ABSTRACTIVE TEXT SUMMARIZATION BASED ON NEURAL FUSION

WENZHAO ZHU
Bachelor of Science, Beijing University of Technology, 2021

A thesis submitted
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

ABSTRACTIVE TEXT SUMMARIZATION BASED ON NEURAL FUSION

WENZHAO ZHU

Date of Defence: December 11th, 2023

Dr. Chali Thesis Supervisor	Professor	Ph.D.
Dr. Anvik Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. Zhang Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. Osborn Chair, Thesis Examination Committee	Associate Professor	Ph.D.

Abstract

Abstractive text summarization, in comparison to extractive text summarization, offers the potential to generate more accurate summaries. In our work, we present a stage-wise abstractive text summarization model that incorporates Elementary Discourse Unit (EDU) segmentation, EDU selection, and EDU fusion. We first segment the articles into a fine-grained form, EDUs, and build a Rhetorical Structure Theory (RST) tree for each article in order to represent the dependencies among EDUs; those EDUs are encoded in Graph Attention Networks (GATs); those with higher importance will be selected as candidates to be fused and the fusing stage is done by Bidirectional and Auto-Regressive Transformers (BART) model which merges the selected EDUs into summaries. A Greedy Method is leveraged to greedily select those EDUs whose combinations can maximize the ROUGE scores. Our model outperforms the baseline of BART (large) on the CNN/Daily Mail dataset, showing its effectiveness in abstractive text summarization.

Acknowledgments

First, I appreciate the patience and instructions of my supervisor, Dr. Yllias Chali. He gave me a chance to pursue my Master's degree. I am very grateful for the resource support from him.

Also, I appreciate the feedback and suggestions of all the committee members, Dr. John Anvik, and Dr. John Zhang.

I would also like to thank my workmates and friends for the enlightenment and support they have provided. They make me feel less lonely on the way of my research.

I am also grateful to all the staff in the Department of Computer Science for their guidance and support.

Finally, my thanks go to the authors of all the referenced papers and the contributors to the GitHub repositories I utilized.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Text summarization, an important task in natural language processing, finds widespread applications in people's daily lives. It is employed for tasks such as extracting news summaries, summarizing extensive document collections, processing medical case records, generating business reports, and so on. The primary approach to text summarization is distilling key information from diverse redundant and unstructured sources, thereby enhancing the efficiency of information acquisition and enabling rapid and accurate comprehension. For example, the summarization of chapters in a book serves as a quick reference for readers This enables them to discern which sections of the book align with their interests and increase time efficiency.

Text summarization can be broadly categorized into two types: extractive summarization and abstractive summarization, depending on whether it reproduces content directly from the source text or produces novel content. An example of extractive summarization and abstractive summarization is provided in Table 1.1.

The abstractive summarization approaches exhibit reduced reliance on the copied content, leading to the generation of summaries that are closer to human-annotated ones, so it is more likely to establish a coherent context.

Neural networks play an important role in our work. They are a type of computing model inspired by how the human neural system works. It consists of a certain number of 'neurons' that connect through weights. Usually, there is more than one layer of 'neurons'. The input signals are analyzed and feature extraction is performed by neurons in a particular

Table 1.1: An example of extractive and abstractive summarization.

	Article	Summary
Extractive	**people's content technology application layer can be divided into four parts, the first is content originality**, although the media industry ... **the second is content wind control**, ... **the third is content aggregation and distribution**, ... **and finally content operation**...good interaction with the public and users.	**people's content technology application layer can be divided into four parts, the first is content originality, the second is content wind control, the third is content aggregation and distribution, and finally content operation**.
Abstractive	prison link cymru had 1,099 referrals in 2015-16 and said some ex-offenders were living rough for up to a year before finding suitable accommodation ... changes to the **housing** act **in wales**, introduced in 2015, removed the right **for prison leavers** to be given priority for accommodation ...	there is a "chronic" need for more **housing for prison leavers in wales**, according to a charity.

layer through the weights between every two nodes. Those signals are then passed to the next layer. This process continues iteratively until the final layer outputs corresponding features or labels. Through training, neural networks can acquire the knowledge of how to update weights in order to achieve the desired output. In practice, neural networks have reached outstanding results in most of the natural language processing tasks including text summarization. Leveraging the remarkable capabilities of neural networks, several state-of-the-art abstractive summarization techniques rely on them, such as pointer generator networks (See et al., 2017), the Transformers (Vaswani et al., 2017) and most of the derived structures of the Transformers (Lewis et al., 2020; Devlin et al., 2019; Radford et al., 2018).

Nevertheless, a majority of the existing methodologies either treat the entire text se-

quences (See et al., 2017; Lewis et al., 2020) or salient sentences (Chen and Bansal, 2018) as input, introducing excessive noise during the summarization process. To address the problem of the unexpected noise that the input sequences may bring to the predicted abstracts, we propose employing fine-grained units, EDUs, as the foundational units to generate summaries. We only take the concatenation of those informative EDUs as the input to be fused to avoid most of the redundant content. Therefore, how to truncate the whole text sequences into fine-grained units and selecting those important ones are the two necessary steps before starting to fuse them.

Our experiments are conducted on the CNN/Daily Mail (Hermann et al., 2015) dataset, primarily due to the copyright restrictions and the quality of some of the text summarization datasets like New York Times (NYT) (Consortium and Company, 2008) and XSum (Narayan et al., 2018). Also, the required time and resource investment for the creation of a reliable, high-quality dataset further exceeds our ability. Additionally, our work only focuses on short documents (sequences that are less than 2048 tokens) since long document processing requires a higher hardware capacity.

To transform the original articles into groups of EDUs, we first followed the guidelines provided by Fairseq (Ott et al., 2019) to convert the raw text files into formatted documents, facilitating subsequent processing steps. We then deployed the segmentation tool proposed by Wang et al. (2018) to further convert the preprocessed documents of articles into EDU lists. In the selecting stage, we first use the method provided by Ji and Eisenstein (2014) to establish the graphs that illustrate the relations between every two EDUs. Then we design a graph neural network-based (Veličković et al., 2017) model, taking the graphs as inputs to classify the EDUs as '0' (unimportant) and '1' (important). Finally, we use BART (Lewis et al., 2020) to fuse those EDUs that are labeled with '1' to increase the coherence of the outputs. The details of the implementations will be elaborated in the corresponding chapters.

Essentially, our method optimizes the data for fusing. Specifically, it converts the orig-

inal long sequences into lists of informative units, greatly reducing the hardware requirements of the training device. Also, the performance of the fusing model reaches a higher level with an optimization of the datasets. Additionally, our model can be used on other types of text that have close contextualizations, such as academic papers and novel chapters. In this way, our method can help people have a quick and accurate understanding of the text within a certain length.

In the upcoming chapter, a background introduction will be provided. This will commence with an exploration of the recent techniques pertaining to Natural Language Processing (NLP). It will encompass an introduction to the identification of high-quality neural language models and some linguistic theories. Also, an introduction to the training process and evaluating metrics will be given in the next chapter.

Following the next chapter, in Chapter 3, we introduce the literature review of our research. In Chapter **??**, an introduction to the dataset and the methodology for each stage (segmentation, selection, fusion) will be given separately. In Chapter 5, the implementation details and the analysis of the experiment results will be given, which offer a comprehensive understanding of the contributions and findings of our research. Finally, a conclusion of our study and some perspectives for future work will be presented in Chapter 6.

Our main contribution lies in the utilization of graph attention networks to select important EDUs and fuse them. This approach substantially reduces input redundancy for BART, resulting in an improved overall segmentation-selection-fusion process that outperforms the baseline of BART.

Chapter 2

Background

Many previous techniques were used in our work. To have a good comprehension of the general framework of our work, it is necessary to have an understanding of the conceptions and methodologies we leveraged.

2.1 Rhetorical Structure Theory

Rhetorical Structure Theory (Mann and Thompson, 1988) is a linguistic theory and analytical framework aimed at studying and describing the rhetorical and organizational structure within texts. It focuses on the organization of information, logical relationships, and semantic connections within texts, emphasizing the interaction between linguistic units, EDUs, and how they form a coherent and cohesive text. RST contributes to enhancing the ability to achieve a profound comprehension of texts and facilitates their processing.

2.1.1 Elementary Discourse Unit

The basic unit for an RST tree (Section 2.1.2) is the Elementary Discourse Unit (EDU). An EDU is the smallest unit used to represent the structure of a text and usually consists of one or more neighboring sentences or phrases that form a separate linguistic unit within the discourse. Each EDU has a certain level of syntactic and semantic integrity, i.e., it can be understood to some extent independently of the other parts of the text. In text-level linguistic analysis, EDUs are often used to represent the relationships and structures between paragraphs, sentences, and phrases of a text. Table 2.1 is an example of how a

sentence can be segmented into several EDUs. In the table, the sequences with the suffix $e_1 - e_5$ are the EDUs.

Table 2.1: An example of segmentation.

[“He is the best actor]$_{e1}$[I have ever known,”]$_{e2}$[said Ben,]$_{e3}$[who played mechanic cooter on the show]$_{e4}$[in a Facebook post.]$_{e5}$

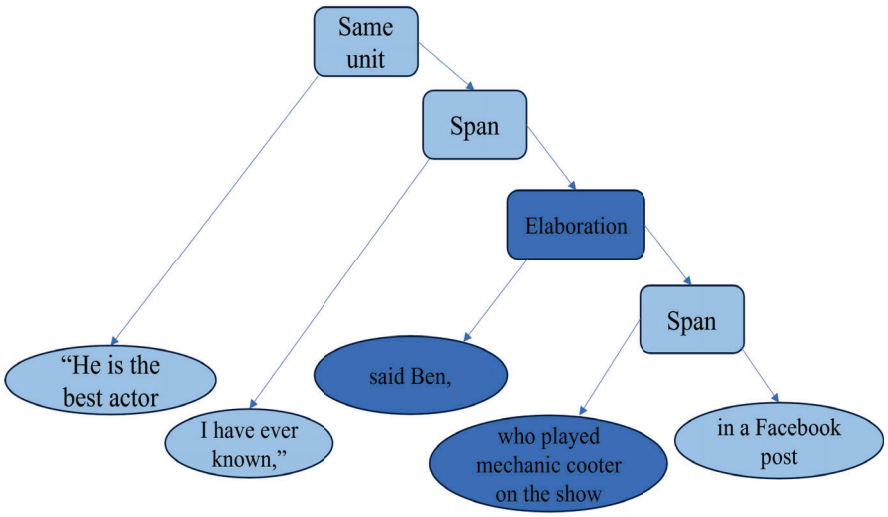
In natural language processing, the concept of EDUs is widely used in tasks such as text segmentation, chapter structure analysis, summary generation, and dialog systems. By identifying and understanding the relationships between EDUs, the organization of information and contextual relationships in text can be better understood and processed, which helps improve the performance of text comprehension and generation tasks.

2.1.2 RST Tree

Typically, RST employs a tree structure, known as an RST tree. An illustration of the RST structure is provided in Figure 2.1. The intermediate nodes signify the discourse relations between two EDUs, such as elaboration, condition, cause-effect, contrast, etc. Based on different types of relations and the semantic information of the EDUs themselves, it is possible to predict whether the EDUs are pivotal or not. Additionally, EDUs are categorized into two different nuclearities: satellite and nucleus, which hinge upon their degree of importance within the discourse structure. An RST tree represents the relations between each of the two EDUs within articles in the form of a graph structure. The terminal nodes of the RST graph correspond to the EDUs present in an article, while the non-terminal nodes signify the relations between EDUs. In Figure 2.1, “Same unit”, “Elaboration”, and “Span” indicate the relations between each of the two EDUs. The terminal nodes “He is the best actor”, “I have ever known”, “said Ben,”, “who played mechanic cooter on the show”, and “in a Facebook post.” are EDUs.

Due to its distinctive properties, RST trees are capable of capturing the importance of their constituent nodes. The original RST-based corpus, RST Discourse Treebank (RST-

Figure 2.1: The structure of an RST tree.



DT) (Carlson et al., 2001), is human-annotated and is used in EDU segmentation models (Wang et al., 2018) for the purpose of training. Some of the generated RST-based datasets are based on machine learning methods (e.g., Discourse Parsing from Linear Projection (DPLP) (Ji and Eisenstein, 2014)). RST trees, as graph-based datasets, are widely used in graph analysis models such as Convolutional Neural Networks and Graph Neural Networks.

2.2 Large Language Models

In the context of this thesis, large language models (LLMs) play an important role in both EDU selection and EDU fusing. Therefore, understanding the capabilities and limitations of these models is crucial for our research. A large language model is an AI model designed to understand and generate human language. More precisely, they are trained on a large amount of data to learn the patterns of the data so they can make predictions in a more accurate manner. For example, large language models like GPT-2 and GPT-3 developed by OpenAI[1] have been widely used for natural language processing tasks such as question-answering tasks and sentiment classification tasks.

Usually, large language models have more than one billion parameters (parameters are

[1]Official website: https://openai.com/.

the variables that are presented in a model to be trained and to make predictions). Classic LLMs like GPT-2 and GPT-3 have 1.5 billion and 175 billion parameters, respectively.

2.2.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) is one of the LLMs we employed in our experiments for encoding text into vectors. BERT is a Transformer-based (Vaswani et al., 2017) LLM that was previously proved to have an outstanding ability for natural language understanding. BERT has around 110 million parameters in its base version and 340 million parameters in its large version. There are two tasks in the training stage of BERT, *Masked Language Model(MLM)* and *Next Sentence Prediction(NSP)*. The first training task requires BERT to predict the masked tokens given the sequences that have some of the tokens masked. This allows BERT to learn the contextual meaning of a word by aggregating the information from the context. The NSP task requires BERT to understand the relationship between two sentences. Since the BERT model was pre-trained on these two tasks, it has an excellent understanding of the input of human language text. Hence, BERT can distill the key information of a sequence and convert text into vectors with rich features.

2.2.2 GPT

GPT is a series of large language models that were developed by OpenAI. All the GPT models (including GPT, GPT-2, GPT-3, etc.) have the Transformer-based architecture. They are all composed of stacked decoders of the Transformer. The decoder of the Transformer consists of self-attention layers and feed-forward neural networks with residual connection[2] and normalization[3] layers. The differences between different Transformer-based models are mainly the model architectures and the training tasks. Different architectures

[2]Residual connection is an optimization that reduces the possibility of the gradient vanishing by reserving the information from before the calculation. It is an operation that sums up the input vectors (x) and output vectors ($F(x)$) from a certain layer.

[3]Normalization is a data processing method used to scale data proportionally to fit within a specific range.

and training tasks make them have different performances in different downstream tasks.

BERT employs the bidirectional encoder architecture of the Transformer; this allows it to consider the information from both sides of a sequence of tokens (i.e., left and right). Its pre-training tasks, as mentioned in Section 2.2.1, allow it to have a better performance in understanding the context. Hence, it is widely used for encoding the inputs into vectors rich in semantic information. GPT, as illustrated above, is composed of a unidirectional decoder of the Transformer, and the pre-training tasks are all generative tasks. These two features allow GPT to excel in auto-regressive text-generating tasks.

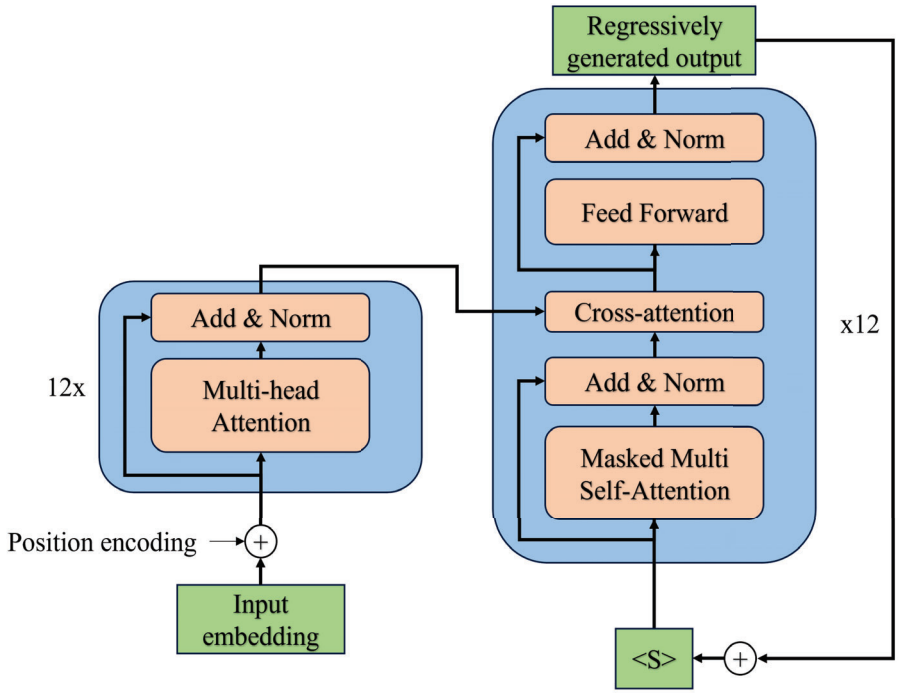
2.2.3 BART

The baseline of our thesis is BART. It is a sequence-to-sequence model whose encoder is BERT and the decoder is GPT. The modifications are made to the encoder. Specifically, there is no feed-forward neural network in the layers of the encoder of BART. However, BERT has it before word predictions. Additionally, BART employs a cross-attention layer before the decoder to distill the information relations between different sequences (the inputs and the currently generated outputs). The specific structure of BART can be seen in Figure 2.2. According to the source document of BART (Lewis et al., 2020), it underwent five pre-training tasks, including:

1. **Token Masking**: Random tokens are masked with special symbols.
2. **Token Deletion**: Random tokens are deleted.
3. **Text Infilling**: Several text spans are sampled with special symbols.
4. **Sentence Permutation**: Documents are divided into sentences and shuffled.
5. **Document Rotation**: A token is randomly chosen and set as the start of the document. Then the first half of the document is swapped with the second half of the document.

All the pre-training tasks aim at generating the original document without any noise. After the pre-training, BART has a strong ability to reduce the noise in the input sequences, thus, can generate more complete and coherent text. BART can be fine-tuned on tasks such as sequence classification tasks, token classification tasks, sequence generation tasks, and machine translation tasks. When doing text summarization tasks, we regard them as sequence generation tasks.

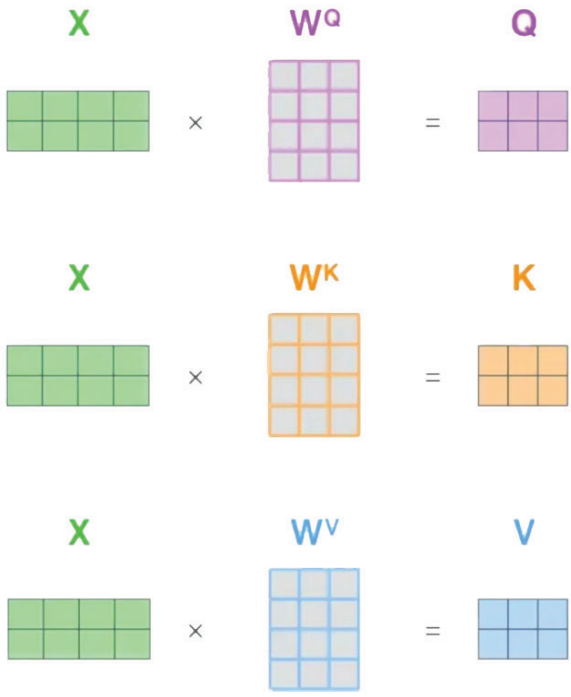
Figure 2.2: Inner structure of BART.



Attention mechanism plays a vital role in all the Transformer-based models. It can represent the relations among word vectors. In each attention layer, three weight matrices (they are parts of the parameters to be updated in the training process) indicate the weights of the query (W^Q), key (W^K) and value (W^V), separately. The matrices are then used to calculate the matrices of Query (Q), Key (K) and Value (V) by dot-producting the word vectors X and the weight matrices W^Q, W^K, W^V. The schema of getting Q, K, and V is shown in Figure 2.3.

The final attention scores are then computed with these three matrices (Q, K, V) based

Figure 2.3: Schema of calculating matrices Q, K and V (Vaswani et al., 2017).



on Equation 2.1.

$$Attention(Q,K,V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

In this way, each word will get a score indicating the weight of itself in the whole sequence. This allows the model to identify the degree of importance of the words in the sequence. Additionally, self-attention is only used to deal with relationships between elements in a single sequence without involving alignment between two different sequences; however, cross-attention can process the relationships between two different sequences. The main difference between self-attention and cross-attention is that the Q matrix of self-attention is from the same sequence but the one from cross-attention is from another sequence. In the structure of BART shown in Figure 2.2, in the cross-attention layer, the K and V matrices are both from the output of the encoder, and the Q matrix is obtained from the generated sequence.

As is shown in Figure 2.2, when doing text generation, the encoder takes the word

embeddings and the corresponding position embeddings to the multi-head attention layer, followed by the residual connection and the normalization layer. When decoding the given sequence, K and V from the encoder are passed into the decoder for calculating the cross-attention values. The cross-attention values indicate the weight of the tokens from the inputs. This helps the model identify which parts of the inputs relate the most to the next word that is going to be generated and make predictions accordingly. The predicting process usually starts from a starting symbol (e.g., $< START >$). After generating a new word, BART will take the new word, as the rear of the input, to the decoder to generate the next word regressively. This procedure will keep looping until the ending token (e.g., $< END >$) is predicted.

2.3 Neural Network Training

When a neural network is being trained, it first makes predictions based on the initial parameters. However, those firstly predicted results are not accurate enough. Therefore, the model needs to know how far these predicted results are from the reference results so that it can make adjustments. The functions used to calculate the difference between predicted results and reference results are known as loss functions. The most widely used loss function in language models is the Cross-Entropy (Zhang and Sabuncu, 2018) loss function. When it comes to two-class classification tasks, the Cross-Entropy function will be in the format of a Logit loss function (Equation 2.2).

$$l_n = -\frac{1}{N}\sum_{n=1}^{N}[y_n \log \sigma(x_n) + (1-y_n)\log(1-\sigma(x_n))] \tag{2.2}$$

In Equation 2.2, x_n indicates the nth input, and y_n indicates the referenced result of the nth input. σ stands for the *sigmoid* activation function (Equation 2.3) which is used for normalizing the input to a number between 0 and 1 for later operations. N indicates the

total number of inputs in one batch.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.3}$$

Neural networks make adjustments to the parameters by decreasing the value of the loss. One of the best algorithms for adjusting the parameters is *Adam* (Kingma and Ba, 2017). The following equations show its detailed procedure:

$$m_t = \beta_1 * m_{t-1} + (1-\beta_1) * gradient, \beta_1 = 0.9 \tag{2.4}$$

$$v_t = \beta_2 * v_{t-1} + (1-\beta_2) * gradient^2, \beta_2 = 0.999 \tag{2.5}$$

$$m_{corrected} = \frac{m_t}{1-\beta_1} \tag{2.6}$$

$$v_{corrected} = \frac{v_t}{1-\beta_2} \tag{2.7}$$

$$w_{new} = w_{old} - \frac{lr * m_{corrected}}{\sqrt{v_{corrected}} + \varepsilon}, \varepsilon = 10 - e8 \tag{2.8}$$

where m and v are initially zero vectors. They will be updated with the training process. The *gradient* is the partial derivative of the loss function with respect to a parameter w. w_{old} is the parameter before updating, and w_{new} is the parameter after updating. lr indicates the learning rate of the training process.

By adjusting all the parameters with algorithms like *Adam*, the model can re-predict the results with the updated parameters. This process keeps going until the parameters barely need to be changed.

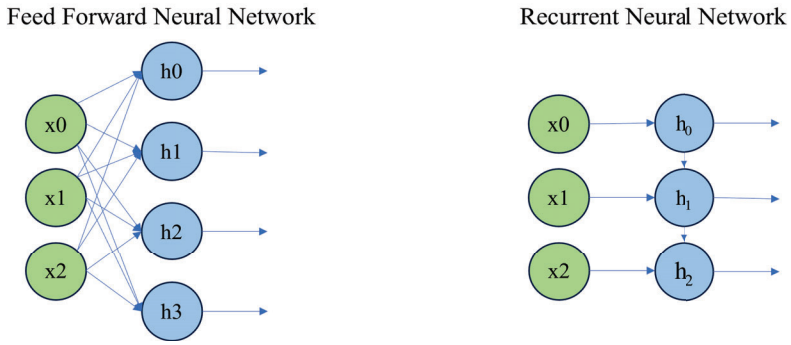
2.4 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a type of artificial neural network designed to process sequential data by maintaining an internal state or memory. Unlike feedforward neural networks, RNNs have connections that loop back on themselves, allowing them to exhibit temporal dynamics and capture dependencies in sequences.

In the context of our research, RNN is used to segment the long text into EDUs by taking the articles as inputs and outputting the labels of each word in the articles. RNNs allow inputs with different sequence lengths, so that different lengths of articles can be batched and input into the same RNN structure.

Traditional feed-forward neural networks (FFNN) have many different neurons; thus, the parameters are not shared among them. Each token of a sequence should go through all the neurons to ensure the model can fully get the information from the whole input. Differently, RNN neurons have links back to themselves, which means they can share the parameters and the information from the input among the neurons. A brief description of the differences between the FFNNs and RNNs is shown in Figure 2.4.

Figure 2.4: The differences between FFNNs and RNNs.



In Figure 2.4, the architecture of a FFNN is shown on the left and the structure of an RNN is shown on the right. *x0-x2* are the input tokens and *h0-h3* are the hidden units of the FFNN, while h_0-h_2 is the hidden unit of the RNN at different time steps. Due to their characteristics, the amount of parameters of an RNN is usually much less than the same size of FFNN. Also, RNNs are sensitive to chronological information and sequential information due to their features.

Long Short-Term Memory (LSTM) is a specialized variant of RNN units that has gained prominence in recent years due to its ability to effectively capture long-range dependencies in sequential data. Unlike traditional RNNs, LSTM networks are equipped with a memory cell and a set of gating mechanisms that allow them to selectively store,

update, or retrieve information over extended periods of time. This architecture mitigates the vanishing gradient problem, which is a common challenge in training standard RNNs. LSTMs have found wide application in various fields, including natural language processing, time series analysis, and speech recognition, due to their capacity to model complex sequential patterns and their resistance to gradient vanishing issues.

BiLSTM stands for bidirectional LSTM and represents an extension of the LSTM framework. It empowers units at time step t not only to gain information from the prior time step $t-1$ but also proficiently acquire insights from the subsequent time step $t+1$. This bidirectional characteristic enhances the predictive accuracy of RNNs, enabling them to make more refined and comprehensive predictions, ultimately contributing to the robust modeling of sequential data.

2.5 Graph Neural Networks

Graphs are common data structures that represent relationships and interactions between entities in a wide range of fields, including social networks, recommendation systems, biology, and more. Analyzing and making predictions on such structured data poses unique challenges. This is where Graph Neural Networks come into play.

Graph Neural Networks (GNNs) are neural networks that operate by propagating information across graph nodes and edges through layers. They iteratively aggregate information from the node features, neighborhood features, and edge features, followed by the update of the node representations. This allows GNNs to capture local and global patterns so that they can serve as good tools in many downstream tasks.

Different from traditional deep learning models, GNNs learn their patterns regardless of the order of the inputs because when GNNs are updating their parameters, the way they aggregate information ignores the order of the nodes in a graph. For example, the average function can always get the same result when the numbers to be summed up are shuffled. This allows GNNs to make graph-related predictions even when the graphs don't contain

the position information of their nodes.

2.5.1 Traditional GNNs

The predicting process of GNNs can be described in two steps: message computation and aggregation. In the first stage, the representations of all the nodes u ($u \in N(v)$, where $N(v)$ are the set of all the neighbors of a certain node v) will be calculated, and in the next stage, the representations of the neighbors of node v will be aggregated by a certain function (e.g., average). Then a non-linear function will be applied to update the representation of node v. All the representations of nodes are updated in the same way.

An example of how the representations of nodes in a graph are updated by Graph Convolutional Network (GCN) is shown in Equations 2.9 and 2.10:

$$Message : m_u^{(l)} = \frac{1}{|N(v)|} W^{(l)} h_u^{(l-1)} \tag{2.9}$$

$$Aggregation : h_v^{(l)} = \sigma(Sum(m_u^{(l)},\ u \in N(v))) \tag{2.10}$$

where $W^{(l)}$ indicates the parameter matrix at layer l, $|N(v)|$ indicates the degree of node v, meaning that the neural network normalizes the node representations by their degrees. σ is a non-linear function (e.g., *sigmoid*) and $Sum()$ is the summation function to aggregate the information. $h_v^{(l)}$ is the new representation of node v at layer l. According to the equations, only the representations of the neighbors of node v from the last layer ($h_u^{(l-1)}$) are used to compute the new representations.

2.5.2 Graph Attention Networks

Traditional GNNs have been widely used in graph-based machine learning tasks. But they still have certain limitations. When dealing with large-scale datasets, the way they aggregate information from all the neighbors can cause massive amounts of calculations. Additionally, traditional GNNs treat all the neighbor nodes with equal importance. When aggregating information, they don't distinguish between neighbors that carry more or less

relevant information. This can lead to suboptimal representations when important neighbors are overwhelmed by less important ones. GATs (Veličković et al., 2017), known as the Graph Attention Networks, can easily solve the problems raised above.

The attention mechanism has been proven to be effective in capturing context and dependencies in sequences. Therefore, GATs treat the nodes differently with attention scores because the weights indicate the relevance of the neighbor nodes to the target node. The core idea of GATs is that, although it is based on traditional GNNs, GATs compute node messages by weighting the nodes with attention scores instead of weighting the nodes all the same. The workflow of representation updating of GATs is illustrated as:

$$e_{vu} = a(W^{(l)}h_u^{(l-1)}, W^{(l)}h_v^{(l-1)}) \tag{2.11}$$

$$\alpha_{vu} = \frac{exp(e_{vu})}{\sum_{k\in N(v)} exp(e_{vk})} \tag{2.12}$$

$$Message : m_u^{(l)} = \alpha_{vu} W^{(l)} h_u^{(l-1)} \tag{2.13}$$

$$Aggregation : h_v^{(l)} = \sigma(Sum(m_u^{(l)},\ u \in N(v))) \tag{2.14}$$

where a is the attention mechanism process. The attention approach varies from different a. For example, $a(W^{(l)}h_u^{(l-1)}, W^{(l)}h_v^{(l-1)})$ can be the concatenation of $W^{(l)}h_u^{(l-1)}$ and $W^{(l)}h_v^{(l-1)}$ followed by a linear layer[4]. In that case, e_{vu} can be depicted as Equation 2.15. α_{vu} is the normalization function that ensures the summation of all the weights u to v is 1.

$$e_{vu} = Linear(Concat(W^{(l)}h_u^{(l-1)}, W^{(l)}h_v^{(l-1)})) \tag{2.15}$$

The information aggregated for updating the node v embedding isn't necessarily related to the previous layer of node v. Thus, GATs leverage the concatenation of the information of neighbor node u and the information of v from the last layer, as indicated in Equation 2.11.

[4]A linear layer is a neural network in which every node in two neighboring layers is connected to every node in the other layer.

2.6 Recall-Oriented Understudy for Gisting Evaluation

Recall-Oriented Understudy for Gisting Evaluation (ROUGE) (Lin, 2004) metrics are a set of measures used to assess the quality of automatic summaries. They are primarily designed to gauge the similarity between generated summaries and reference summaries. There are multiple different types of ROUGE scores, like ROUGE-1, ROUGE-2, and ROUGE-L. Every ROUGE metric has three sub-metrics: recall, precision, and F1.

ROUGE-1 measures unigram overlap between the generated summary and the reference summary; it computes the overlap of individual words (unigrams) between the two summaries. ROUGE-2 measures the bigram overlap between the generated summary and the reference summary; it computes the overlap of consecutive pairs of words (bigrams) between the two summaries; ROUGE-L measures the longest common subsequence (LCS) between the generated summary and the reference summary; it computes the overlap based on the longest sequence of words that appear in both summaries in the same order but not necessarily consecutively. The equations for calculating them are in Appendix B.

The F1 score provides a balanced measure of both precision and recall, which are crucial in tasks like text summarization. By focusing on the F1 score, researchers can present a more holistic view of their system's performance, as it accounts for both the completeness (recall) and accuracy (precision) of the generated summaries. That is also why all the abstractive text summarization methodologies only publish their F1 scores.

2.7 Summary

In this chapter, we introduced the key concepts about the dataset, RST, and the models we used in our work, including the large language model, BART, and the graph neural network, GAT. Bart serves as the fusing model in our work, and GAT plays the role of the EDU selecting model. Additionally, RNNs, as a part of our EDU segmentation model, were explained in general. In the following chapter, we are going to separately make a detailed description of the three stages of our method.

Chapter 3

Related Work

In recent years, the field of text summarization has witnessed significant growth and transformation. The advent of large language models has opened up new possibilities and challenges, making it a dynamic and evolving domain. Despite these advancements, there is still a large gap between summaries generated by the models and those annotated by humans.

To address this gap, it is essential to review and synthesize the relevant literature in text summarization. In the following section, we will explore the key studies, theories, and methodologies that form the foundation of our research. By contextualizing our work within this landscape, we aim to underscore its significance and contributions.

3.1 Basic Text Units

Abstractive text summarization represents a highly precise approach for generating sequences that conclude the essential content of the source text, with reduced reliance on copied material compared to extractive text summarization. One of the centers of the effectiveness of abstractive text summarization is the basic unit of summaries, which plays a pivotal role in determining the summarization accuracy. In contrast to full articles, key sentences or paragraphs offer denser information, yet they still have potential redundancy that can be judiciously discarded. For instance, some prior methods (Lebanoff et al., 2019; See et al., 2017; Chen and Bansal, 2018; Brook Weiss et al., 2022; Thadani and McKeown, 2013) employed sentences as the fundamental building blocks for text summarization. Liu et al. (2018) utilized paragraphs as their fundamental units to distill information. The

methodology proposed by Li et al. (2017) combined sentence-level information distilling and word-level distilling. Similarly, Cohan et al. (2018) employed a hierarchical model from section level to word level to distill information. Although their fusing units are more fine-grained than the whole article, they can still cause a great amount of unexpected noise.

The field of multi-document text summarization (Li et al., 2017; Liu et al., 2018; Bing et al., 2015; Brook Weiss et al., 2022) and long document text summarization (Cohan et al., 2018) are not the center of our research, but we can survey them as references to design our method. For the scope of our study, we will be focusing on single document summarization.

3.2 Discourse Parsing Techniques

RST-based (Mann and Thompson, 1988) datasets are used in many of the works. Recently, Wang et al. (2018) employed a neural network-based methodology in conjunction with the RST Discourse Treebank (RST-DT) (Carlson et al., 2001) dataset to segment sentences into EDUs. These EDUs are then utilized as inputs for downstream tasks. This method was used in the research of Li et al. (2020), as well as ours. Moreover, diverse techniques exist for sentence segmentation, including the methods rooted in Text Cohesion Theory (Lebanoff et al., 2020) and phrase division (Li et al., 2017; Bing et al., 2015).

3.3 Encoding Techniques

The subsequent stage of segmentation in text summarization involves the selection of important units. Investigations, as demonstrated in the works of researchers such as Li et al. (2020), Lebanoff et al. (2019), and Chen and Bansal (2018) all employed the selection of the key informational components before they fuse them. To this end, the infusion of semantic information into these units has great significance. More precisely, the key lies in the word representations, wherein Embeddings from Language Models (ELMo) (Peters et al., 2018) is used as a strategic choice, as exemplified by Wang et al. (2018). ELMo was pre-trained on large-size datasets, serving the purpose of avoiding the size constraints of segmentation

corpora. Additionally, alternative approaches include static word representations such as Convolutional Neural Networks (CNNs) as investigated by Chen and Bansal (2018), and the utilization of Bag of Words (BoW) vectors as demonstrated by Li et al. (2017). Nonetheless, attention-based embeddings, with the dynamic representations[5] of words, were leveraged by Rush et al. (2015) as a prevailing choice due to their extraordinary ability to distill contextually embedded information.

3.4 EDU Selection Techniques

DiscoBERT proposed by Xu et al. (2020a) stands out as a prominent EDU selecting model with BERT (Devlin et al., 2019) as its encoder. It used Discourse Parsing from Linear Projection (DPLP) (Ji and Eisenstein, 2014) as the method to build RST trees for each dataset it was using. Also, this method adeptly processed converted RST graphs and coreference graphs derived from articles, identifying the most coherent concatenation of EDUs that integrate into summaries. These concatenated sequences then directly assume the role of summaries. Inspired by this method, we propose to use RST graphs as inputs for our Graph Attention Network (Veličković et al., 2017) which serves as the selector in our framework. Also, we employ BERT as the encoder to enhance the extraction of dependencies among EDUs. For the convenience of deploying the model, we leverage the repository proposed by Fey and Lenssen (2019) to build the architecture of our model.

3.5 Dataset Optimizations

Some of the prior approaches have centered their focus on dataset optimization as an effective avenue for enhancing accuracy. Typically, the extensively employed CNN/Daily Mail (Hermann et al., 2015) dataset emerges as an optimal selection of Li et al. (2020); Lebanoff et al. (2019); Xu et al. (2020b); See et al. (2017); Liu et al. (2018); Lebanoff et al. (2020), offering a comprehensive corpus that spans diverse domains. Additionally,

[5]The way data or features are represented can evolve, adapt, or change based on context, input, or learning.

New York Times (NYT) (Consortium and Company, 2008), Gigaword (Graff et al., 2003), and Extreme Summary (XSum) (Narayan et al., 2018) are also datasets with good quality. The manual curation of datasets (e.g., PYRFUS (Thadani and McKeown, 2013) based on Pyramid Method (Nenkova and Passonneau, 2004)) proves to be an endeavor of substantial temporal and financial investment. Therefore, it is considered only under feasible circumstances. However, dataset optimization built upon pre-existing manually curated collections (Brook Weiss et al., 2022) can be a better option.

3.6 Model Architecture Optimizations

An array of diverse architectures characterizes the state-of-the-art approaches. Typically, the traditional RNN-based neural network, Pointer Generator Network (PGNet) (See et al., 2017) emerges as a favorable choice for sentence fusion, as evidenced by Li et al. (2020), Lebanoff et al. (2019) and Xu et al. (2020b). The integration of Pointer Network (Ptr-Net) (Vinyals et al., 2015) augments the efficacy of the selection phase by facilitating the generation of output completely from input, a strategy notably adopted by Chen and Bansal (2018). Furthermore, Transformer-based (Vaswani et al., 2017) models (Lewis et al., 2020; Radford et al., 2018; Devlin et al., 2019), constitute valuable alternatives in instances where neural networks are not employed (Rush et al., 2015). Other architectural enhancements are often realized through structural modifications[6] (Li et al., 2017) or the incorporation of supplementary sub-structures[7] (Xu et al., 2020b).

3.7 Optimization Mechanisms

Various mechanisms are harnessed to enhance the model's precision. Notably, Xu et al. (2020b), See et al. (2017) and Chen and Bansal (2018) employed the copy mechanism, facilitating not only the generation of novel terms but also the direct replication of words from the source text, effectively addressing the out-of-vocabulary (OOV) challenge. The

[6]Modifying certain components of some previously proposed models.
[7]Adding extra components based on previouly proposed models.

efficacy of the coverage mechanism (Tu et al., 2016), widely employed in Neural Machine Translation (NMT) (Luong et al., 2015; Cohan et al., 2018), extends to the domain of text summarization. Moreover, optimization strategies, like reinforcement learning leveraged by Li et al. (2020) and Chen and Bansal (2018), window size used by Luong et al. (2015), Wang et al. (2018), and (Song et al., 2019), and minimum-error rate translation (Rush et al., 2015), are prevalent. Post-fusion refinement is achievable through the imposition of linguistically-motivated constraints (Li et al., 2017; Bing et al., 2015; Thadani and McKeown, 2013) or the application of Conditional Random Field (CRF)[8] (Wang et al., 2018).

3.8 Summary

Based on the review of the existing body of research, as outlined above, we then meticulously determined the direction for our study. It incorporates various neural network-based models with some optimization on the datasets. This comprehensive survey not only serves to acquaint us with the landscape of prior investigations but also facilitates the specification of the precise research direction we are poised to pursue. In the upcoming chapter, we will delve deeper into our research, where we shall introduce the relevant conceptual frameworks and methodologies that underpin our work.

[8] A probabilistic model that is usually used to filter the unreasonable output sequences.

Chapter 4

Approach

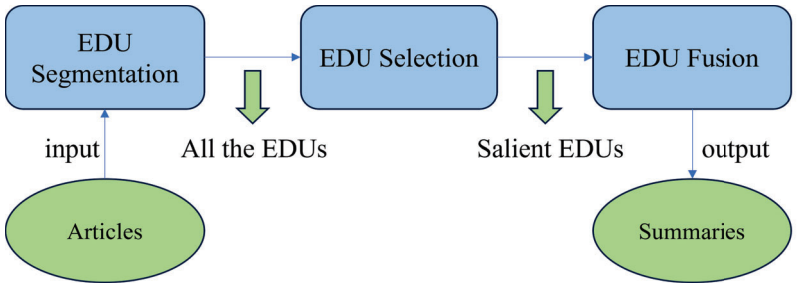
Our proposed approach can be generally summarized in the following three stages (Figure 4.1):

1. Segmenting the original long text sequences into fine-grained units in comparison with sentences and preserving the dependencies among them in order to be selected.

2. Selecting the informative units as the candidate units to be fused according to the semantic relations among all the discourse units.

3. Fusing the selected units by taking the concatenations of them as the inputs of the fusing model to ensure the coherence of the final outputs in an abstractive way.

Figure 4.1: The three stages of our work.



We propose to use this procedure to maximize the semantic coherence of the summaries while ensuring a high density of information on the summarized content. Specifically, our objective is to enhance the summarization accuracy of our model by identifying an appropriate method for EDU selection and subsequent EDU fusion. We intend to exceed the BART-large baseline by using the method above.

4.1 EDU Segmentation

EDU segmentation is the process of segmenting the source articles into EDU lists based on the discourse rules. Being fine-grained, EDUs have less redundancy than sentences. Thus, are more appropriate to be treated as the basic units to be fused. We treat the human-annotated dataset, RST Discourse Treebank (RST-DT) (Carlson et al., 2001), as the ground-truth segmentation dataset for training. RST-DT dataset is one of the most used references for training EDU segmentation models and RST tree building models. It is a collection of text documents annotated according to the principles of Rhetorical Structure Theory (Mann and Thompson, 1988).

In this section, we introduce the source corpus used to generate the EDU-based corpus and the target corpus with their structures. Moreover, we introduce the model for EDU segmentation, followed by the methodology that illustrates how RST trees can be built theoretically.

4.1.1 Dataset

In the EDU segmentation stage, the RST Discourse Treebank dataset was used to serve as the corpus to train, test, and validate the neural segmentation model proposed by Wang et al. (2018). The dataset was originally provided by the Linguistic Data Consortium Catalog (LDC)[9] and was preprocessed by the code of Wang et al. (2018).

RST-DT dataset typically includes text documents, which can be articles, essays, news, or any form of written discourse. And it also has the segmented units of the text. For our task specifically, these are the EDUs. Additionally, the labels for the relations between every two EDUs are annotated. Common relations include “elaboration”, “list” and “span”. The labels of whether an EDU is “satellite” (non-core) or “nucleus” (core) are included as well. We leveraged the model trained based on the preprocessed RST-DT dataset to directly segment the CNN/Daily Mail dataset.

[9] https://catalog.ldc.upenn.edu/products/LDC2002T07.

The raw CNN/Daily Mail stories can be obtained on the DeepMind[10] website. We downloaded approximately 300,000 articles from it and distilled the articles and highlights separately. The highlights are then treated as the referenced summaries. Each article-summary pair is stored in a JSON file with the format of a Python dictionary. An example of the format is shown in Appendix A.

4.1.2 Segmentation Model

The neural model provided by Wang et al. (2018) can segment sentences into EDUs with an accuracy of around 94.5% on the RST-DT dataset. Therefore, we consider the method to be relatively highly accurate. This tool leveraged a pre-trained word encoder model ELMo (Peters et al., 2018) to counter the limitation of the dataset size. Since ELMo was pre-trained on a billion-word benchmark corpus, it has a great ability to capture word features even though the dataset has a limited size.

Figure 4.2: The structure of EDU segmentation model (Wang et al., 2018).

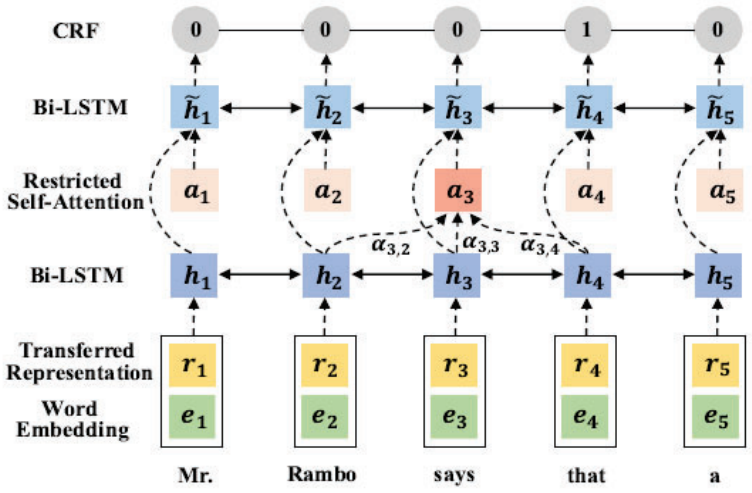


Figure 4.2 depicts the architecture of the model for EDU segmentation. The overall structure of the neural model consists of two layers of bi-LSTM with one layer of CRF, and the self-attention mechanism was employed in between the two layers of bi-LSTM to capture the long-distance dependencies among words. However, the long-distance information

[10]https://cs.nyu.edu/ kcho/DMQA/.

could sometimes bring more noise to the calculation of attention scores. Therefore, a window size is applied to restrict the scope from which the self-attention mechanism extracts previous information.

The original attention score can be acquired from the following Equations:

$$\alpha_{i,j} = \frac{e^{s_{i,j}}}{\sum_{k=0}^{N} e^{s_{i,k}}} \tag{4.1}$$

$$a_i = \sum_{j=0}^{N} \alpha_{i,j} * h_i \tag{4.2}$$

The attention score with the window size K is computed from:

$$\alpha_{i,j} = \frac{e^{s_{i,j}}}{\sum_{k=-K}^{K} e^{s_{i,i+k}}} \tag{4.3}$$

$$a_i = \sum_{k=-K}^{K} \alpha_{i,i+k} h_{i+k} \tag{4.4}$$

where $s_{i,j}$ indicates the similarity between word x_i and each nearby word x_j, and K is the hyper-parameter that represents the window size. h_i indicates the hidden state of the first bi-LSTM layer of word x_i. The outputs are the sequences composed of 1 and 0, where 1 represents the last word of an EDU and 0 indicates other words in an EDU.

According to the predicted sequences (composed with '1' and '0'), sentences are segmented into Elementary Discourse Units and persisted as arrays of strings. These arrays are assigned as values for the 'EDUs' keys and then saved into JSON files alongside the original list of sentences.

4.1.3 RST Tree Building

Discourse Parsing from Linear Projection (DPLP) was proposed by Ji and Eisenstein (2014) to conduct RST parsing and make predictions for relations between each of the two EDUs and nuclearities of EDUs. Essentially, DPLP is a shift-reduce[11] method and

[11] Shift-reduce is a technique used for syntax analysis. It's based on syntax rules. There is a stack and a queue in the method to help with analyzing the syntax.

the operations (*shift* or *reduce*) are determined by $\hat{m}$ which is obtained by maximizing the objective function. Equation 4.5 is the objective function of the model. The overall task can be seen as a multi-class classification task.

$$\hat{m} = \arg\max w_m^T f(v;A),\ m \in \{1,..,C\} \tag{4.5}$$

where w_m refers to the weight for the mth class and f signifies the function that transforms the lexical features into a lower-dimensional latent representation and is parameterized by the projection matrix A (e.g., $f(v,A) = vA$). v is the concatenation of the features of the first two EDUs in the stack and the features of the head EDU in the queue[12] (Ji and Eisenstein, 2014).

If the determination of *reduce* is made, the type of relation and the nuclearity of the EDUs should also be determined. Consequently, the built RST trees have both the relations between each of the two EDUs and the nuclearities of EDUs.

4.2 EDU Selection

The consideration of nuclearity in an RST tree provides a general indication of the hierarchical importance of different units, but it might not capture the full spectrum of significance or relevance in all contexts. Moreover, nuclearity is a valuable concept in RST for understanding the general structure and hierarchy of discourse units. Selecting important EDUs provides a more refined and task-specific perspective on the significance of individual elements within the discourse. Therefore, selecting EDUs based on their nuclearities and making determinations under global conditions is necessary.

Due to the heterogeneity of RST trees, the predictions of the nodes can be difficult since both the classifications of relations and EDUs should be made. To tackle this problem, we consider employing the converted RST tree graph, which is essentially a type of directed

[12]The stack stores the *shifted* EDUs from the queue in a *shift-reduce* process, and the queue represents the inputs which in this case are all the EDUs in an article.

acyclic homogenous graph. The converted RST graph is introduced in Section 4.2.1.

The converted RST graph can be regarded as the graph that demonstrates the dependencies among the EDUs in an article. The prediction of the importance of EDUs is essentially a node-level classification task. We label an EDU as '1' if it's important and as '0' if it's not. Accordingly, we decided to use Graph Neural Networks that can fit the demand of taking graphs as the inputs and making node-level predictions.

GAT is utilized as the model for EDU selection due to its use of the mechanism of weighted nodes (Section 2.5.2), which meets the requirement of assigning different importance to different nodes in our task. Alternatively, the greedy method was originally employed to test if the dataset has at least a regular performance.

4.2.1 Dataset

Considering that predicting the importance of EDUs doesn't require us to know the exact types between each of the two EDUs, we decided to use the converted RST graphs as our dataset. The converted RST graphs of CNN/Daily Mail were uploaded to the repositry[13] by Xu et al. (2020a). The conversion from the original RST trees to converted homogeneous graphs is shown in Figure 4.3:

Figure 4.3: The converting process of an RST tree.

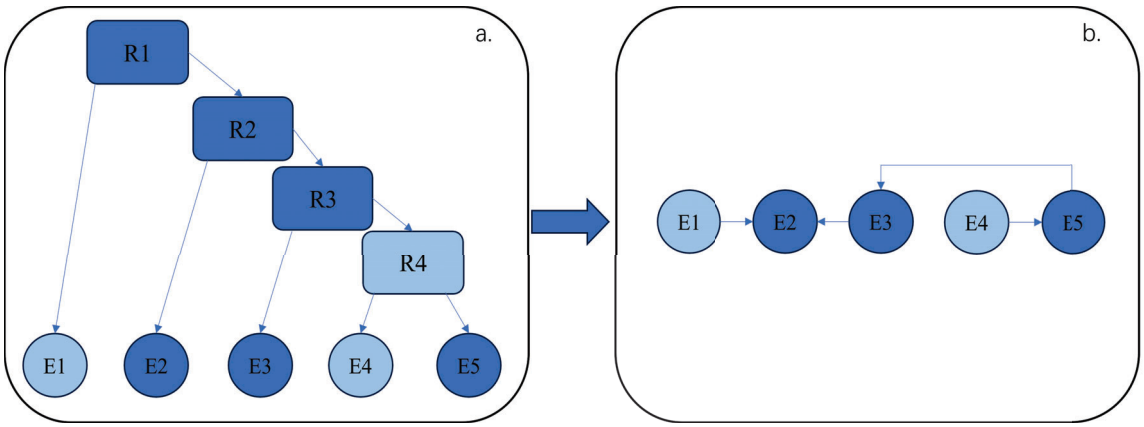


Figure 4.3.a shows the original RST tree and Figure 4.3.b shows the converted RST

[13]https://utexas.app.box.com/v/DiscoBERT-ACL2020.

discourse tree. The dark blue nodes are nucleus nodes and the light blue nodes are satellite ones. **R1-R4** are relations between each of the two nodes and **E1-E5** indicate the EDUs. The converting principles can be summarized as two rules: *In terms of the intermediate nodes, (a) if the two children are both nucleus nodes, then the head of the left node is the right node. (b) if there is one satellite node and one nucleus node, the head of the nucleus node is the satellite node* (Xu et al., 2020a). For example, in Figure 4.3.a, children of $R1$ are $E1$ and $R2$, where $E1$ is a satellite node and $R2$ is a nucleus node. After applying rule (b), $E1$ points at the entire span of $R2$, which is the origin of node $E1$ and its arrow in Figure 4.3.b. In this way, the converted RST graphs always have *N* nodes and *N-1* edges.

The EDUs in each article will be sequentially numbered from *0* to *N-1* (N is the number of EDUs in an article), and the RST graphs are stored in a list in the form of tuples. Each tuple indicates a binary relation. In these tuples, the first column represents the indices of the head nodes, the second column indicates the indices of the tail nodes and the third column shows the classes of the relations. For example, tuple $(1, 2, `elaboration')$ means there is a *elaboration* relation between node 1 and node 2 where node 2 is an elaboration of node 1.

4.2.2 Data Loading

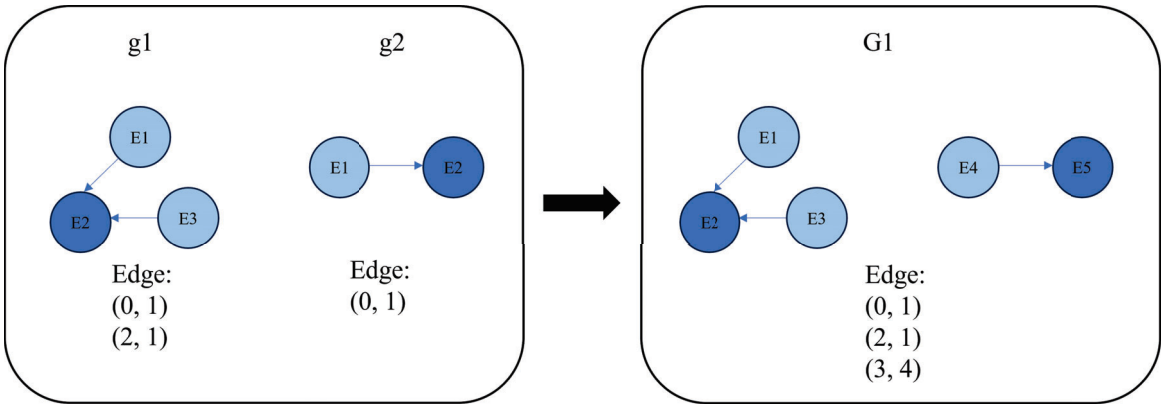
Since each article has a different number of EDUs, the articles have different shapes after being encoded into vectors, which makes it hard to employ mini-batches to train the model based on the dataset. Also, a batch size that is too small will lead to the non-convergence of training results. To tackle these problems, we used the external package *PyG*[14] from Fey and Lenssen (2019) to store our graphs and load our datasets by aligning the dimensions of the inputs before batching them.

Under this circumstance, each article is stored in a special data structure *Data*. The data loader allows us to fuse different graphs to build a large graph[15]. The number of sub-graphs

[14] https://github.com/pyg-team/pytorch_geometric.

[15] The specific way is reordering the indices of all the EDUs in a large graph, from 0 to $N_{large} - 1$, where

Figure 4.4: An example of fusing graphs.



in each large graph is fixed. When training, each batch will only contain one large graph that consists of a certain number of sub-graphs.

4.2.3 BERT-GAT Model

BERT has been proven to have a good performance in text understanding (See et al., 2017; Xu et al., 2020a) tasks. In this section, we will introduce the architecture of our selecting model and how it is trained on the graph dataset.

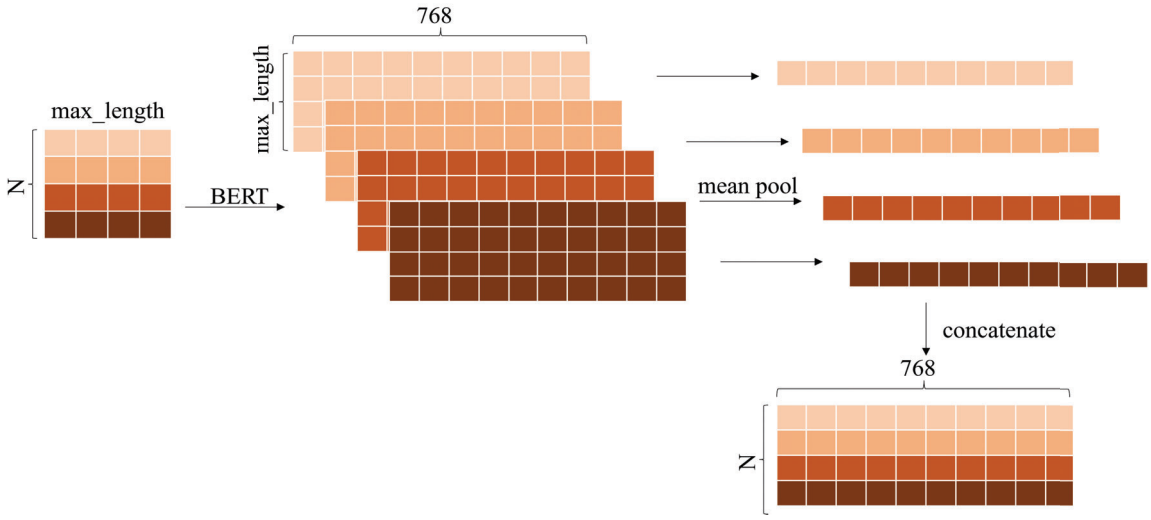
EDU encoding

Articles are segmented in the segmentation stage. We then initialize the EDUs as sequences of word indices from the BERT vocabulary. The pre-trained BERT model is leveraged to encode each article into a 3-dimensional vector $(N, max\_length, 768)$[16], where N is the number of EDUs in an article and max_length is the maximum length of an EDU. Focusing on fitting the input of GAT, we convert the 3-dimensional vectors into 2-dimensional ones by using the mean pooling strategy. Finally, we get the articles in the form of 2-dimensional vectors $(N, 768)$, where N is the number of EDUs in an article. The process is depicted in Figure 4.5. For BERT specifically, the number of features is fixed to be 768. Hence, each encoded EDU can be represented as a vector with 768 features.

N_{large} is the number of EDUs in a large graph; the indices of edges will also be reordered accordingly. Figure 4.4 shows an example of fusing the graphs.

[16]BERT encodes all inputs into vectors with a third dimension of 768

Figure 4.5: The overall process of encoding articles.



EDU classification

We designed a simple 3-layer GAT model for predicting the labels of EDUs. The task is therefore a node-level classification task. The first two layers of GAT are for information aggregation and the last GAT layer is designed for normalizing the features into the dimension of the number of classes. To solve the possible vanishing gradient problem, we used Exponential Linear Unit (ELU) Clevert et al. (2016) as the activation function. ELU can be shown as the following equations:

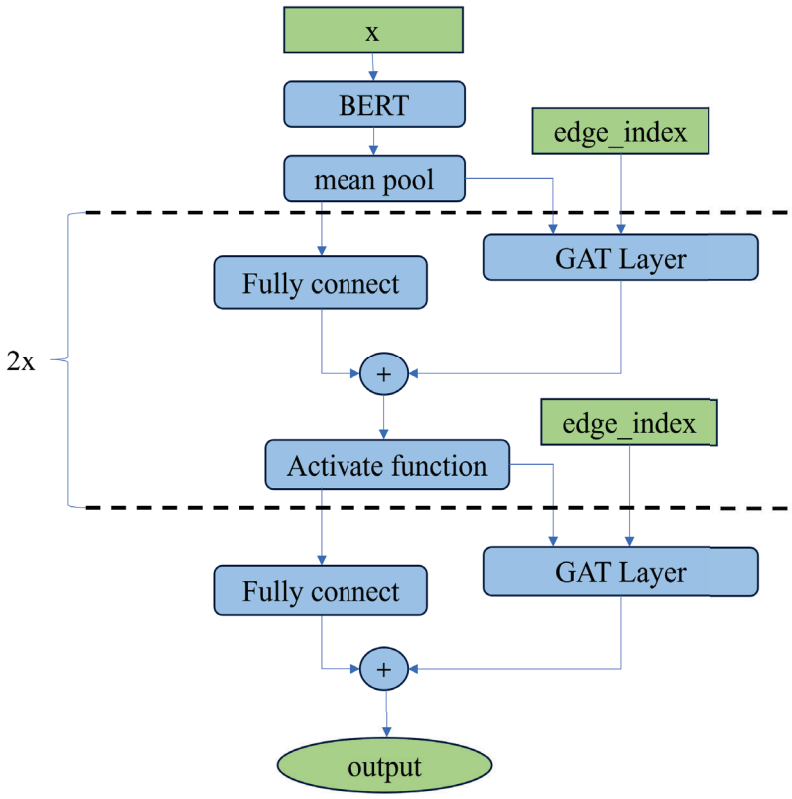
$$f(x) = x, \; if \; x > 0 \tag{4.6}$$

$$\alpha(exp(x) - 1), \; if \; x \leq 0 \tag{4.7}$$

Additionally, we leveraged element-wise summation to combine the outputs of different GAT layers with the original input, allowing the model to capture various levels of abstraction and information from different sources. Figure 4.6 shows the overall architecture of our model for EDU selection.

In all three layers of GAT, we used the multi-head mechanism to reduce the randomness of the predicted results. The first two layers of GAT concatenate the outputs of the 4 heads.

Figure 4.6: The architecture of EDU selecting model.



The last layer averages these heads and, along with the final fully connected layer, aggregates the information to predict the classes of EDUs. In Figure 4.6, x indicates the index sequences of EDUs and *edge_index* represents the edges that connect EDUs. For example, $(3,5)$ shows the edge from EDU_3 to EDU_5.

In the training stage, the logit loss function is utilized because our task is essentially a classification task. Also, for the parameter updating algorithm, *Adam* (Kingma and Ba, 2017) is employed in the model.

The trained model is then used for generating the sequences of labels of the EDUs from all the articles, where only those EDUs that are labeled with '1' will be chosen and concatenated article by article.

4.2.4 Greedy Method

An alternative approach we utilized in our study is the application of a greedy method. The article-summary pairs are first obtained and segmented as previously shown. We then combine 1, 2 and 3 EDUs from an article respectively. For all the combinations, we compute their $ROUGE-L_{recall}$ scores with respect to each sentence in the corresponding summary. Finally, we treat those EDUs that combine the combinations that maximize the $ROUGE-L_{recall}$ scores as the important EDUs.

Figure 4.7: Greedy Method.

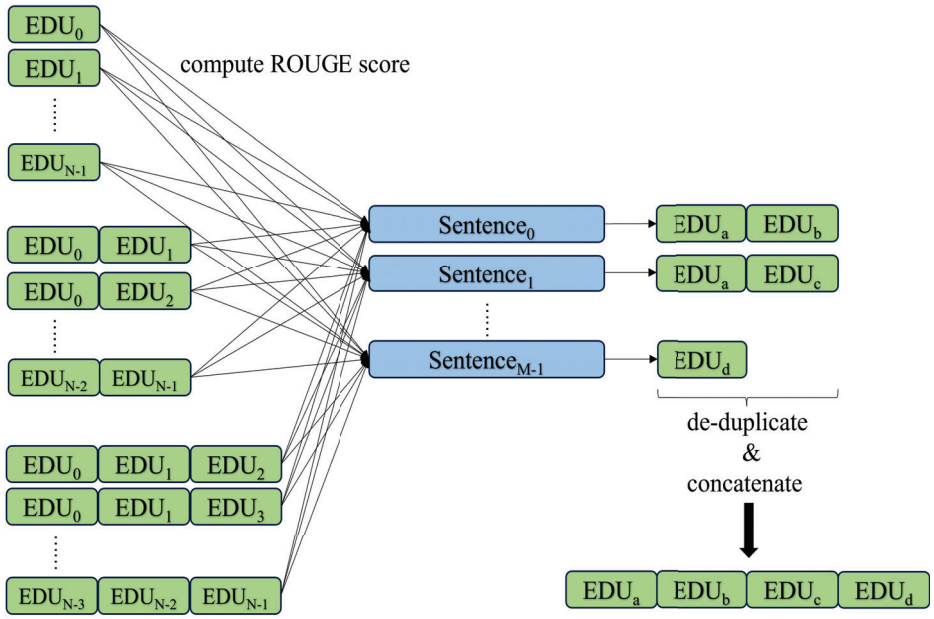


Figure 4.7 illustrates the process of selecting important EDUs with the greedy method. Within the diagram, N represents the count of EDUs within an article, while M signifies the number of sentences in a referenced summary. The indices a, b, c, and d denote potential candidate EDU positions. The procedure can also be represented using the following equation:

$$\widehat{EDU_{comb}} = \sum_{k=0}^{M-1} \arg\max(ROUGE-L_{recall}(EDU_{comb}, sentence_k)) \tag{4.8}$$

where EDU_{comb} is the set encompassing all combinations of 1, 2 and 3 EDUs, and $\widehat{EDU_{comb}}$

is the outcome achieved by concatenating the selected EDUs. $sentence_k$ indicates the kth sentence in the summary. With the objective of diminishing redundancy within concatenated EDUs, in cases where duplicate EDUs are present in the candidate lists, we eliminate the subsequent occurrences of these EDUs and retain only those EDUs that make their initial appearance in the candidate list.

Also, to counter the problem of potential empty EDUs, we set up all the ROUGE scores of an empty EDU to zero so those empty EDUs won't be selected in this method.

4.3 EDU Fusion

Abstractive summarization requires more than copying contents from the source text. Following the stage of selecting the salient units and concatenating them, the generation of summaries encompassing optimal encapsulation of the semantic content within the concatenated textual sequences becomes necessary. Due to the state-of-the-art performance of BART in natural language processing, it is employed to fuse the selected EDUs in our work.

Since there is no evaluation method to test if the selected EDUs are informative enough, both the results from the GNN-based selecting method and the greedy selecting method will be input to BART to compare which selecting method has a better performance in generating summaries.

4.3.1 Dataset

In the previous stage, we capitalized on two distinct datasets, converted RST graphs for the GNN methodology and article-summary pairs for the greedy method. Therefore, for the fusing stage, we use the two datasets from the previous outcome.

For the fusion of the converted RST graphs, we gathered all the labels assigned to the EDUs within it. This leads us to focus solely on those EDUs marked with a '1' label, which are then concatenated to serve as the input for our fusing model. For the article-summary

pairs, the outputs are the indices of important EDUs. Essentially, both datasets are the combined sequences of the selected EDUs.

For the evaluation of the converted RST-DT dataset, we encounter a situation where the associated graphs lack referenced summaries. To tackle this issue, we need to match each graph with the summaries from the article-summary pair dataset by using a hash table.

Hash Algorithm To align the RST discourse tree dataset with the CNN/Daily Mail article-summary pair dataset, we established a hash table. In this table, the document IDs serve as the strings to generate the hash values, and the values in the table are the corresponding summaries from the article-summary pair dataset. The function we used for generating hash value is *SHA-256* hash function. The *SHA-256* is used for converting the same strings into the same hexadecimal numbers and converting different strings into different hexadecimal numbers. In this way, when matching a great number of strings with their corresponding values, it's far faster than matching them by traversing the array that stores the strings. As a result, the time complexity of the hash algorithm is $O(2N)$ instead of $O(N^2)$ (N is the number of article-summary pairs in a dataset).

The summaries are stored as dictionaries within JSON files, with file names corresponding to the respective RST graph files.

4.3.2 BART

The model we used for EDU fusion was the pre-trained BART-large (Lewis et al., 2020) model. As previously presented in Section 2.2.3, BART is a sequence-to-sequence model that incorporates ideas from BERT (Devlin et al., 2019) and GPT (Radford et al., 2018). The total number of parameters of BART-large is 400M, with around 55M more than BERT-large.

Owing to the pretraining tasks of BART, it is capable of capturing the rich semantic information from the given text and making predictions regressively. We need only fine-tune BART to fit it more to the given downstream task and the corresponding dataset. The

fine-tuning object for our thesis is making BART have a better performance when taking the concatenated important EDUs as inputs, especially when the whole sequence is not coherent. The model can be trained to be capable of outputting coherent sequences with correct grammar.

Table 4.1: BART fine-tuning hyper-parameters.

hyper-parameter	value
TOTAL_NUM_UPDATES	20000
WARMUP_UPDATES	500
LR	3e-05
MAX_TOKENS	1024
UPDATE_FREQ	4

Some of the hyper-parameters are shown in Table 4.1. We set those hyper-parameters based on the ones shown on the Fairseq (Ott et al., 2019) repository[17]. The exact steps of the fine-tuning process are also done as they appear on the repository. The general process is as follows: firstly, build the vocabulary, then binarize the dataset, and finally perform model fine-tuning.

With the filtering of the selecting model, the input sequences are much shorter than the original inputs, which reduces the requirements of the device. Therefore, we modified the max length of the input from 2048 to 1024, in order to boost the encoding process. Additionally, we limit the maximum size (in megabytes) of GPU memory that can be allocated for a single operation or tensor to 32 megabytes[18]. This avoids the large allocations of operations or tensors of PyTorch, which can be useful in tackling the problem of Out-Of-Memory (OOM). The fine-tuning results of BART will then be used to compare with the fine-tuning of GPT-3 on the same datasets, the dataset gained from the GNN model and the dataset obtained from the greedy method.

[17] https://github.com/facebookresearch/fairseq/blob/main/examples/bart/README.summarization.md.
[18] export PYTORCH_CUDA_ALLOC_CONF=max_split_size_mb:32.

4.4 Summary

In this chapter, we outline all three stages (EDU segmentation, EDU selection, and EDU fusion) of our model, encompassing the development of the datasets and the architectures of each model in these stages. Throughout the chapter, we also address various challenges and obstacles encountered during our experimental process. For each of these challenges, we provide a detailed account of the solutions and strategies we employed to overcome them. This not only includes technical fixes and modifications to the code but also innovative approaches to data handling and processing.

Chapter 5

Experiments

Our work leveraged many of the methods proposed by the previous researchers. But due to the different tasks we have from theirs, modifications had to be done. Also, to measure if our work reaches the expected result, we need to use some evaluation metrics to test our model.

5.1 Implementation Details

EDU segmentation: For EDU segmentation we use the pre-trained model proposed by Wang et al. (2018). The environment is set according to the repository[19] uploaded by Li et al. (2020), with the modification of the updating of the Python package *SpaCy* to solve a compatibility issue (the old versions of *SpaCy* cannot be run under an environment with a newer version of Python).

Greedy Method: We employ the code from Wang et al. (2018) and make modifications to the calculation of ROUGE score. *pyrouge* is installed for calculating $ROUGE-L_{recall}$ score. The datasets may contain some empty EDUs or EDUs which only contain invalid characters. When meeting those EDUs, we set the ROUGE scores for all sentences to 0.

EDU Selection: GNN structures are utilized from *PyG*. Specifically, we use the environment configured as in Table 5.1.

In the data processing stage, as previously introduced, the data structure *Data* is employed for storing the graphs. *DataLoader* sews up a certain number of graphs in a mini-

[19] https://github.com/PKU-TANGENT/EDUSum.

Table 5.1: EDU selection environment.

package	version
pytorch	1.12.0-gpu
cuda	11.7
torch_geometric	2.3.1
transformers	4.30.2

batch into a large graph. The batch size of the training stage is 24 and the max length of each EDU is 20. We leverage the multi-head mechanism in each GAT layer. The number of heads in each layer is set to 4.

The training process is done on *Compute Canda - Cedar*[20] server and we use the multi-GPU mechanism to boost the training speed. Specifically, we use 4 GPUs on the server. Additionally, we utilize the *DataParallel* function in PyTorch to split the dataset into multiple ones and assign them to different GPUs.

EDU Fusion: We use the pre-trained BART-large model as our baseline, and the fine-tuning is also done based on the model. The pre-trained model can be accessed from a GitHub repository[21]. With two different datasets preprocessed by GNN and Greedy Method respectively, the max length of the input sequence is modified from 2048 to 1024[22] Other hyper-parameters are set the same as the BART model fine-tuning on the original CNN/-Daily Mail dataset.

The specific Python packages and versions are shown in Table 5.2.

Other than using BART, we also use GPT-3 as one of the testing models. GPT-3 is one of the most powerful language models proposed by OpenAI. We use the fastest model, *Ada*, in GPT-3. OpenAI has a series of well-defined APIs for users to use their models quickly and easily. Therefore, we don't need to do further configuration to set up the hyper-parameters, meaning that all the hyper-parameters were preconfigured.

[20] Official website: https://www.alliancecan.ca/en.
[21] https://github.com/facebookresearch/fairseq/tree/main/examples/bart.
[22] Due to the redundancy reduction of the original articles, the inputs are far shorter than before.

Table 5.2: EDU fusion environment.

package	version
pytorch	1.10.0-gpu
cuda	10.2
hydra-core	1.0.7
omegaconf	2.0.6
bitarray	2.6.2
sacrebleu	2.3.1
regex	2022.10.31

5.2 Evaluation

The evaluation metrics for all the experiments are the ROUGE scores of the precision, recall and F1 scores. We first leverage the Penn Treebank Tokenize from StanfordNLP[23] to tokenize all the articles and referenced summaries into lists of tokens, subsequently followed by the utilization of Python package *files2rouge* to automatically calculate the metrics between two files (hypotheses and references). The command *files2rouge* requires the two files to have the same number of lines.

The evaluation has 9 metrics in total, as declared in Appendix B. We only take the F1 score of each ROUGE metric in our results since the F1 score combines the recall and precision score, as is shown in Equation B.3, B.6 and B.9.

5.3 Results

Our work focuses on the comparison between the performance of BART fine-tuning on the original CNN/Daily Mail dataset and the performance of BART fine-tuning on the concatenated sequences of selected EDUs from the CNN/Daily Mail dataset.

With different pre-processing methods, we have two different datasets processed based on the CNN/Daily Mail dataset, the article-summary pair dataset (Table 5.3) and the converted RST tree dataset (Table 5.4).

[23] Official website: https://stanfordnlp.github.io/CoreNLP/.

Table 5.3: Article-summary pair dataset size.

	Number of Graphs
Train	287,227
Test	11,490
Valid	13,368

Table 5.4: Converted RST discourse tree dataset size.

	Number of Graphs
Train	287,105
Test	11,490
Valid	13,368

Our results are presented in Table 5.5. The validation metrics are the F1 scores of all the ROUGE metrics. The first line introduces the baseline of BART-large, and it's directly referenced from the paper where BART was proposed. The second line shows an alternative method that combines the greedy method and *Ada* in GPT-3. The results from the third line are obtained by replacing the GPT-3 model with BART based on the method of the second line. The last line is the result of the combination of GNN and BART, which shows the best ROUGE-1 and ROUGE-2 scores among all the methods.

Table 5.5: Experimental results on CNN/Daily Mail dataset.

	ROUGE-1	ROUGE-2	ROUGE-L
BART-Large	44.16	21.28	40.90
Greedy Method + GPT-3	48.74	23.15	45.35
Greedy Method + BART	58.12	34.68	**55.37**
GNN + BART	**58.80**	**35.70**	55.30

All the validations are based on the source files that contain the sequential concatenations of selected EDUs and the target files that contain the highlights of articles. Compared with BART-Large, all the experiments with EDU selection outperform it, among which the GNN method has the best performance in terms of the ROUGE-1 (58.8%) and ROUGE-2 (35.7%), and the greedy method has the best ROUGE-L score with about 0.07% higher

than the GNN method, reaching a percentage of 55.37. Additionally, an example of the predicted summary from different method combinations is shown in Appendix C.

The results show that the summarization performance is better after changing the inputs from the whole article to the combinations of EDUs. With the same constituent of the dataset, BART outperforms *Ada* from GPT-3. Additionally, with more parameters, GNN has a better performance in terms of selecting salient EDUs than the greedy method.

5.4 Summary

The implementation details of the three phases in our work were introduced in this chapter, including the configurations of the environments, the significant data structures and some hyper-parameters. Additionally, the evaluation metrics, ROUGE scores, were further explained in this chapter. The results of all the outperforming experiments were shown in the end.

Chapter 6

Conclusion

In our work, we chose to optimize the datasets by truncating the sentences in articles into fine-grained units, EDUs (Elementary Discourse Units), and selecting the most informative ones as the candidates to be fused. We use both GNN and a greedy method to separately select those salient EDUs, and then apply BART to fuse them. In our approach, the segmentation and the selection stages serve as the encoder to label the important EDUs and concatenate the EDUs that are labeled with '1'. The decoder consists of the BART-large pre-trained model. When using GNN as the selecting method, the encoder and the decoder can be trained separately, but the inputs of the decoder have to be restricted to the outputs of the encoder. Our stage-wise method outperforms the BART-large baseline with a 14.64% higher ROUGE-1 score, 14.42% higher ROUGE-2 score and 14.4% higher ROUGE-L score in text summarization.

Transformer-based models like BART outperform most of the other neural models in terms of natural language generation. At the same time, due to the great amount of parameters of these models, they require great computational resources which means they take more time to be trained and fine-tuned. Complete articles naturally have rhetorical structures that can assist in comprehending the semantic relations, and this type of structure can always be represented as graphs, where Graph Neural Networks can come into play. Graph neural networks perform well in handling graph-based data, although they require strict preprocessing of datasets. This preprocessing can enhance performance in graph-based tasks.

Our model is only proven to work better on the CNN/Daily Mail dataset whose reference summaries are usually more than one sentence. This reduces the difficulty of selecting the salient EDUs. Furthermore, there is a large number of parameters in our model which requires a long time to fine-tune it. In our future work, we can try to use a better selection method that has a smaller volume (e.g., fewer neural network layers) and has a better performance in choosing salient units.

Bibliography

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca Passonneau. Abstractive multi-document summarization via phrase selection and merging. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1587–1597, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1153. URL https://aclanthology.org/P15-1153.

Daniela Brook Weiss, Paul Roit, Ori Ernst, and Ido Dagan. Extending multi-text sentence fusion resources via pyramid annotations. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1854–1860, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.135. URL https://aclanthology.org/2022.naacl-main.135.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurovsky. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*, 2001. URL https://aclanthology.org/W01-1605.

Yen-Chun Chen and Mohit Bansal. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1063. URL https://aclanthology.org/P18-1063.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016.

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 the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2097. URL https://aclanthology.org/N18-2097.

Linguistic Data Consortium and New York Times Company. *The New York Times Annotated Corpus*. LDC corpora. Linguistic Data Consortium, 2008. URL https://books.google.ca/books?id=D4F2AQAACAAJ.

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, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34, 2003.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Ad-*

vances in Neural Information Processing Systems, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/afdec7005cc9f14302cd0474fd0f3c96-Paper.pdf.

Yangfeng Ji and Jacob Eisenstein. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 13–24, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Logan Lebanoff, Kaiqiang Song, Franck Dernoncourt, Doo Soon Kim, Seokhwan Kim, Walter Chang, and Fei Liu. Scoring sentence singletons and pairs for abstractive summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2175–2189, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1209. URL https://aclanthology.org/P19-1209.

Logan Lebanoff, John Muchovej, Franck Dernoncourt, Doo Soon Kim, Lidan Wang, Walter Chang, and Fei Liu. Understanding points of correspondence between sentences for abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 191–198, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-srw.26. URL https://aclanthology.org/2020.acl-srw.26.

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, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

Piji Li, Wai Lam, Lidong Bing, Weiwei Guo, and Hang Li. Cascaded attention based unsupervised information distillation for compressive summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2081–2090, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1221. URL https://aclanthology.org/D17-1221.

Zhenwen Li, Wenhao Wu, and Sujian Li. Composing elementary discourse units in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6191–6196, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.551. URL https://aclanthology.org/2020.acl-main.551.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL https://aclanthology.org/D15-1166.

William C Mann and Sandra A Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text-interdisciplinary Journal for the Study of Discourse*, 8 (3):243–281, 1988.

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, 2018.

Ani Nenkova and Rebecca Passonneau. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 145–152, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics. URL https://aclanthology.org/N04-1019.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI*, 2018.

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://aclanthology.org/D15-1044.

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, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL https://aclanthology.org/P17-1099.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*, 2019.

Kapil Thadani and Kathleen McKeown. Supervised sentence fusion with single-stage inference. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1410–1418, Nagoya, Japan, October 2013. Asian Federation of Natural Language Processing. URL https://aclanthology.org/I13-1198.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811*, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

Yizhong Wang, Sujian Li, and Jingfeng Yang. Toward fast and accurate neural discourse segmentation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 962–967, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1116. URL https://aclanthology.org/D18-1116.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. Discourse-aware neural extractive text summarization. In *Proceedings of the 58th Annual Meeting of the Asso-*

ciation for Computational Linguistics, pages 5021–5031, Online, July 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.451. URL https://aclanthology.org/2020.acl-main.451.

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, Online, July 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.125. URL https://aclanthology.org/2020.acl-main.125.

Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels, 2018.

Appendix A

Article-Summary Pair

{
“id”: “6bd843fe975d42b0f2d5eedb1a02d918e4a397f5”,
“article”: [
“los angeles , california (cnn) – a 24 - year - old gang member was arrested thursday in connection with a shooting at a los angeles bus stop in which eight people were wounded , city officials said .”,
“bystanders express shock after a shooting at a bus stop in los angeles wednesday .”,
“billy ray hines is believed to have fired into a crowd of people at the bus stop . hines was apprehended thursday afternoon as he was walking down the street , about a half - mile from the scene of the shooting , police chief william bratton told reporters .”,
“hines will face 10 counts of attempted murder – one for each of the eight victims , and two more for what authorities believe to be his two intended victims , who were still being sought thursday , bratton said .”,
“authorities are also seeking the gun used in the incident , he said .”,
“the shootings took place wednesday afternoon at the intersection of central and vernon avenues , in an area where police are concerned about gang violence .”,
“five of the victims were children . an 11 - year - old girl was shot in the chest , and another girl , age 11 , was shot in the right arm . three boys were wounded – ages 10 , 12 and 14 .”,
“one was shot in the leg , one in the buttocks and the third in the ankle , police said . one man was wounded in the leg and another in the ankle , and a woman was shot in the face .”,
““while no one died yesterday , the bullets unleashed shot through the core of the entire community , ” los angeles mayor antonio villaraigosa said in announcing the arrest thursday . ” the decent people of this community responded with force . ” watch mayor , authorities discuss arrest of gunman \u00bb .”,
“witnesses came forward after the incident to identify the gunman as hines , bratton said . the shooting was believed to stem from a dispute between the gunman and the two intended victims , he said . e-mail to a friend .”
],
“abstract”: [
“billy ray hines is believed to have fired into a crowd , police say .”,
“eight people , including five children , were shot , authorities say .”,
“shooting happened wednesday near a middle school , but not on school grounds .”,

“hines will face 10 counts of attempted murder , police say .”
]
}

Appendix B

ROUGE Equations

$$ROUGE-1_{recall} = \frac{Number\ of\ matching\ unigrams}{Total\ number\ of\ unigrams\ in\ reference\ summary} \tag{B.1}$$

$$ROUGE-1_{precision} = \frac{Number\ of\ matching\ unigrams}{Total\ number\ of\ unigrams\ in\ generated\ summary} \tag{B.2}$$

$$ROUGE-1_{f1} = \frac{2 * ROUGE-1_{recall} * ROUGE-1_{precision}}{ROUGE-1_{recall} + ROUGE-1_{precision}} \tag{B.3}$$

$$ROUGE-2_{recall} = \frac{Number\ of\ matching\ bigrams}{Total\ number\ of\ bigrams\ in\ reference\ summary} \tag{B.4}$$

$$ROUGE-2_{precision} = \frac{Number\ of\ matching\ bigrams}{Total\ number\ of\ bigrams\ in\ generated\ summary} \tag{B.5}$$

$$ROUGE-2_{f1} = \frac{2 * ROUGE-2_{recall} * ROUGE-2_{precision}}{ROUGE-2_{recall} + ROUGE-2_{precision}} \tag{B.6}$$

$$ROUGE-L_{recall} = \frac{Length\ of\ LCS}{Length\ of\ reference\ summary} \tag{B.7}$$

$$ROUGE-L_{precision} = \frac{Length\ of\ LCS}{Length\ of\ generated\ summary} \tag{B.8}$$

$$ROUGE-L_{f1} = \frac{2 * ROUGE-L_{recall} * ROUGE-L_{precision}}{ROUGE-L_{recall} + ROUGE-L_{precision}} \tag{B.9}$$

Appendix C

Result Examples

Reference: classic comic book “ the dark knight returns ” is getting a second sequel . legendary comics writer frank miller is returning to the story that made him famous .

Hypothese from Greedy Method + GPT-3: frank miller will write final chapter ’s of “ the dark knight returns ” trilogy . miller is returning to “ the dark knight iii : the dark knight returns ... again ”

Hypothese from Greedy Method + BART: comic book writer frank miller is returning to his best - known story .“ the dark knight iii : the dark knight returns ” is the final chapter in his “ the dark knight returns ” trilogy . the final chapter in his “ the dark knight returns ” trilogy is “ the dark knight iii : the dark knight returns ... again ”

Hypothese from GNN + BART:

“ the dark knight returns ” comic book writer frank miller is returning to his best - known story . dc comics announced the return of “ the dark knight ” on tuesday . miller ’s “ the dark knight ” comic book series has been canceled twice in recent years .