19/0.68
CLSum-AUS				
LLaMA-7B	56.24/28.56/28.71	55.87/28.92/29.38	57.27/26.07/26.39	56.78/28.93/29.29
LLaMA-13B	58.76/31.04/30.64	58.69/31.03/30.70	57.81/30.02/29.74	58.25/31.10/30.73
Vicuna-7B	57.95/30.46/30.77	58.18/27.92/27.64	57.84/30.10/30.70	57.17/29.88/30.33
Vicuna-13B	58.17/30.51/30.86	58.10/30.71/30.98	57.17/30.30/31.27	58.57/31.22/30.96
LongT5	61.99/31.82/31.55	60.90/30.51/30.31	61.57/31.52/31.18	60.89/30.77/30.74
LED-Base	62.65/32.38/30.92	61.75/31.63/30.52	**63.14**/**35.19**/**33.32**	62.40/32.14/30.84
Legal-LED	62.42/32.11/30.54	61.61/31.70/30.47	62.77/34.44/32.61	62.27/31.88/30.78
LED-Large	62.64/32.78/31.57	62.44/32.38/31.14	62.72/32.66/31.27	63.07/33.01/31.47
Average Improvement %		-0.66/-1.92/-1.76	**-0.11**/**0.09**/**0.31**	-0.29/-0.27/-0.14

Table 6.10: Effect of the amount of trainable parameters in the QLoRA adapter.

Dataset	Method	Rank=8	Rank=16	Rank=32
		R1 / R2 / RL	R1 / R2 / RL	R1 / R2 / RL
CLSum-CA	LLaMA-7B	47.91/18.10/20.74	40.35/13.40/20.93	42.04/13.09/21.49
	LLaMA-13B	48.09/17.00/20.45	48.54/14.42/19.16	46.86/16.50/20.91
	Vicuna-7B	47.62/17.36/22.05	48.26/17.47/22.49	46.22/16.13/22.39
	Vicuna-13B	**50.66**/**19.22**/22.68	49.66/19.04/22.29	50.44/19.14/**22.84**
CLSum-HK	LLaMA-7B	51.71/23.30/26.18	53.73/24.31/26.32	53.64/24.25/26.16
	LLaMA-13B	52.21/23.99/26.06	54.13/25.31/26.61	54.89/25.83/26.90
	Vicuna-7B	55.01/25.26/26.42	55.02/25.20/26.40	55.13/25.82/26.58
	Vicuna-13B	55.07/26.18/26.78	**55.82**/**27.25**/**27.54**	**55.82**/26.95/27.00
CLSum-UK	LLaMA-7B	60.68/27.65/26.04	61.04/27.88/26.05	60.55/27.96/26.21
	LLaMA-13B	61.13/28.49/26.52	60.75/28.19/26.34	60.52/28.14/26.03
	Vicuna-7B	61.42/29.04/26.83	61.45/28.44/26.56	61.19/28.80/26.47
	Vicuna-13B	**61.47**/**29.15**/**27.07**	61.30/28.42/26.71	60.93/28.66/26.81
CLSum-AUS	LLaMA-7B	56.24/28.56/28.71	56.21/28.61/28.95	56.32/28.55/28.82
	LLaMA-13B	58.76/31.04/30.64	**59.15**/**31.82**/**31.81**	58.62/31.35/31.60
	Vicuna-7B	57.95/30.46/30.77	56.98/30.14/30.35	57.45/30.34/31.00
	Vicuna-13B	58.17/30.51/30.86	57.19/30.09/30.94	57.88/30.56/30.94

6.5.4 Discussion on data augmentation methods

The performance of supervised models trained from scratch is typically constrained by the training set size. As introduced in subsection 6.3.1, we adopt and compare different data augmentation methods, including rephrasing, knowledge-constrained rephrasing, and back translation, to expand the training sets and reduce overfitting to the limited training samples. In our experiments, we doubled the training set size using each data augmentation method. Table 6.9 shows the impact of three data augmentation methods on summarization results. These data augmentation methods

Table 6.11: Details of summarization models.

Model	**Architecture**	**Params**	**Enc/Dec Layers**	**Heads**	$\mathbf{d_{model}}$	$\mathbf{d_{ff}}$	**Input Len**
LED_{base}	Enc-Dec	161.8M	6	12	768	3,072	16,384
$\text{LED}_{\text{large}}$	Enc-Dec	459.8M	12	16	1,024	4,096	16,384
Legal-LED	Enc-Dec	161.8M	6	12	768	3,072	16,384
$\text{LongT5}_{\text{base}}$	Enc-Dec	247.6M	12	12	768	2,048	16,384
LLaMA_{7B}	Dec Only	6.7B	32	32	4,096	11,008	2,048
$\text{LLaMA}_{\text{13B}}$	Dec Only	13.0B	40	40	5,120	13,824	2,048
$\text{Vicuna}_{\text{7B}}$	Dec Only	6.7B	32	32	4,096	11,008	2,048
$\text{Vicuna}_{\text{13B}}$	Dec Only	13.0B	40	40	5,120	13,824	2,048

bring different performance gains to summarization models trained on different subsets of CLSum. Experimental results verify that our proposed knowledge-constrained rephrasing method is helpful in the absence of labeled data. As shown in Table 6.1, CLSum-CA has the smallest training set. Data augmentation methods bring the most significant performance gain to summarization models trained on this subset. CLSum-AUS has the largest training set. Data augmentation methods bring marginal performance gain to models trained on that subset. This verifies that our data augmentation method primarily mitigates the impact of insufficient labeled data. The original rephrasing method and back translation method can benefit the ROUGE-L scores, but they often yield negative effects on ROUGE-1 and ROUGE-2 scores. Without the constraints of legal knowledge, there may be many errors in the data synthesized by data augmentation, which can adversely affect the summarization performance. Adding constraints in the rephrasing process can ensure the accurate use of legal terms in the synthesized data. This helps train models to accurately use relevant terms when generating judgment summaries.

6.5.5 Discussion on adapters' trainable parameters

The adapter is a small set of trainable parameters added to the large language models. We use the Low-rank Adapter (LoRA) [31, 55] to reduce the consumption of GPU memory when fine-tuning LLaMA and Vicuna models. As shown in Eq. 6.3, LoRA supplements the original linear projection $h = W_0x$ with an additional factorized projection. During training, $W_0 \in \mathbb{R}^{d\times k}$ remain unchanged, whereas $A \in \mathbb{R}^{r\times k}$ and $B \in \mathbb{R}^{d\times r}$, which have a rank $r \ll \min(d, k)$, comprise trainable parameters.

$$h = W_0x + \Delta Wx = W_0x + BAx \tag{6.3}$$

Table 6.10 shows the impact of LoRA's rank r on summarization results. The number of trainable parameters in the adapters expands as the rank r increases. Results show that increasing the LoRA's rank r does not necessarily improve the generated summaries. The primary constraint on the model performance stems from the inadequate quantity of training samples.

6.6 Chapter Summary

In this chapter, we introduce CLSum, a large-scale summarization dataset covering court judgments from four common law jurisdictions, including the United Kingdom, Canada, Australia, and Hong Kong SAR. Besides, we propose a foundation model-based solution for the low-resource court judgment summarization. We present a series of methods to deal with three challenges: 1) identifying the salient information scattered in the long document, 2) training supervised models with very limited labeled data, and 3) improving models' efficiency in processing long inputs and outputs. Additionally, we propose an evaluation metric named LTScore to assess the quality of the generated legal text. We benchmark advanced extractive and abstractive summarizers as baselines on our CLSum dataset. Our experimental results verify that the

foundation model-based summarization methods can perform well in the few-shot or zero-shot settings.

Chapter 7

Conclusions and Future Directions

7.1 Conclusions

In this thesis, I study neural abstractive summarization for long documents. When generalizing the summarization research from short documents to long documents, various new challenges arise. Firstly, the scarcity of large-scale datasets limits the long document summarization research. The labeled dataset is necessary for training and evaluating summarization methods. Secondly, it is difficult to completely identify and encode multi-granularity salient content scattered in a large amount of input content. Thirdly, integrating multi-document and multimodal salient content into the generated summaries is also challenging. Additionally, evaluating the quality of generated summaries from different aspects is also an important issue. Last but not least, improving the efficiency of model training and inference is challenging, especially when summarizing very long documents. To tackle the above challenges, I built multiple large-scale datasets, novel summarization methods, and evaluation metrics.

Specifically, I review existing work on neural abstractive summarization in Chapter 2. I first briefly introduce the taxonomies of existing document summarization work.

Then, I illustrate the development of neural abstractive summarization methods, including the RNN-based, transformer-based (trained from scratch), and pre-trained foundation model-based methods. I also introduce existing document summarization datasets.

In Chapter 3, I propose the key phrase aware transformer (KPAT), a lightweight model achieving great performance on multiple abstractive summarization tasks. This work focuses on enhancing the transformer encoder to completely encode the key phrases in input documents. I present the highlighting mechanism incorporating the prior knowledge of key phrases when calculating attention weights for tokens within key phrases.

In Chapter 4, I propose a new task named long text and multi-table summarization, which generalizes the long document summarization from unimodal (text) summarization to multimodal. Previous document summarization datasets and methods are usually restricted to summarizing the text content and excluding tables and figures from the input. In financial report documents, the key information can be distributed across both textual and non-textual content. The absence of tabular data can restrict the informativeness of generated summaries, particularly when summaries necessitate the quantitative descriptions of vital metrics within tables. Existing summarization methods and datasets fail to meet the demands of summarizing extensive textual and tabular content within financial reports. To deal with the scarcity of available datasets, this work builts FINDSum, the first large-scale dataset for long text and multi-table summarization. Besides, this work presents four types of summarization methods to jointly consider the text and table content when summarizing reports. Additionally, this work designs a set of evaluation metrics assessing the utilization of numerical information within the generated summaries.

In Chapter 5, I study how to summarize numerous academic papers about the same topic into a structured summary. Existing multi-document summarization (MDS) work usually focuses on producing an unstructured summary that encompasses only

a limited number of input documents. Meanwhile, previous structured summarization work focuses on summarizing a single document into a multi-aspect summary. Existing methods and datasets fail to fulfill the demands of summarizing numerous academic literature. This work builts BigSurvey, the first large-scale dataset for generating comprehensive summaries of numerous academic papers on each topic. Besides, this work proposes the category-based alignment and sparse transformer (CAST) to effectively arrange the diverse content from a large number of input documents while simultaneously ensuring efficiency when processing long inputs.

Finally, I illustrate our work on low-resource court judgment summarization. Judges in common law systems need to find similar precedents in all common law jurisdictions and refer to the reasoning in previous decisions. There exist hundreds of thousands of reported cases in common law jurisdictions, and the number of cases is still increasing. It can be challenging for legal practitioners to read through abundant cases' judgment documents. This work aims to let the computer generate high-quality court judgment summaries, which can help readers quickly browse key information in long judgment documents. To deal with the scarcity of available datasets, this work builts CLSum, the first large-scale dataset for summarizing common law court judgment documents from multiple jurisdictions. Like other domain-specific tasks, court judgment summarization usually suffers from the shortage of labeled samples. To address this challenge, this work proposes a foundation model-based solution for the low-resource court judgment summarization. To the best of our knowledge, this work is the first to employ large language models for data augmentation, summary generation, and evaluation in court judgment summarization. Additionally, this work designs an evaluation metric named LTScore to assess the quality of the generated legal text.

7.2 Future Directions

Although this thesis presents several novel datasets, methods, and evaluation metrics for long document summarization, there are still several open problems to be addressed in the future. I list the open problems and future directions as follows:

- Controllability of abstractive summarization models: Ensuring controllability of text generation is a pivotal and fundamental research problem in the natural language generation [148]. It requires text generation models to reliably generate text that conforms to users' specific requirements or constraints. Currently, all mainstream abstractive summarization models still have controllability problems. Although the RLHF technique [94] improves the alignment between user instructions and model outputs [124], the models trained with RLHF (e.g., ChatGPT) still struggle to follow some simple constraints, like the length of the generated text, not to mention more complex and multifaceted user requirements. There is still a lot of work to be done to improve the controllability of the abstractive summarization models.

- Generalizing from monolingual to multilingual summarization: Current long document summarization research mainly focuses on monolingual documents. However, valuable content about an object may come from documents in different languages. In the context of globalization, generalizing long document summarization research from monolingual to multilingual is very valuable. It can be of great help to many transnational studies and cooperation.

- Generalizing from unimodal to multimodal summarization: Many long documents contain multimodal content. Current document summarization work usually filters out non-textual content. Missing non-textual content can limit produced summaries' informativeness, especially when some critical information only appears in non-textual content. Multimodal long document summariza-

tion would be a promising research direction. There is still a lot of work to be done to efficiently integrate multimodal content into a text summary.

- Generalizing from closed-domain to open-domain summarization: Previous summarization research usually focuses on closed domains. Commonly used summarization datasets are usually collected from a single or limited source. In some real-world applications, people need to summarize numerous documents from diverse sources. These documents can follow diverse formats and writing styles. Adapting summarization models to various documents in the open domain deserves further study.

- Better evaluation metrics for generated summaries: Many studies found that the results of existing commonly used evaluation metrics (e.g., ROUGE [67]) are not always consistent with human preferences [120, 124]. Designing more effective automatic evaluation metrics is a valuable research direction. When the quality of model-generated text is close to that of human writing, further research on the annotation process is needed to ensure human evaluation's objectivity, accuracy, and consistency.

- More efficient neural models for processing long inputs and outputs: Neural summarization models' training and inference efficiency is very important when deploying them to real-world applications. Adopting neural summarization models with lower complexity together with more efficient training techniques can substantially reduce cost while maintaining comparable performance. Lighter and more effective summarization models are worth further exploration.

References

[1] Waleed Ammar, Dirk Groeneveld, et al. Construction of the literature graph in semantic scholar. In *NAACL-HLT*, 2018.

[2] Shir Aviv-Reuven and Ariel Rosenfeld. Publication patterns' changes due to the covid-19 pandemic: A longitudinal and short-term scientometric analysis. *Scientometrics*, pages 1–24, 2021.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[5] Ahsaas Bajaj, Pavitra Dangati, Kalpesh Krishna, Pradhiksha Ashok Kumar, Rheeya Uppaal, Bradford Windsor, Eliot Brenner, Dominic Dotterrer, Rajarshi Das, and Andrew McCallum. Long document summarization in a low resource setting using pretrained language models. In *Proc. ACL-SRW*, pages 71–80, 2021.

[6] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[7] Ramesh Chandra Belwal, Sawan Rai, and Atul Gupta. Text summarization

using topic-based vector space model and semantic measure. *Information Processing & Management*, 58(3):102536, 2021.

[8] Ahmet Benzer, Ayşegül Sefer, Zeyneb Ören, and Sümeyye Konuk. A student-focused study: Strategy of text summary writing and assessment rubric. *Education & Science/Egitim ve Bilim*, 41(186), 2016.

[9] Paheli Bhattacharya, Kaustubh Hiware, Subham Rajgaria, Nilay Pochhi, Kripabandhu Ghosh, and Saptarshi Ghosh. A comparative study of summarization algorithms applied to legal case judgments. In *Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part I 41*, pages 413–428. Springer, 2019.

[10] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[11] Adrien Bougouin, Florian Boudin, and Béatrice Daille. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International joint conference on natural language processing (IJCNLP)*, pages 543–551, 2013.

[12] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[13] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[14] Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel S Weld. Tldr: Extreme summarization of scientific documents. In *Proceedings of the 2020 Conference*

on Empirical Methods in Natural Language Processing: Findings, pages 4766–4777, 2020.

[15] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proc. KDD*, pages 785–794, 2016.

[16] Jianpeng Cheng and Mirella Lapata. 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, 2016.

[17] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *https://lmsys.org/blog/2023-03-30-vicuna/*, 2023.

[18] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1724. Association for Computational Linguistics, 2014.

[19] S. Chopra, M. Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *HLT-NAACL*, 2016.

[20] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2020.

[21] Mu-hsuan Chou. Implementing keyword and question generation approaches in teaching efl summary writing. *English Language Teaching*, 5(12):36–41, 2012.

[22] Hans Christian, Mikhael Pramodana Agus, and Derwin Suhartono. Single document automatic text summarization using term frequency-inverse document frequency (tf-idf). *ComTech: Computer, Mathematics and Engineering Applications*, 7(4):285–294, 2016.

[23] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of NAACL-HLT*, pages 615–621, 2018.

[24] Arman Cohan, Iz Beltagy, King, et al. Pretrained language models for sequential sentence classification. In *EMNLP-IJCNLP*, pages 3693–3699, 2019.

[25] Edward Collins, Isabelle Augenstein, and Sebastian Riedel. A supervised approach to extractive summarisation of scientific papers. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 195–205, 2017.

[26] Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, et al. Auggpt: Leveraging chatgpt for text data augmentation. *arXiv preprint arXiv:2302.13007*, 2023.

[27] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[28] Diego de Vargas Feijó and Viviane Pereira Moreira. Rulingbr: A summarization dataset for legal texts. In *Computational Processing of the Portuguese Language: 13th International Conference, PROPOR 2018, Canela, Brazil, September 24–26, 2018, Proceedings 13*, pages 255–264. Springer, 2018.

[29] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.

int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

[30] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *9th International Conference on Learning Representations, ICLR*, 2022.

[31] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *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, 2019.

[33] HKSAR DOJ. The common law and rules of equity. URL `https://www.doj.gov.hk/en/our_legal_system/the_common_law.html`.

[34] Qiang Du. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. In *IJCAI*, 2018.

[35] Vladimir Eidelman. Billsum: A corpus for automatic summarization of us legislation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 48–56, 2019.

[36] Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. Edgesumm: Graph-based framework for automatic text summarization. *Information Processing & Management*, 57(6):102264, 2020.

[37] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22: 457–479, 2004.

[38] Alexander Richard Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, 2019.

[39] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[40] Corina Florescu and Cornelia Caragea. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115, 2017.

[41] Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, 2018.

[42] Alexios Gidiotis and Grigorios Tsoumakas. A divide-and-conquer approach to the summarization of long documents. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:3029–3040, 2020.

[43] Ingo Glaser, Sebastian Moser, and Florian Matthes. Summarization of german court rulings. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 180–189, 2021.

[44] Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. 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 708–719, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1065. URL `https://www.aclweb.org/anthology/N18-1065`.

[45] Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. LongT5: Efficient text-to-text transformer for long sequences. In *Findings of NAACL*, pages 724–736, 2022. doi: 10.18653/v1/2022.findings-naacl.55.

[46] Som Gupta and SK Gupta. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 121:49–65, 2019.

[47] Sandra Hargreaves and Jamie Crabb. *Study Skills for Students with Dyslexia: Support for Specific Learning Differences (SpLDs)*. Sage, 2016.

[48] James Hartley. Current findings from research on structured abstracts. *Journal of the Medical Library Association*, page 368, 2004.

[49] James Hartley. Current findings from research on structured abstracts: An update. *Journal of the Medical Library Association*, pages 146–148, 2014.

[50] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE Computer Society, 2016.

[52] Karl Moritz Hermann, Tomáš Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1693–1701, 2015.

[53] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[54] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *Proc. ICLR*, 2019.

[55] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

[56] Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 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, June 2021. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2021.naacl-main.112`.

[57] Hanqi Jin, Tianming Wang, and Xiaojun Wan. Multi-granularity interaction network for extractive and abstractive multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6244–6254, 2020.

[58] Ambedkar Kanapala, Sukomal Pal, and Rajendra Pamula. Text summarization from legal documents: a survey. *Artificial Intelligence Review*, 51:371–402, 2019.

[59] Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. Abstractive summarization of Reddit posts with multi-level memory networks. In *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 2519–2531, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1260. URL `https://www.aclweb.org/anthology/N19-1260`.

[60] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[61] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M

Rush. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, 2017.

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

[63] Logan Lebanoff, Kaiqiang Song, and Fei Liu. Adapting the neural encoder-decoder framework from single to multi-document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4131–4141, 2018.

[64] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

[65] Wei Li, Xinyan Xiao, Jiachen Liu, Hua Wu, Haifeng Wang, and Junping Du. Leveraging graph to improve abstractive multi-document summarization. *arXiv preprint arXiv:2005.10043*, 2020.

[66] Zeyu Liang, Junping Du, Yingxia Shao, and Houye Ji. Gated graph neural attention networks for abstractive summarization. *Neurocomputing*, 431:128–136, 2021.

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

[68] Hui Lin and Vincent Ng. Abstractive summarization: A survey of the state of the art. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9815–9822, 2019.

[69] Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. Global encoding for abstractive summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 163–169, 2018.

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

[71] Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. Tapex: Table pre-training via learning a neural sql executor. In *Proc. ICLR*, 2021.

[72] Shuaiqi Liu, Jiannong Cao, Ruosong Yang, and Zhiyuan Wen. Highlight-transformer: Leveraging key phrase aware attention to improve abstractive multi-document summarization. In *Findings of ACL*, pages 5021–5027, 2021. doi: 10.18653/v1/2021.findings-acl.445.

[73] Shuaiqi Liu, Jiannong Cao, Ruosong Yang, and Zhiyuan Wen. Generating a structured summary of numerous academic papers: Dataset and method. In *Proc. IJCAI*, pages 4259–4265, 2022. doi: 10.24963/ijcai.2022/591.

[74] Shuaiqi Liu, Jiannong Cao, Ruosong Yang, and Zhiyuan Wen. Key phrase aware transformer for abstractive summarization. *Information Processing & Management*, 59(3):102913, 2022.

[75] Shuaiqi Liu, Jiannong Cao, Ruosong Yang, and Zhiyuan Wen. Long text and multi-table summarization: Dataset and method. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1995–2010, 2022.

[76] Shuaiqi Liu, Jiannong Cao, Zhongfen Deng, Wenting Zhao, Ruosong Yang, Zhiyuan Wen, and S Yu Philip. Neural abstractive summarization for long text and multiple tables. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[77] Yang Liu and Mirella Lapata. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, 2019.

[78] Yang Liu and Mirella Lapata. 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 3721–3731, 2019.

[79] Yinhan Liu, Myle Ott, Naman Goyal, et al. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[80] Yizhu Liu, Qi Jia, and Kenny Zhu. Keyword-aware abstractive summarization by extracting set-level intermediate summaries. In *Proceedings of the Web Conference 2021*, pages 3042–3054, 2021.

[81] Yao Lu, Yue Dong, and Laurent Charlin. Multi-XScience: A large-scale dataset for extreme multi-document summarization of scientific articles. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8068–8074, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.648. URL `https://www.aclweb.org/anthology/2020.emnlp-main.648`.

[82] Duy Khang Ly, Kazunari Sugiyama, Ziheng Lin, and Min-Yen Kan. Product review summarization from a deeper perspective. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pages 311–314, 2011.

[83] Rui Meng, Khushboo Thaker, Lei Zhang, Yue Dong, Xingdi Yuan, Tong Wang, and Daqing He. Bringing structure into summaries: a faceted summarization dataset for long scientific documents. *arXiv preprint arXiv:2106.00130*, 2021.

[84] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004.

[85] Begum Mutlu, Ebru A Sezer, and M Ali Akcayol. Candidate sentence selection for extractive text summarization. *Information Processing & Management*, 57 (6):102359, 2020.

[86] Ramesh Nallapati, Bowen Zhou, C. D. Santos, Çaglar Gülçehre, and B. Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*, 2016.

[87] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[88] Courtney Napoles, Matthew R Gormley, and Benjamin Van Durme. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100, 2012.

[89] NARA. Regulation s-k item 303 management's discussion and analysis of financial condition and results of operations. URL `https://www.ecfr.gov/current/title-17/chapter-II/part-229`.

[90] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1206. URL `https://www.aclweb.org/anthology/D18-1206`.

[91] Shashi Narayan, Shay B Cohen, and Mirella Lapata. Ranking sentences for extractive summarization with reinforcement learning. 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 1747–1759, 2018.

[92] OpenAI. Introducing chatgpt. *https://openai.com/blog/chatgpt*, 2022.

[93] Myle Ott et al. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL-HLT*, 2019.

[94] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[95] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[96] Amanda Parker, Edward Wilding, and Colin Akerman. The von restorff effect in visual object recognition memory in humans and monkeys: The role of frontal/perirhinal interaction. *Journal of cognitive neuroscience*, 10(6):691–703, 1998.

[97] Romain Paulus, Caiming Xiong, and R. Socher. A deep reinforced model for abstractive summarization. *ArXiv*, abs/1705.04304, 2018.

[98] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.

[99] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proc. EMNLP*, pages 1532–1543, 2014.

[100] Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Christopher Pal. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9308–9319, 2020.

[101] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

[102] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

[103] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[104] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

[105] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *Proc. ICLR*, 2016.

[106] Luz Rello, Horacio Saggion, and Ricardo Baeza-Yates. Keyword highlighting improves comprehension for people with dyslexia. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 30–37, 2014.

[107] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280, 2020.

[108] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1044. URL `https://www.aclweb.org/anthology/D15-1044`.

[109] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, 2015.

[110] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[111] Evan Sandhaus. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752, 2008.

[112] Yogesh Sankarasubramaniam, Krishnan Ramanathan, and Subhankar Ghosh. Text summarization using wikipedia. *Information Processing & Management*, 50(3):443–461, 2014.

[113] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[114] United States SEC. Form 10-k general instructions. URL `https://www.sec.gov/about/forms/form10-k.pdf`.

[115] United States SEC. How to read a 10-k/10-q, January 2021. URL `https://www.sec.gov/fast-answers/answersreada10khtm.html`.

[116] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, 2017.

[117] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proc. ACL*, pages 86–96, 2016.

[118] Eva Sharma, Chen Li, and Lu Wang. Bigpatent: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213, 2019.

[119] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

[120] Abhay Shukla, Paheli Bhattacharya, Soham Poddar, Rajdeep Mukherjee, Kripabandhu Ghosh, Pawan Goyal, and Saptarshi Ghosh. Legal case document summarization: Extractive and abstractive methods and their evaluation. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, pages 1048–1064, 2022.

[121] Arnab Sinha, Zhihong Shen, Yang Song, et al. An overview of microsoft academic service (mas) and applications. In *WWW*, pages 243–246, 2015.

[122] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926–5936. PMLR, 2019.

[123] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[124] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[125] Ming-Hsiang Su, Chung-Hsien Wu, and Hao-Tse Cheng. A two-stage transformer-based approach for variable-length abstractive summarization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2061–2072, 2020.

[126] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[127] Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181, 2017.

[128] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 2023.

[129] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro,

Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

[131] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[132] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, 2019.

[133] Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. Tl; dr: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, 2017.

[134] Hedwig Von Restorff. Über die wirkung von bereichsbildungen im spurenfeld. *Psychologische Forschung*, 18(1):299–342, 1933.

[135] Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuan-Jing Huang. Heterogeneous graph neural networks for extractive document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6209–6219, 2020.

[136] Tianxin Wang, Jingwu Chen, Fuzhen Zhuang, Leyu Lin, Feng Xia, Lihuan Du, and Qing He. Capturing attraction distribution: Sequential attentive network for dwell time prediction. In *ECAI 2020*, pages 529–536. IOS Press, 2020.

[137] Xun Wang, Masaaki Nishino, Tsutomu Hirao, Katsuhito Sudoh, and Masaaki Nagata. Exploring text links for coherent multi-document summarization. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 213–223, 2016.

[138] Thomas Wolf, Julien Chaumond, Lysandre Debut, et al. Transformers: State-of-the-art natural language processing. In *EMNLP*, pages 38–45, 2020.

[139] Xiaoxia Wu, Cheng Li, Reza Yazdani Aminabadi, Zhewei Yao, and Yuxiong He. Understanding int4 quantization for transformer models: Latency speedup, composability, and failure cases. *arXiv preprint arXiv:2301.12017*, 2023.

[140] Wen Xiao, Patrick Huber, and Giuseppe Carenini. Do we really need that many parameters in transformer for extractive summarization? discourse can help! In *Proceedings of the First Workshop on Computational Approaches to Discourse*, pages 124–134, 2020.

[141] Song Xu, Haoran Li, Peng Yuan, Youzheng Wu, Xiaodong He, and Bowen Zhou. Self-attention guided copy mechanism for abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1355–1362, 2020.

[142] Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. Recent advances in document summarization. *Knowledge and Information Systems*, 53(2):297–336, 2017.

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

[144] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[145] Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277, 2021.

[146] Carole L Yue, Benjamin C Storm, Nate Kornell, and Elizabeth Ligon Bjork. Highlighting and its relation to distributed study and students' metacognitive beliefs. *Educational Psychology Review*, 27(1):69–78, 2015.

[147] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020.

[148] Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. A survey of controllable text generation using transformer-based pre-trained language models. *arXiv preprint arXiv:2201.05337*, 2022.

[149] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020.

[150] Mengli Zhang, Gang Zhou, Wanting Yu, and Wenfen Liu. Far-ass: Fact-aware reinforced abstractive sentence summarization. *Information Processing & Management*, 58(3):102478, 2021.

[151] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[152] Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuan-Jing Huang. Searching for effective neural extractive summarization: What works and what's next. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1049–1058, 2019.

[153] Li Zhuang, Feng Jing, and Xiao-Yan Zhu. Movie review mining and summarization. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 43–50, 2006.