MASTER'S THESIS

A Redundancy-Aware Length Control Framework for Extractive Summarization

LI, Shuxin

Date of Award:
2021

HONG KONG BAPTIST UNIVERSITY

Master of Philosophy

THESIS ACCEPTANCE

DATE: August 16, 2021

STUDENT'S NAME: LI Shuxin

THESIS TITLE: A Redundancy-Aware Length Control Framework for Extractive Summarization

This is to certify that the above student's thesis has been examined by the following panel members and has received full approval for acceptance in partial fulfilment of the requirements for the degree of Master of Philosophy.

Chairman:	Prof Zhang Hui Professor, Division of Science and Technology, BNU-HKBU UIC (Designated by Dean of Division of Science and Technology)
Internal Members:	Dr Lee Raymond Shu-Tak Associate Professor, Division of Science and Technology, BNU-HKBU UIC (Designated by Head of Computer Science and Technology)
	Prof Su Weifeng Professor, Division of Science and Technology, BNU-HKBU UIC
External Examiner:	Dr Wu Hejun Associate Professor Department of Computer Science Sun Yat-sen University

Issued by Graduate School, HKBU

A Redundancy-Aware Length Control Framework for Extractive Summarization

LI Shuxin

A thesis submitted in partial fulfilment of the requirements

for the degree of

Master of Philosophy

Principal Supervisor:

Prof. LIU Jiming (Hong Kong Baptist University)

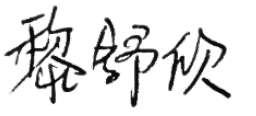
Prof. SU Weifeng (BNU-HKBU UIC)

August 2021

DECLARATION

I hereby declare that this thesis represents my own work which has been done after registration for the degree of MPhil at Hong Kong Baptist University, and has not been previously included in a thesis or dissertation submitted to this or any other institution for a degree, diploma or other qualifications.

I have read the University's current research ethics guidelines, and accept responsibility for the conduct of procedures in accordance with the University's Research Ethics Committee (REC). I have attempted to identify all the risks related to this research that may arise in conducting this research, obtained the relevant ethical and/or safety approval (where applicable), and acknowledged my obligations and the rights of participants.

Signature: 

Date: August 2021

Abstract

While extractive summarization is an important approach of the NLP text summarization task, redundancy in the generated extractive summary is a big problem. Previous works usually set the length of the output summary to a fixed number, which might be appropriate for some of the documents while too long for others. At the same time, though extractive summarization possesses high readability as it directly selects sentences from the document, the unimportant parts within sentences are also selected. We propose a length control framework for extractive summarization, named *LenC*, in a two-stage pipeline to reduce the redundancy in the output. We first use a pretrained BERT-based summarizer to select smaller units (i.e., EDUs) than original sentences to abandon the insignificant parts of a sentence. Then a light-weighted length controller which could be attached to any summarization model is implemented to prune the output summary to an appropriate length. Experiments show that the proposed model outperforms the state-of-the-art baseline models and successfully reduces the redundancy in the extractive summaries.

Keywords: Single document Summarization, Redundant Information, Length Control

Acknowledgements

Firstly, I particular appreciation to my supervisor Prof. Weifeng Su, who guided me to study for a master's degree, which allowed me to further develop new knowledge in the field of natural language processing in artificial intelligence. I am also grateful cordially to my co-supervisor Prof. Jiming Liu for his guidance during the six months of studying at HKBU. I also learned from him the spirit of being a scientific researcher.

Then, I would like to give my special thanks to my fellows, my friends, and my family. I am grateful to the NLP study group for sharing knowledge and providing opinions for my works; I am grateful to my friends for always being by my side to encourage me and help me; I am grateful to my family for not giving me pressure, but for giving love silently.

Lastly, I thank myself for not giving up. During my study, I realized that research on artificial intelligence is not as "intelligent" as I thought. A lot of "dirty works" make me realize that research requires patience and insight. I hope that I can further use the valuable knowledge I learned during graduate school to contribute my little power to the application of artificial intelligence.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

It is the era of information explosion today. Hundreds of millions of gigabytes of new data are generated on the Internet every day, and the amount of new Internet data is greatly increasing every year. It is unrealistic to process such huge quantity of data by manual or traditional data processing tools, so artificial intelligence (AI) is also closely followed. Standing at the forefront of the era of big data, many new AI tasks have emerged, including the application of deep learning in neural language processing. Language is the basic and main way of human communication. Therefore, in the era of big data, the processing of text data is particularly important.

In this chapter, we first introduce the background and the related works for text summarization task in neural language preprocessing, and then we explain the motivation and contributions of this research.

1.1 Background

Neural Language Preprocessing (NLP) tasks can be divided into five categories: Lexical Analysis, Sentence Analysis, Semantic Analysis, Information Extraction, and High-level Tasks. Lexical analysis is the basic work of NLP. It aims to preprocess the neural language on the word level, such as word tokenization and morphological analysis. Sentence analysis, as it implies, is about sentence-level analysis, it contains syntactic parsing, language identification, sentence boundary detection, and so on, which is mainly about the analysis of sentence structure. Semantic analysis is to analyze the understanding and expression of a word in context, tasks about semantic analysis include word sense disambiguation, word embedding, semantic role labeling, and so on. Information extraction intends to extract structured information from unstructured text, for instance, sentiment analysis, intent detection, and named entity recognition.

High-level tasks can be constructed based on the first four NLP tasks, and are mainly implemented by deep learning. There are 6 high-level tasks in NLP: Machine Translation, Text Summarization, Question-Answering System, Dialogue System, Reading Comprehension, and Automatic Essay Grading.

Text summarization aims to compress long texts into a short version summary. In the field of NLP, the study of automatic text summarization has become very important. It helps to find related documents [21] and maintaining useful information while reducing the manual workload. Text summariza-

tion can be divided into *single-document summarization* and *multi-document summarization* according to the input type. Single-document summarization generates summaries from one document, and multi-document summarization generates summaries from a set of topic-related documents. According to the output type, it can be divided into *abstractive summarization* and *extractive summarization*. Abstractive summarization generates summaries word by word and allows using the word out of the input documents, possessing more flexibility. Extractive summarization selects the sentences/phrases from the original article directly, maintaining better readability usually. According to the existence of supervision data, it can be divided into supervised summarization and unsupervised summarization. In this research, we focus on single-document, supervised, extractive summarization.

1.2 Motivation

The abstractive methods usually generate summaries with high Rouge scores because they usually entail larger models. Their summaries usually have less readability and may contain grammatical errors. In contrast, extractive methods usually generate summaries with better readability and fewer grammatical errors.

Extractive summarization usually extracts a whole sentence as part of the output, so it has problems with verbose information inside a sentence. To solve the problem of redundancy, we preprocessed on the sentence level. Through

syntactic analysis, we first divide the sentence into multiple language units and then input each language unit as a separate sentence into the BERT-based model in the format of BertSum. This task is regarded as a sequence labeling task, and the model judges whether the current sentence should be selected as a summary based on sentence expression. According to our observation, the extractive summarization model usually outputs a fixed number of sentences, but not all articles need the same length of summaries, and extracting too many sentences is unnecessary for some articles. Therefore, to further reduce the redundancy of extracting summaries, we built a framework for the extractive model to control the output of the number of sentences based on the characteristics of the article.

1.3 Contributions

In this paper, we propose a two-step framework, **Len**gth **C**ontroller for extractive summarization (**LenC**), which can identify the appropriate length of summary for an article. To furthermore compress the redundancy, we train and evaluate our model on a segmented CNNDM dataset preprocessed by Xu et al. [35]. The *LenC* outperforms the state-of-the-art models with Rouge-1, Rouge-2, and Rouge-L scores to our knowledge.

Our contributions are list as follows:

- We investigate the correlation between the content of an article and the length of the summary for it. we notice that the gap between the length

of the oracle summary and the output summary might lead to a bad result for the extractive summary.

- We propose a light-weighted and portable length controller for the extractive summarization model. It can output a proper size summary for an article while preserving the good performance of the summarization model. At the same time, it can be easily attached to any extractive model.

- We treat EDUs as separate sentences and input EDUs directly into BERT. We discover that a BERT-based sentence encoder can also capture features very well on the discourse-level texts, even better than on the sentence-level one.

Chapter 2

Related Works

In the following, we will have a literature review for the extractive summarization task.

2.1 Extractive summarization

The encoder-decoder framework is the major stream for extractive summarization. Various of encoders (e.g. RNN [41, 33], Transformer [31, 9] and CNN [38, 39]), have been adopted to generate representations of sentences/units or articles. With the feature embeddings from the encoder, the decoder extracts summaries from the document by two approaches, sentence labeling [2] and sequential scoring [41]. In addition to the different kinds of encoder-decoder models above, reinforcement learning models have also been applied in extractive summarization via ranking sentences [20, 7, 40, 36]. However, the performance of the reinforcement learning model is usually worse than the

encoder-decoder in text summarization, which might be attributed to the difficulty for reinforcement learning to learn the semantics relationship between sentences.

2.2 BERT-based model

BERT [6] refers to the Bidirectional Encoder Representations from Transformers [30]. It is a landmark advancement of NLP and has lots of applications including text summarization. Different from the traditional word embedding models such as Skip-gram [18], CBOW [13], and GloVe [25], BERT can generate dynamic word embeddings according to the input context. To transform word embeddings into sentence embeddings, a hierarchical Transformer modeled as a document-level encoder is demonstrated by Zhang et al. [37]. Zhong et. al [40] used BERT to generate sentences embeddings once a time. Additionally, BERTSUMEXT [15] is the first model using BERT for multi-sentence encoding, making BERT output multiple sentence embeddings at once. In our model, we apply a BERT-based multi-sentence encoder proposed by BERTSUMEXT in our summarizer.

2.3 Redundancy-awareness

We have summarized the following 3 situations that cause the redundant information: One is the selection of similar sentences, the second is verbose

information inside a sentence, and the last one is the selection of top-k sentences.

To prevent selecting two similar sentences, some research work on selecting a sentence by measuring its similarity with the previous selected sentences. Zhou et al. [41] calculates the ROUGE gains for each sentence in the articles that can give to the previous selected sentence. For reinforcement learning, Dong et al. [7] construct a neural network training via the policy gradient reinforcement learning to output a vector of self-defined *sentence affinities* for sentence selection.

To solve the overlap and irrelevant problem inside a sentence, the compressive models like JECS [34] learn sentence syntactic relation to further delete useless information inside the selected sentences. Xu et al. [35] divides the sentence into EDUs based on RST (Rhetorical Structure Theory) [16] and extracts EDUs as the basic units for summaries.

Nevertheless, the foregoing models continue to output sentences until a length limit is achieved, which means that the redundancy might be added back to the summary. Inspired by the previous works, we employ our model on a discourse-level dataset and further reduce the redundancy from the output summary by controlling the output length to cut off the unimportant messages.

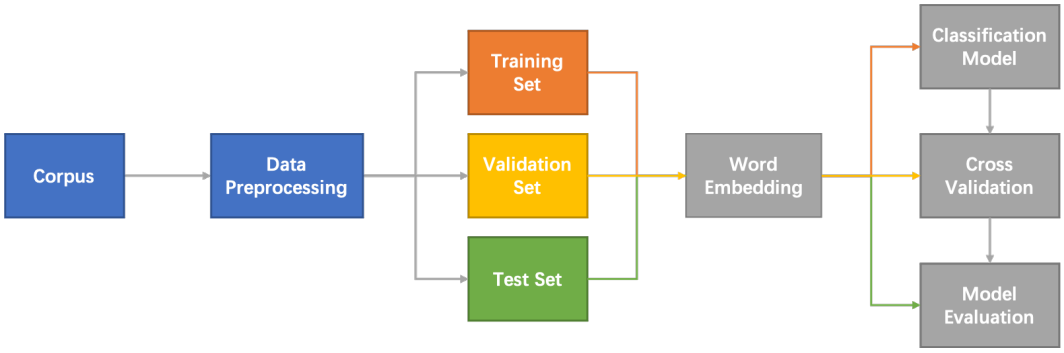


Figure 2.1: Flowchart for extractive summarization based on deep learning

2.4 Two-stage summarization

In two-stage summarization, a model consists of two sub-tasks to generate a summary from the text. Inspired by NeuSum [41], a two-stage summarization model AREDSUM [4] builds up a redundancy-aware model by learning sentence salience firstly and then selecting new sentences based on the salience and redundancy. The *LenC* is also a two-stage extractive text summarization framework. At its first stage, the EDUs are ranked by the summarizer. And at its second stage, they are selected by the length controller.

2.5 Dataset

Figure 2.1 describes a basic flowchart for extractive summarization in deep learning. The first step to build an extractive summarization model is to choose a dataset/corpus which is suitable for the task. There are 6 commonly used English corpora for text summarization (see Table 2.1 for details):

- **DUC/TAC [22, 23]** is an English text summarization dataset published

by DUC annually from 2001 to 2007. From 2003, the published datasets are mainly oriented towards multi-document English summarization. So for single document text summarization, the DUC-2002 corpus is generally used. Due to the small size of the dataset, the neural network model is usually trained on other datasets and then tested on the DUC dataset.

- **Gigaword [11]** was first used by Rush et al., 2015 [27]. It consists of the first sentence of the article and the title combined with heuristic rules.

- **Multi-News [10]** is the first large-scale multi-document summary dataset for news. It consists of news articles collecting from a diverse set of news sources and human-written summaries of these articles from the site newser.com. Each article-summary pair consists of 2-10 source documents and a human-written summary.

- **NYT [28]** contains articles published in New York Times between 1996 and 2007, with abstracts written by experts. The summary of the dataset is sometimes not a complete sentence, and the length is short, about 40 words on average.

- **X-SUM [19]** is a summary dataset that does not support extraction strategies, so abstractive modeling methods are required. The purpose of this dataset is to create a short one-sentence news summary. The data is composed of online articles collected from the BBC.

- **CNNDM [12]** is widely used currently, which is a multi-sentence sum-

Table 2.1: Overview of common English datasets used in text summarization.

Dataset	Input type[a]	Output type[b]	Data size	How to obtain
DUC/TAC	Single/Multi	Extract/Abstract	< 1000/issue	Apply online
Multi-News	Multi	Extract/Abstract	$\approx 60K$	For Free
CNNDM	Single	Extract/Abstract	$\approx 300K$	For Free
X-SUM	Single	Abstract	$\approx 200K$	For Free
NYT	Single	Abstract	$\approx 650K$	Paid application
Gigaword	Single	Abstract	$\approx 4M$	Paid application

[a] whether the dataset is suitable for single-document or multi-document summarization.
[b] whether the dataset is suitable for extractive or abstractive summarization.

mary dataset and is often used to train on single-document summarization. The data consists of online news articles from CNN and Daily Mail. There are two versions of this dataset, the anonymized version, and the non-anonymized version. The non-anonymized version includes real entity names. The anonymized version replaces entities with specific indexes. The non-anonymized version of the CNNDM dataset is applied in this study for model training and evaluation.

In machine learning, a dataset is usually divided into three parts: training set, validation set, and test set. The training set is used to construct the model. The validation set assists in the construction of the model, which is used to evaluate the model during the construction process, providing an unbiased estimate for the model, and adjusting the hyper-parameters of the model to find out the best performance. The test set does not participate in the model construction process It is only used during the evaluation for the performance of the final trained model. In order to divide these three sets, various methods such as the hold-out method, K-fold cross-validation, and

bootstrap resampling are provided. The CNNDM dataset used in this study contains 287,227/13,368/11,490 examples for training, validation, and testing. We follow this standard for model training and validation as in other papers.

Chapter 3

Methodology

This chapter introduces the fundamental knowledge needed in the text summarization task and describes how our extractive summarization model is built in order based on the flowchart of the extractive summarization learning in Figure 2.1. The flowchart for our model is described in Figure 3.1

3.1 Sentence Segmentation

In order to eliminate the redundancy inside a sentence, we also apply sentence segmentation in this study. Sentences are segmented based on relations between clauses.

3.1.1 Universal Conceptual Cognitive Annotation [1]

We have tried to use Universal Conceptual Cognitive Annotation (UCCA) to perform semantic analysis on sentences and to segment and reorganize

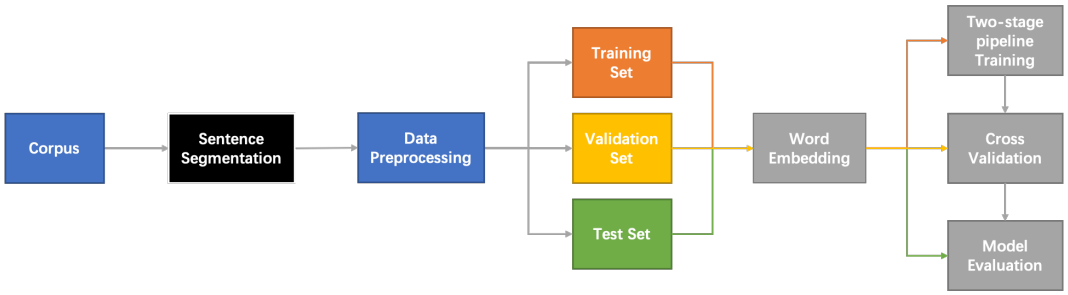


Figure 3.1: Flowchart for extractive summarization learning in LenC

sentences into multiple *scenes* logically. UCCA uses directed acyclic graphs (DAGs) to represent its multi-layered semantic structures (see Figure 3.2). Each *scene* is expected to be a complete grammatical sentence, for example, "*Hip-hop became a passion for David: he took daily lessons and danced very well, reaching the International Hip-hop Championship.*" is expected to be divided into four *scenes*, "*Hip-hop became a passion for David.*", "*He took daily lessons.*", "*He danced very well.*" and "*He reaching the International Hip-hop Championship.*". Unfortunately, the results were not as good as expected. Probably because of the disagreements among the annotators in the definition of some relations while constructing the UCCA-Annotated Corpus, which leads to problems with the accuracy of the results from the UCCA model. The reorganized *scenes* segmented from sentences showed various problems, such as incorrectly supplemented subjects, syntactic errors, missing sentence elements, etc, which led to the poor performance of the extractive summarization model. So we found another way to analyze the structure of the text.

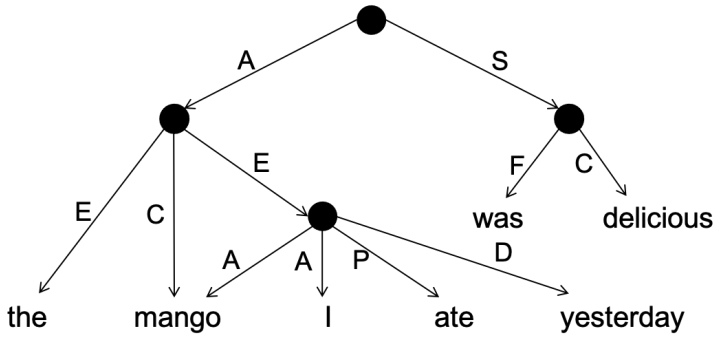


Figure 3.2: Example of UCCA annotation graph.

3.1.2 Rhetorical Structure Theory [16]

Rhetorical Structure Theory (RST) is one of the most popular text analysis and text processing theories. RST defines various rhetorical relations between clauses based on the author's purpose and the author's assumptions about the reader, providing a framework for analyzing the text. A compound sentence can contain multiple relations. Different from UCCA annotating relations at the word level, each relation in RST is defined between two non-overlapping text spans (see Figure 3.3). The spans are usually in *Nucleus::Satellite* relations, such as Background, Elaboration, Condition. There are also *Multi-nuclear* relations, such as Contrast, Joint, List. The nucleus contains the most important and basic information, and the satellite is usually additional information or supplementary explanations of the nucleus. Satellites without a nucleus are usually incomprehensible, but after deleting the satellite, the text can still express its meaning to a certain extent.

In this study, by following the method for sentence segmentation in the CNNDM dataset by Xu et al., 2020 [35], sentences are segmented into mul-

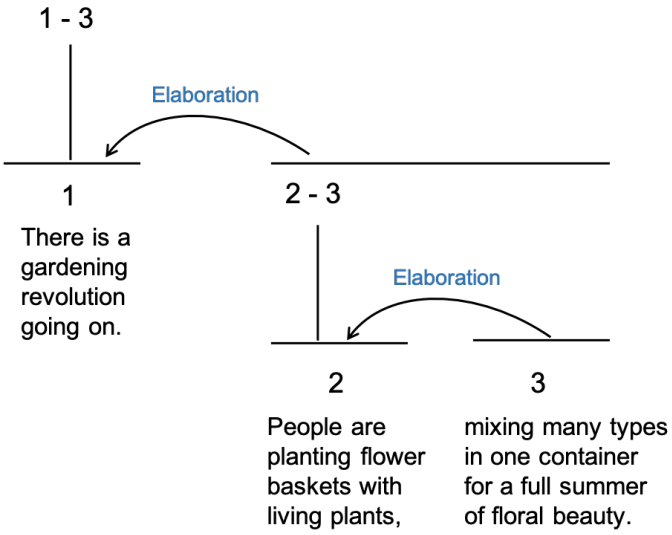


Figure 3.3: Example of RST diagram.

tiple language units, called elementary discourse units (EDUs), through the syntactic structure based on the RST Treebank. For example, "*As previously reported, member of the Philippines' House of Representatives has sued to stop the plant.*" will be divided into 3 EDUs: "*As previously reported,*", "*a member of the Philippines' House of Representatives has sued*" and "*to stop the plant.*". Compared with UCCA, the labeling of relations between clauses in RST Treebank is relatively reliable, so the error rate of EDU segmented by segmenter trained with RST Treebank is also relatively low. The segmented CNNDM dataset is mentioned as the Disco-CNNDM dataset in the following.

3.2 Data Preprocessing

Data Preprocessing is a crucial part of machine learning. How the data is processed affects the results of the model to a large extent. Appropriate data

processing methods further enhances the effect of the model. On the contrary, using improper processing methods leads to a worse model. In this study, the text data are preprocessed with the following methods before being fed into the model:

3.2.1 Document slicing

To facilitate the shuffling of the data while training the model, the articles in the training set are sliced into multiple files. Articles are distinguished from each other using some markers, such as blank lines, specific symbols, etc. In this study, the training set of the CNNDM dataset is divided into 144 data files, each with 2000 articles, and these 144 files are loaded to train the model in random order.

3.2.2 Tokenization

Tokenization is an essential step for data preprocessing in NLP tasks. In NLP tasks, neural network models are trained and predicted with the help of a lexicon for the representation of sentences. Tokenization aims to accurately segment sentences as well as words and punctuation marks within them. It helps the construction of the lexicon of the dataset and prepares the separated words for the follow-up operations, such as stopword removal, stemming, lemmatization, word frequency statistics, and word vectorization. Traditionally, a lexicon is constructed by first subdividing individual sentences into words and then

counting and selecting the top N words with the highest frequency to form the lexicon.

Byte Pair Encoding [29]

Since in English grammar, suffixes is used to indicate the different lexical and tense forms of a word, such as “looked”, “looking” and “looks”. Therefore, the English lexicon will become very large. Sennrich et al. first introduced the Byte Pair Encoding (BPE) algorithm (a data compression algorithm) to NLP in 2015, enabling the above three words to be split into “look”, “ed”, “ing” and “s”. This method separates the meaning of the word itself from the suffixes, effectively reducing the number of word in vocabulary and reducing the occurrence of Out of Vocabulary (OOV) during the model testing.

WordPiece [32]

The WordPiece tokenization used for the input to BERT is based on the BPE algorithm. The main difference between WordPiece and BPE is that WordPiece generates a new subword based on the highest probability of improving the model rather than the next highest frequency byte pair. WordPiece means to break a word into pieces. Using WordPiece tokenization allows BERT to store only 30,522 words when processing English text, and it rarely encounters OOV words. The subword model effectively accelerates the learning of NLP tasks and improves the semantic distinction between different words.

3.3 Word Representation

Machines cannot read text, so the text needs to be transformed into a form that is computable before being fed into the model. In Chapter 3.2.2, sentences are divided into words after data preprocessing, and these words are transformed into structured data that can be processed by the model through vectorization, which is called word representation.

3.3.1 One-hot encoding

The traditional one-hot encoding uses a unique one-hot vector to represent each word, which means that only one word can be stored in each dimension. An English lexicon typically consists of hundreds of thousands of words, so hundreds of thousands of sparse vectors are needed to represent each word when using one-hot encoding. One-hot encoding not only requires a vector of huge dimensions but also cannot encode semantic similarities between words due to the independence between dimensions, making it difficult to perform fuzzy matching between them. Therefore, we need to find a better way for word representation.

3.3.2 Word embedding

Embedding refers to the mapping of one-dimensional vectors into another dimensional space. In 2001, Bengio et al. formally proposed the Neural Network Language Model (NNLM) [3] to implement a word prediction task. While

learning the language model, the model also obtained word embeddings, which could be regarded as the incidental output of the language model. In 2013, Tomas Mikolov proposed two language models for training word embedding, the CBOW, and Skip-gram [18], and open-sourced the *Word2vec* toolkit. The word embedding is context-independent. In other words, the same word always corresponds to the same word embedding in different contexts, which leads to the lack of word sense disambiguation (WSD). The word "*bank*" in "*I deposited the money in the bank.*" and "*The ship has many oars in each bank.*" should not have the same embedding.

ELMo [26] and BERT [6] are proposed to solve the WSD problem. They dynamically adjust the word embedding to the context. ELMo is less efficient since it is impossible to operate in parallel via LSTM. BERT is a sentence-level language representation model based on Transformer encoders, which allows for deeper layers and better parallelism. BERT further increases the generalization ability of the language representation model to output word embeddings that fully reflect the word-level, sentence-level, and even inter-sentence relationships. In our model, we used the BERT as our encoder to generate the representation for EDUs (see Chapter 3.5.1 for details).

3.4 Encoder-Decoder Framework

Deep learning is a process of automatic feature engineering using neural networks. It allows the acquisition of feature representations of higher complexity

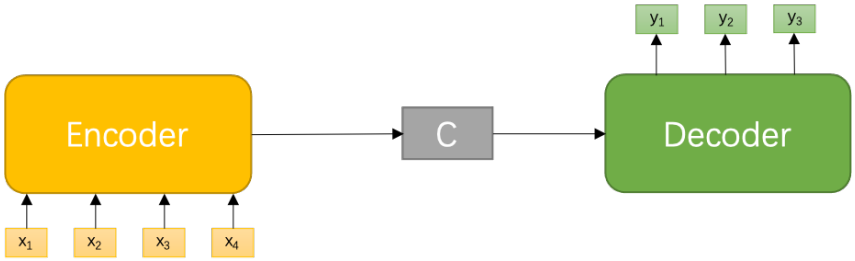


Figure 3.4: Overview of Encoder-Decoder Framework

and dimension than traditional machine learning, thereby increasing the accuracy of the output.

Traditional extractive summarization methods use graphs or clustering to accomplish unsupervised summarization. With deep learning techniques, extractive text summarization is often modeled as sequence labeling task and sentence ranking task. It is usually built using the Encoder-Decoder framework, shown as Figure 3.4. The encoder turns a variable-length sequence into a fixed-length vector C, and the decoder turns this fixed-length vector into a variable-length sequence of the target.

Recurrent Neural Network (RNN) has been a commonly-used network to encode sequence data. However, due to the long-distance dependency problem, RNN has lost a significant amount of information by the time the word is input at the last time step, which means the semantic vector generated by the encoder also loses a lot of information. Since then, attention mechanism has been incorporated into NLP, and BERT was subsequently proposed and became a popular encoder in NLP tasks.

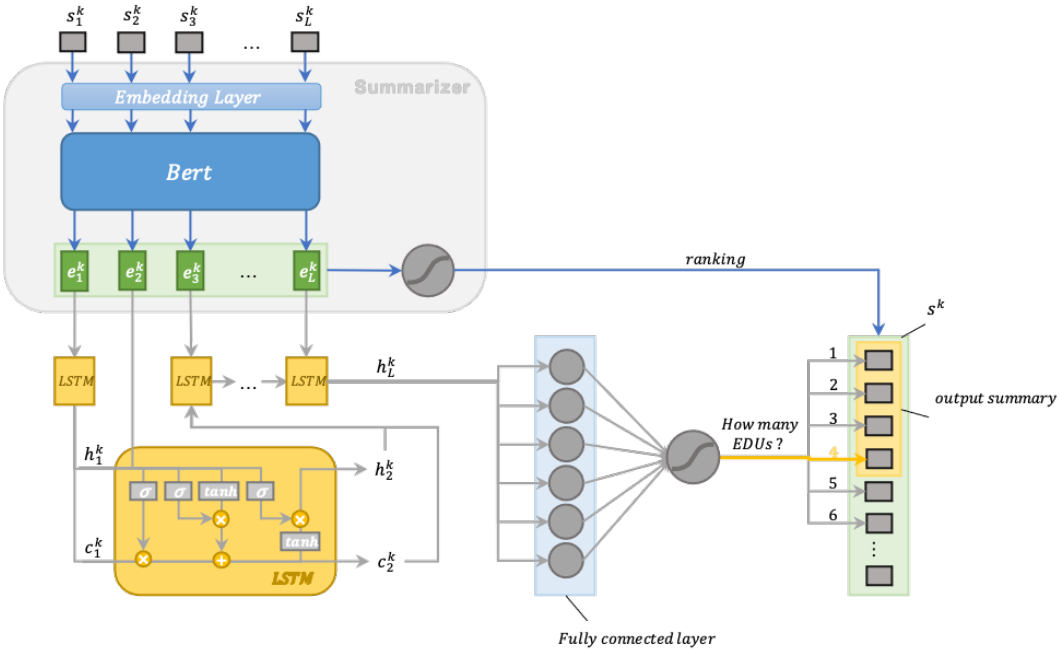


Figure 3.5: The two-stage pipeline of *LenC*. Each s_i^k is wrapped with $[CLS]$ and $[SEP]$. Embedding layer processes texts with token embedding, interval segment embedding, and position embedding. A BERT-based summarizer is illustrated in the gray box. The length controller is attached to the discourse-level embeddings from the BERT-based encoder to control the output number of EDUs from the ranked s^k.

3.5 LenC Framework

LenC is a two-stage summarization model with a light-weighted controller to control the length of an output summary. A summarizer is first trained. Then a length controller determining the length of the output summaries is attached. Fig. 3.5 shows the overall structure of our proposed framework. We use a BERT-based summarizer to output the probabilities of each EDU being selected in the summary, and a length controller determines the output length of a summary depends on the document embeddings from an LSTM layer.

3.5.1 Summarizer

Inspired by BERTSUMEXT proposed by Liu et al.[15], a BERT-based multi-sentence encoder with a simple classifier was adopted as our summarizer to select sentences. Based on the vanilla BERT, BERTSUMEXT inserts a "$[CLS]$" token in front of each sentence of the article to change the input format of BERT from "$[CLS]sent_1[SEP]sent_2[SEP]$" to "$[CLS]sent_1[SEP][CLS]sent_2$ $[SEP][CLS]sent_3[SEP]$". To provide positional information between sentences, BERTSUMEXT assigns interval segment embeddings to each sentence, so that BERT outputs "$[CLS]$" tokens as sentence representations.

We fine-tune BERT with the pretrained "bert-base-uncased" model on discourse-level by using segmented Disco-CNNDM dataset provided by Xu et al.[35]. In our model, EDUs in a document D_k are treated as separated sentences $\{s_1^k, s_2^k, s_3^k, \ldots, s_L^k \mid s_i^k \in D_k\}$, instead of nested units in the sentence. EDUs are embedded with both token and positional information before inserting into BERT. The EDU representations are processed by BERT as:

$$\{e_1^k, e_2^k, e_3^k, \ldots, e_L^k\} = BERT(Embed(\{s_1^k, s_2^k, s_3^k, \ldots, s_L^k\})) \tag{3.5.1}$$

The summarizer is constructed as a sequential labeling model, which can be seen as a binary classification. The probability of the i-th EDU s_i^k being selected to the summary is predicted as:

$$\hat{Y}_i^k = \sigma(W_{sum} e_i^k + b_{sum}) \tag{3.5.2}$$

, where Y^k is the ground truth label for summary referred to document D_k. Binary cross-entropy (BCE) is applied to calculate the loss between Y^k and $\hat{Y}^k$ as

$$loss_{sum}(\hat{Y}^k, Y^k) = -\frac{1}{L}\sum_{i=1}^{L}[Y_i^k log(\hat{Y}_i^k) + (1 - Y_i^k) log(1 - \hat{Y}_i^k)] \tag{3.5.3}$$

3.5.2 Length Controller

Our length controller can be attached to any pretrained extractive summarizer modeled as sequential labeling, sentence scoring, or ranking. It is implemented with an LSTM and a classification layer to finally determine the length of the output summary extracted by the pretrained summarizer. The model reads a matrix of EDU embeddings $e^k \in \mathbb{R}^{1 \times L \times M1}$ from BERT into the LSTM layer to obtain a matrix of document embedding $d^k \in \mathbb{R}^{1 \times \frac{M}{2}}$ containing the context information of an article. Document embedding d^k is obtained as follow,

$$h_i^k, c_i^k = LSTM(e_{i-1}^k, h_{i-1}^k, c_{i-1}^k) \tag{3.5.4}$$

[1]M is the dimension of the output size of BERT, M = 768 in this paper.

$$d^k = h_L^k \tag{3.5.5}$$

Although LSTM is not as good for acquiring long-term dependency as Transformer, it is suitable for classification on the length of text summary to use the feature vector h_L from the last hidden state of the LSTM. In this case, we consider using LSTM a better trade-off other than a two-layer Transformer, since LSTM is much light-weighted.

To obtain the length of a summary from the document embedding, we use a fully connected layer on top of the document vector d^k to output the feature vectors for each category. A Softmax classifier is used to output the probabilities of every candidate category from the feature vectors. We define the truth value $C_k \in \mathbb{R}^{N\times 1}$ as a one-hot vector for the length of the oracle summary of document D_k. $\hat{C}_k$ is predicted from the model as

$$\hat{C}_k^j = Softmax(W_{con}^j d_k + b_{con}^j) \tag{3.5.6}$$

where j represents the j-th category. Cross-entropy is used to calculate the similarity between $\hat{C}_k$ and C_k. Considering the problem of the imbalanced distribution of each category, different weights are assigned to each category, which defines as $\gamma \in \mathbb{R}^{1\times N}$. The weighted cross-entropy loss function is presented as

$$loss_{con}(\hat{C}_k, C_k) = \gamma[-C_k + log(\sum_{j=1}^{N} exp(\hat{C}_k^j)] \tag{3.5.7}$$

3.6 Chapter Summary

The redundancy of extractive text summarization can be solved from two aspects: one is to divide sentences into language units with smaller granularity through data preprocessing; the other is to discard unimportant information in the generated summary by controlling the output of the number of language units. The extractive text summarization task is completed in two steps respectively: one is to model the extractive summarization problem as a sequence labeling task, and the other is to identify the summary length for an article. The model is divided into two parts, the summarizer, and the length controller. The summarizer is an encoder-decoder model with BERT as encoder and a classifier as decoder, which is used to complete the sequence labeling task. The length controller also follows the encoder-decoder framework, using a single layer LSTM as the encoder to encode the content vector of the article, and then using the decoder to classify the summary length for an article via the content vector. The Disco-CNNDM dataset is used to train and evaluate the model, and ROUGE is applied to measure the final performance of the model.

Chapter 4

Experiments

This chapter describes the details of the experiments, the performance of *LenC*, and the analysis of experimental results.

4.1 Baseline Models

The *LenC* is compared with the following five state-of-the-art extractive summarization models:

- **NeuSum(2018)** [**41**] is a hierarchical sequence to sequence extractive model. It uses the MMR method to train an attention scorer to select sentences, aiming to maximize the relative ROUGE gain compared to the previously extracted sentences.

- **BanditSum(2018)** [**7**] treats extractive summarization as a contextual bandit problem. A policy gradient reinforcement learning algorithm is

applied to train the model to select sentences with maximum ROUGE scores.

- **HIBERT(2019)** [**37**] adopts a hierarchical transformer as a document-level encoder and further pretrains on unlabeled data by sentence masking. The model is constructed as a sequential labeling problem for a text summarization task.

- **BERTSUMEXT(2019)** [**15**] applies and fine-tunes BERT to a multi-sentence encoder by simply altering the input format of BERT. It is an extractive summarization model with an inter-sentence Transformer to capture document-level features from BERT.

- **DISCOBERT(2020)**[**35**] formulates a segmented CNNDM dataset with RST relations between semantic units. It adopts BERT-based encoder from BERTSUMEXT for sentence-level encoding and a self-attentive span extractor (SpanExt)[5] for discourse-level encoding.

The first Four output top-3 sentences and train on the CNNDM dataset. DISCOBERT output top-6 EDUs and select EDUs from the Disco-CNNDM dataset.

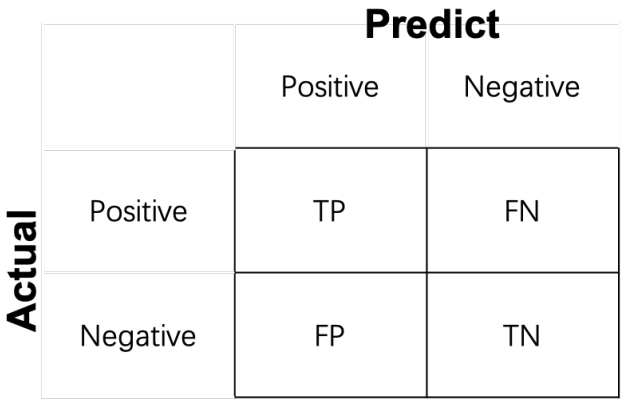


Figure 4.1: Confusion Matrix for Binary Classification. Predict refers to generated text/summary. Actual refers to reference text/summary.

4.2 Experimental Setup

4.2.1 Metrics

After the model is trained, the performance of the model is evaluated with some standard metrics. In the text summarization task, there are two commonly used evaluation methods: one is to calculate the text-similarity; the other one is human evaluation.

Text similarity evaluation

Text similarity uses the string co-occurrence and overlap as a measure of similarity. It includes Hamming distance, longest common substring (LCS), n-gram, etc. The calculation of Hamming distance between texts indicates the number of characters that need to be replaced to transform a string into another string. The accuracy of n-gram similarity, the accuracy of uni-gram can be used to measure the accuracy of word-level prediction, and the accuracy

of higher-order n-grams can be used to measure the fluency of the predicted sentences.

The generated text's string prediction problem for reference text can be regarded as a binary classification problem. The predicted results can all be attributed to the following four scenarios: True Positive (TP), False Positive (FP), False Negative (FN), True Negative (TN). Their relationship can be visualized as a confusion matrix shown in Figure 4.1. There are two important concepts about binary classification prediction, one is *precision* and the other is *recall*. It can be seen from Equation 4.2.1 that *precision* is from the perspective of the prediction result to calculate the ratio of the prediction accuracy. And *recall* (see Equation 4.2.2) is to calculate how many true values are recalled from the perspective of true values.

$$Precision = \frac{TP}{TP + FP} \tag{4.2.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2.2}$$

For automatic evaluation, the common evaluation metrics are BLEU and ROUGE, both of which have some drawbacks but are still mainstream evaluation metrics for NLP tasks. BLEU (Bilingual Evaluation Understudy) [24] was the first machine translation evaluation metric to be proposed. BLEU analyzes the co-occurrence of n-grams in the generated translation and the

reference translation. The higher the number of overlaps, the higher the translation quality. However, BLEU focuses on precision and ignores recall. As a result, it is biased towards shorter translation results which leads to discarding some important information.

$$ROUGE - N = \frac{\sum_{S \in \{Reference\}} \sum_{gram_N \in S} Count_{match}(gram_N)}{\sum_{S \in \{Reference\}} \sum_{gram_N \in S} Count(gram_N)} \tag{4.2.3}$$

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4.2.4}$$

ROUGE (Recall-Oriented Understudy for Gisty Evaluation)[14] is another commonly-used metric in the NLP field to evaluate based on the similarity measure of recall between reference and generated summary on different scales. ROUGE has three types of metrics based on the level of granularity: ROUGE-N, ROUGE-L, and ROUGE-S. ROUGE-N in Equation 4.2.3 calculates the ratio of n-grams of reference summary recalled in the generated summary, but the F1-score (see Equation 4.2.4) of ROUGE-N is usually used to evaluate the quality of the generated summary. ROUGE-L is about the longest common subsequence between reference and generated summary. ROUGE-L does not require continuous matching, and it reflects the sequential matching of word order at the sentence level. ROUGE-S uses skip-grams, which means that when matching reference summary and generated summary. The grams do not need

to be consecutive and several words can be skipped in between. For evaluating the quality of the generated summary from the model, the commonly used ROUGE metrics calculated by F1-score are uni-gram, bi-gram, and the LCS, which are called Rouge-1, Rouge-2, and Rouge-L, respectively[1].

Human evaluation

Owing to the shortcomings of the similarity calculation (e.g., inability to identify synonyms, semantic similarity, etc.), human evaluation is also used in many studies for further clarification. A study usually employs three or more Turkers from the Amazon Mechanical Turk platform, who compare the generated summaries of dozens of articles sampled from the model output with the manual abstracts, and make a fair judgment on multiple aspects (e.g., coherence, grammaticality, and the overall preference). However, the manual evaluation metric is mostly customized and not as intuitive and generalizable as the similarity metric. So it is optionally used as a subsidiary evaluation after the automatic metric evaluation.

4.2.2 Rule-based system

Most gold summaries in the dataset are generated manually. In these summaries, most sentences do not come from the input document directly. However, the original sentences which are most similar to the golden summaries are required during the training process in the extractive summarization model. In

[1]Rouge-1, Rouge-2, and Rouge-L indicate ROUGE in F1-score in this thesis.

general, a rule-based system is entailed for this task. A rule-based system usually consists of two parts, the calculation of the similarity between sentences and the criteria for selecting the sentences that make up the summary.

In this study, we apply ROUGE to calculate the similarity between sentences. To obtain as much information as possible while preserving the sentence structure in the manual summary, we combine the F1-score of both ROUGE-1 and ROUGE-2 to calculate the similarity between texts. We use the greedy algorithm (see Equation 4.2.8) to obtain the sentence with the greatest similarity to the artificial summary as the extracted summary of the article. Although the greedy algorithm does not necessarily obtain a globally optimal solution, it is fast and performs well enough. The algorithm of the rule-based system is as follow:

$$S_0^k = \emptyset, \quad D^k = \{s_1^k, s_2^k, s_3^k, \ldots, s_L^k\} \tag{4.2.5}$$

$$sim(S^k, g^k) = ROUGE_1(S^k, g^k) + ROUGE_2(S^k, g^k) \tag{4.2.6}$$

$$S_{t\_j}^k = S_{t-1}^k + s_j^k, \quad 1 \leq t \leq N, s_j^k \in D^k \tag{4.2.7}$$

$$S_t^k = \begin{cases} S_{t-1}^k + s_i^k, & sim(S_{t\_i}^k, g^k) = \max\limits_{s_j^k \notin S_{t-1}^k} sim(S_{t\_j}^k, g^k) \\ S_{t-1}^k, & sim(S_t^k, g^k) \leq sim(S_{t-1}^k, g^k) \end{cases} \tag{4.2.8}$$

The artificial summary (gold summary) from the CNNDM dataset is defined as g^k. S^k represents a set of candidate EDUs for an article D^k generated via the greedy algorithm, and it is mentioned as the *oracle summary* in the following. According to our examination, the oracle summary S^k for the Disco-CNNDM dataset reaches the best-defined similarity $sim(S^k, g^k)$ with the gold summary g^k while the largest length N for the oracle summary is set to be 6.

4.2.3 Dataset setup

CNN and Dailymail dataset (CNNDM) [12] is one of the widely used benchmark datasets in single document text summarization. The average length of a single sentence in CNNDM is 30.8 words, which leads to a lot of redundant or irrelevant information in the output of the previous summarization models. Therefore, we train and evaluate our model on a preprocessed CNNDM dataset by Xu et al.[35], where each sentence in a CNNDM news article is parsed into an RST discourse tree based on Stanford CoreNLP[17] and segmented into EDUs. Each EDU has 10.6 words on average. The oracle summaries for the Disco-CNNDM dataset are generated by picking out EDUs greedily until the sum of *Rouge-1* and *Rouge-2* drops.

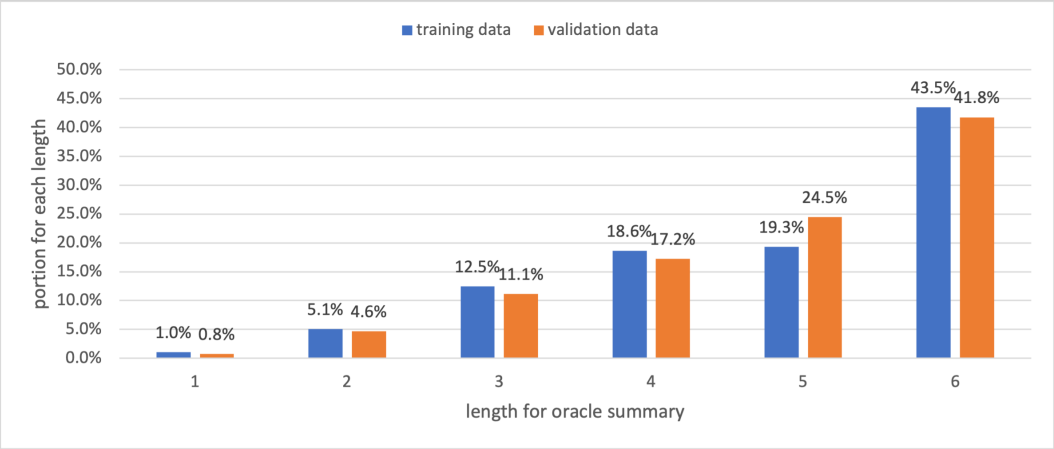


Figure 4.2: Distribution of the oracle length for both training and validation set in Disco-CNNDM.

4.2.4 Data analysis

Since we found that the extractive model fixed the output length of the generated summary might cause redundancy because the oracle summary lengths vary for different documents. Therefore, the analysis of the relations between the length of the oracle and the output summary is conducted. In Fig. 4.2, the distributions of oracle length between the training set and validation set show that documents in these two sets have commonality on the length of the oracle summary. Fig. 4.2 shows that over 40% of the documents have six EDUs in their corresponding oracle summaries. There are still about 60% of them with oracle summary length under 6. If the model outputs 6 EDUs for all documents, the summaries for those 60% documents are likely to be lengthy. Since then, we calculated the average number of EDUs in all the oracle summaries, which is approximately 5.

4.2.5 Summarizer setup

The summarizer is adopted from Liu and Lapata[15][2]. The process of training the summarizer follows the settings in BERTSUMEXT [15]. Owing to the limitation of the input length for BERT, each document is truncated to 512 BPEs. The BERT-based summarizer is fine-tuned with the pretrained 'bert-base-uncased' released by google search[3] on the Disco-CNNDM dataset. The whole summarizer is trained for 20,000 steps on 3 GPUs(Tesla V100) with approximate batch size 56 and warming-up step 5000. The checkpoints are saved for every 500 steps. After evaluating the $LenC$ on the validation set, the checkpoints are preserved with the smallest evaluation $loss_{sum}$ for the second training stage of the length controller.

4.2.6 Length controller setup

Length controller is the second step after the training of the summarizer. When training the length controller, all parameters in the BERT-based encoder of the summarizer are frozen, as we found the model is hard to converge when the parameters are trainable. To ensure the robustness and fault tolerance of the model, we simplified the multi-classification problem into a two-category problem for *longsummary* and *shortsummary* in the experiment. Since the average number of EDUs in all the oracle summaries is about 5 (see Chapter 4.2.4 for details), we use it as a threshold to categorize the document into

[2]https://github.com/nlpyang/PreSumm
[3]https://github.com/google-research/bert

two parts. For simplicity, we use $L(O_k)$ as the length of oracle for document D_k. When $L(O_k) < 5$, O_k is defined as short oracle, otherwise O_k is assigned to be a long oracle. $[1, 0]$ was assigned to C_k for short summary, and $[0, 1]$ for long summary. According to our statistics, documents with a long summary account for 62.8% of the overall training data (see Fig. 4.2 for details), so we assign the weights to the two classes as $\gamma = [1.2, 1]$. An Adam optimizer with the same setting on training summarizer is implemented. We use the learning rate decay schedule from Vaswani et al.[30], according to Equ. 4.2.9,

$$lr = \alpha \cdot min(step^{-0.5}, step \cdot warmup^{-1.5}), \tag{4.2.9}$$

where $\alpha = 0.2$ and $warmup = 10000$. The whole model is trained for 30,000 steps on 3 GPUs(Tesla V100), with about 180 documents processed at each step. We select the top-3 checkpoints with the least evaluation $loss_{con}$ and test the model on the test set. We rank EDUs in document D_k by $p(\hat{Y}_i^k = 1|s_i^k, D, \theta_1)$ and select top-4 EDUs if $p(\hat{C}_k^1 = 1|s_i^k, D, \theta_2) \geq 0.5$, top-6 otherwise. $p(\theta_1)$ is the summarizer, and $\theta_1 \in \theta_2$.

4.3 Experiment Results

Table 4.1 shows the experimental results for all models on the CNNDM dataset using the ROUGE-F_1 score. It also includes two oracles based on both sentence-

Table 4.1: Results on CNNDM test set. F1-score for ROUGE-1, ROUGE-2, and ROUGE-L are reported. Results for models with an asterisk symbol (*) are captured from the corresponding papers. Models with symbol † are trained and tested on Disco-CNNDM.

Model	**R1**	**R2**	**RL**
Oracle	52.59	31.24	48.87
Oracle w. Disco-CNNDM	60.27	36.83	57.73
Lead-3	40.42	17.62	36.67
Lead-6 w. Disco-CNNDM	38.84	16.36	36.40
NeuSum*	41.59	19.01	37.98
BanditSum*	41.50	18.70	37.60
HIBERT-S*	42.31	19.87	38.78
HIBERT-M*	42.37	19.95	38.83
BERTSUMEXT*	43.25	20.24	39.63
DISCOBERT*†	43.38	20.44	40.21
BERTSUMEXT(ours)†	43.43	20.50	41.12
LenC(Summarizer)†	43.45	20.51	41.15
LenC†	**43.60**	**20.59**	**41.26**

level and discourse-level, generated by greedily receiving sentences until the sum of *Rouge-1* and *Rouge-2* decreased or meeting the maximum length limit[4]. Two unsupervised baselines, Lead-3 and Lead-6, are provided. Lead-3 selects the first three sentences from each article in CNNDM as a summary, and Lead-6 chooses the first six EDUs for the Disco-CNNDM dataset. As we see, the Disco-CNNDM dataset leverages the upper bound greatly by 7.68/5.59/8.86 for *Rouge-1/2/L* respectively.

Compared with the models in the middle section of Table 4.1, the *LenC* outperforms all the sentence-based competitors, as well as the discourse-based DISCOBERT, especially on *Rouge-L*, which means that our model captures both the contents and structures among EDUs well. We observe that the

[4]3 sentences for sentence-level oracle and 6 sentences for discourse-level oracle

Table 4.2: The average number of words for the summary of different models.

Model	**Avg. words/doc**
target summary (ground truth)	54
NeuSum	88
BERTSUMEXT	76
DISCOBERT†	71
BERTSUMEXT(ours)†	66
LenC†	**62**

non-BERT-based model is worse than the BERT-based model, which confirms BERT a good choice to obtain sentence embedding.

Xu et al.[35] explore that inputting EDUs in nested $[CLS]$ to BERT leads to a sharp drop in model performance, so they encode document on sentence-level to BERT. Then a SpanExt is used to learn EDU embeddings from a sentence vector. Different from DISCOBERT, we treat the EDUs as separate sentences, making every EDU in a document surround by $[CLS]$ and $[SEP]$. They are then fed to BERT to obtain the EDU embeddings. The improvements on Rouge-1/2/L with 0.18/0.26/1.49 respectively demonstrate that our way to encode EDUs as separate sentences is practical and successful. Especially the improvement in $R-L$ shows that the semantic and syntactic relation among EDUs can be well obtained via the BERT encoder, while hierarchical encoding or nested encoding is not a wise treat. Compared with the original BERTSUMEXT, our way of inputting EDUs as separated sentences improves the BERTSUMEXT, as shown in Table 4.1.

With the help of the length controller, Rouge-1/2/L have further increased

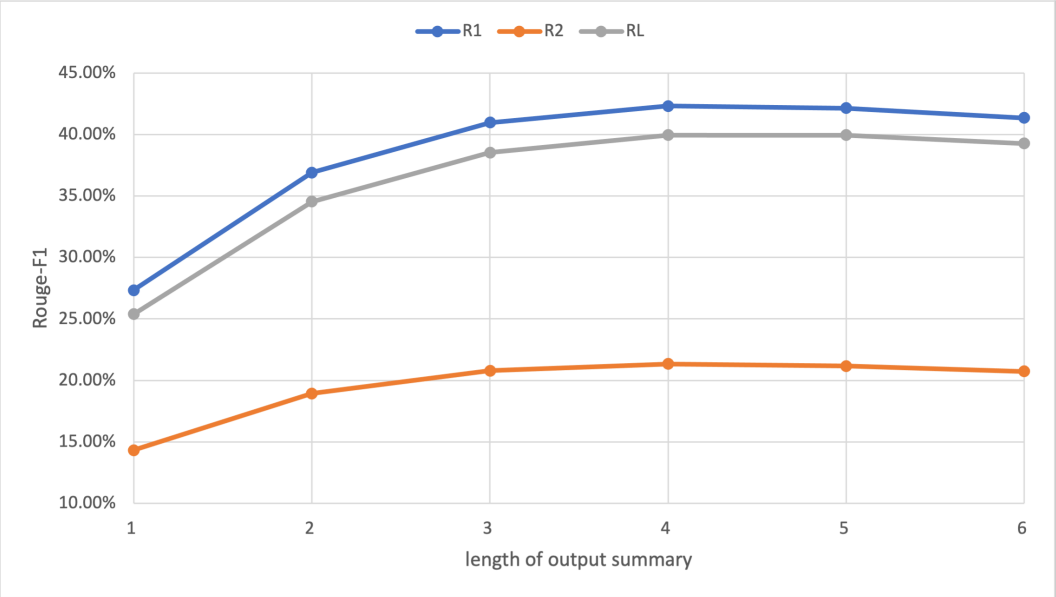


Figure 4.3: Curve of $ROUGE - F_1$ under different output lengths for short summary in validation set

by 0.15/0.08/0.11 respectively. Refer to Table 4.2, *LenC* also outputs the summary with the shortest average word count, which is 62 words per document. Both Table 4.1 and Table 4.2 show that our model not only outputs the shortest summary among the models but also has the best ROUGE score.

4.3.1 Example analysis

We observe that EDUs extracted at the tail of the output summary usually do not provide as much useful information as EDUs selected in the front, especially for documents with a short oracle summary. $Rouge - F_1$ curve with different output lengths of summary for documents with short oracle summaries in Fig. 4.3 indicates that excessive selection of EDUs damages the performance on the previous selected EDUs. In Table 4.3, three examples of summary output from our model show that the main information of articles

Table 4.3: Example of output summaries from LenC for Disco-CNNDM dataset

gold summary
Reanne Evans is bidding to make snooker history. She is three matches away from reaching the world championship. No woman has ever played in the finals at the crucible. Evans's first qualifier is against 1997 world champion Ken Doherty.
output summary
Reanne Evans faces Ken Doherty. In her first qualifier for the world championship. The 29-year-old from Dudley would be the first woman ever to reach the crucible. But she faces a tough start in the form of 1997 world champion Ken Doherty. ~~Reanne Evans, the 10-time ladies' snooker world champion, begins her bid. Should she win three qualifiers.~~
gold summary
Hicks is charged in the deaths of three Muslim college students in chapel hill, North Carolina. Victims ' family members have called on authorities to investigate the slayings as a hate crime.
output summary
That Hicks' case is "death penalty qualified". Who is charged in the deaths of three Muslim college students in chapel hill, North Carolina. Can face the death penalty. The victims ' family members have called on authorities to investigate the slayings as a hate crime. ~~All three were shot in the head. Turned himself in to police the night of the killings.~~

are well captured in the front, and the redundancy at the tail is deleted by the length controller.

Chapter 5

Conclusions and Future Works

In this research, we present a length control framework for extractive summarization, named *LenC*. We build a simple and light-weighted controller to customize the output length for each document We validate our model on Disco-CNNDM and observe that the model achieves its best performance when EDU features are treated as separate sentences and encoded by BERT. Experimental results show that *LenC* successfully reduces the redundancy in the extractive summaries. However, there still exist certain aspects that could be refined in the future.

5.1 End-to-end Model

We have tried the end-to-end approach, but the results are not satisfactory because the model is difficult to converge. Referring to Vision Transformer (ViT) [8], we tried to encode each article with a *[CLS]* token placed in front

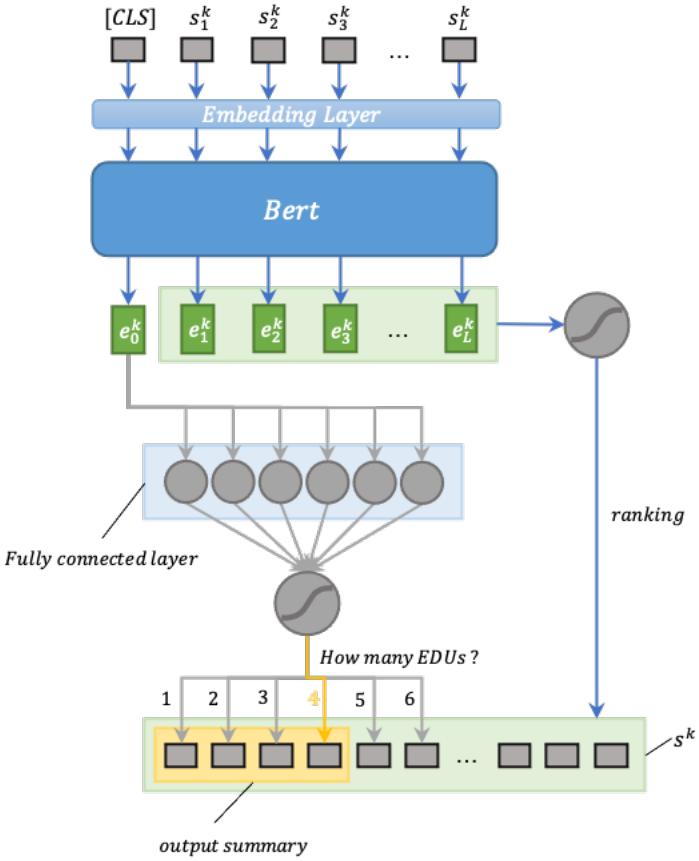


Figure 5.1: Overview of an end-to-end model of *LenC*. *[CLS]* token is placed in front of the input sequence s^k. each sentence representation in e^k goes through a sigmoid classifier to get the probability to be in the output summary. The length representation e_0^k for the first *[CLS]* token will be further "analyzed" in the fully connected layer and a softmax classifier to calculate the length of output summary for the input article.

of the input sequence to decide the length of the output summary, as shown in Figure 5.1. We applied multi-task learning at the time to use the weighted sum of the cross-entropy loss of the two tasks, as the loss of the entire model, but it may be that the correlation between the length classification task and the sequence labeling task is not strong, which makes the model difficult to converge. Multi-task learning may not be the best choice for our model, and the next step should be to find a suitable loss function for this model.

5.2 Loss Function

The adaptation of the end-to-end model for this study should focus on the construction of the loss function. Because the two tasks are not intuitively related and cannot improve the model by helping each other, the design of the loss function is crucial, which determines whether the model can converge or not. Since multi-task learning did not work, the end-to-end model for the next study should focus on the loss of the sequence labeling task, at the same time combine the result of the length classification task in that loss.

5.3 Semantic Segmentation

Since RST can only annotate the relations between two non-overlapping text spans, EDUs segmented from sentences can not be recombined as a sentence with complete structure. The current model extracts EDUs to compose the summary, which suffers from a lack of sentence components such as subjects and objects. Although the use of EDUs effectively reduces redundancy, it affects the readability of the summaries. To ensure the syntactic structure and logic of the summary, it is necessary to use a semantic approach for the sentence segmentation, where UCCA (see Chapter 3.1) might be a better choice. The study could then try to fine-tune the UCCA model to get a more suitable semantic structure for a particular database, and then add a downstream task to output simple sentences segmented from complex sentences.

Bibliography

[1] O. Abend and A. Rappoport. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, 2013.

[2] K. Al-Sabahi, Z. Zuping, and M. Nadher. A hierarchical structured self-attentive model for extractive document summarization (hssas). *IEEE Access*, 6:24205–24212, 2018.

[3] Y. Bengio, R. Ducharme, and P. Vincent. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, pages 932–938, 2001.

[4] K. Bi, R. Jha, W. B. Croft, and A. Celikyilmaz. Aredsum: Adaptive redundancy-aware iterative sentence ranking for extractive document summarization. *arXiv preprint arXiv:2004.06176*, 2020.

[5] X. Cheng, Y. Chen, B. Cheng, S. Li, and G. Zhou. An emotion cause corpus for chinese microblogs with multiple-user structures. *ACM Trans-*

actions on Asian and Low-Resource Language Information Processing (TALLIP), 17(1):1–19, 2017.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[7] Y. Dong, Y. Shen, E. Crawford, H. van Hoof, and J. C. K. Cheung. Banditsum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748, 2018.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] E. Egonmwan and Y. Chali. Transformer-based model for single documents neural summarization. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 70–79, 2019.

[10] A. R. Fabbri, I. Li, T. She, S. Li, and D. R. Radev. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. *arXiv preprint arXiv:1906.01749*, 2019.

[11] D. Graff, J. Kong, K. Chen, and K. Maeda. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34, 2003.

[12] K. M. Hermann, T. Kočiskỳ, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. *arXiv preprint arXiv:1506.03340*, 2015.

[13] T. Kenter, A. Borisov, and M. De Rijke. Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*, 2016.

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

[15] Y. Liu and M. Lapata. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*, 2019.

[16] W. C. Mann and S. 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.

[17] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[19] S. Narayan, S. B. Cohen, and M. Lapata. Don't give me the details, just the summary! *Topic-aware Convolutional Neural Networks for Extreme Summarization. In*, 2018.

[20] S. Narayan, S. B. Cohen, and M. Lapata. Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*, 2018.

[21] N. Nazari and M. Mahdavi. A survey on automatic text summarization. *Journal of AI and Data Mining*, 7(1):121–135, 2019.

[22] NIST. Past data for document understanding conferences, from 2001 to 2007. https://www-nlpir.nist.gov/projects/duc/data.html.

[23] NIST. Past data for text analysis conferences, from 2008. https://tac.nist.gov/data/.

[24] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[25] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[26] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[27] A. M. Rush, S. Harvard, S. Chopra, and J. Weston. A neural attention model for sentence summarization. In *ACLWeb. Proceedings of the 2015 conference on empirical methods in natural language processing*, 2017.

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

[29] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems 30*, jun 2017.

[31] D. Wang, P. Liu, Y. Zheng, X. Qiu, and X. Huang. Heterogeneous graph neural networks for extractive document summarization. *arXiv preprint arXiv:2004.12393*, 2020.

[32] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's

neural machine translation system: Bridging the gap between human and machine translation, 2016.

[33] W. Xiao and G. Carenini. Extractive summarization of long documents by combining global and local context. *arXiv preprint arXiv:1909.08089*, 2019.

[34] J. Xu and G. Durrett. Neural extractive text summarization with syntactic compression. *arXiv preprint arXiv:1902.00863*, 2019.

[35] J. Xu, Z. Gan, Y. Cheng, and J. Liu. Discourse-aware neural extractive text summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5021–5031, 2020.

[36] S. Xu, X. Zhang, Y. Wu, F. Wei, and M. Zhou. Unsupervised extractive summarization by pre-training hierarchical transformers. *arXiv preprint arXiv:2010.08242*, 2020.

[37] X. Zhang, F. Wei, and M. Zhou. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. *arXiv preprint arXiv:1905.06566*, 2019.

[38] Y. Zhang, D. Li, Y. Wang, Y. Fang, and W. Xiao. Abstract text summarization with a convolutional seq2seq model. *Applied Sciences*, 9(8):1665, 2019.

[39] Y. Zhang, J. E. Meng, and M. Pratama. Extractive document summarization based on convolutional neural networks. In *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 918–922. IEEE, 2016.

[40] M. Zhong, P. Liu, D. Wang, X. Qiu, and X. Huang. Searching for effective neural extractive summarization: What works and what's next. *arXiv preprint arXiv:1907.03491*, 2019.

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

CURRICULUM VITAE

Academic qualification of the thesis author, Miss LI Shuxin:

– Received the degree of Bachelor of Science from Beijing Normal University - Hong Kong Baptist University United International College, June 2016.

August 2021