Extractive Text Summarization

25882
39290
26725

Abstract

The recent years have seen remarkable success in the use of neural network on extractive text summarization tasks. In this paper, we seek to gain a thorough understanding of how those summarization models have been improved behind such huge success. Specifically, we build and train a number of extractive summarization models with typical neural network architectures in the process of the model development, and design multiple experiments related to training methods and evaluating metrics, aiming to better understand how neural extractive summarization systems could benefit from different types of model architectures, transferable knowledge and learning schemas. Additionally, we find an effective way to improve the frameworks and achieve the state-of-the-art result on CNN/DailyMail dataset based on our observations and analyses.

Keywords: Extractive Text Summarization; Neural Network Architecture; Sentence Encoder; Document Encoder; Document Decoder; Tansformer

1 Introduction

Text summarization task is often divided into two paradigms, extractive summarization and abstractive summarization. The former one aims at shortening the original text while retaining the key information through selecting sentences from the original text. While in abstractive summarization, target summary contains words or phrases that were not in the original text and usually require various text rewriting operations to generate. In this paper, we primarily focus on extractive summarization since they are more computationally efficient, and can generate more grammatically and coherent summaries compared with abstractive summarization.

The recent years have seen remarkable success in the use of deep neural networks on extractive text summarization. While behind such success, we hope to gain a better understanding of how the performance of those summarization models has improved. Therefore, after carefully studying the relevant research in this field, we sort out the development process of the extractive summarization model in recent years and follow the pace of this development. Specifically, we construct a number of typical extractive summarization models with periodic significance in the development process and design multiple experiments, aiming to seek to better understand how neural extractive summarization systems could benefit from different types of model architectures, transferable knowledge and learning schemas. It is worth mentioning that when evaluating the model, in addition to using the traditional ROUGE metric, we also add some small experiments. For example, we change the threshold used by the binary classifier to obtain the ROC curve to explore the performance of the classifier obtained by our model. Finally, we find an effective way to improve the frameworks and achieve the state-of-the-art result on CNN/DailyMail dataset according to our experimental results.

1.1 Literature Review

Since 2015, the development of the extractive summarization model has mainly revolved around the three-stage structure of the model (sentence encoder, document encoder and decoder): by changing the structure of these three components and using different training methods, the model's performance continues to increase.

There are two main types of sentence encoder: CNN-based encoder and RNN-based encoder. (Yin and Pei,2015[17]) proposed the first CNN-based sentence encoder, applying the convolution layer with the same filter size to convolve the context, and performing max-over-time pooling on the obtained feature map to capture the important semantic information appearing in the sentence as the sentence embedding. (Cheng and Lapata,2016[1]) made improvements on this basis and proposed the convolution layer with different filter sizes. For pooling method, the obtained feature map was first subjected to max-over-time pooling to obtain the representation of each feature, and then all feature representations were performed max-over-feature pooling to obtain the sentence embedding. This is also the structure we adopt when building the CNN-based encoder. Subsequent improvements to this encoder use more complex CNN structures on the one hand, such as the

Convolutional Sentence Tree Indexer (CSTI) proposed by (Singh et al,2018[14]), or use different pooling methods on the other hand, such as the model proposed by (Narayan et al,2018[12]) using concatenating instead of max pooling in the second pooling. For RNN-based sentence encoder, (Nallapati et al,2017[11]) proposed the Bi-GRU sentence encoder for the first time, taking word embedding as input, concatenating the hidden state in both positive and negative directions as the context vector of each token, and then performing average pooling on these context vectors to obtain The sentence embedding of each sentence. This is also the structure of the RNN-based sentence encoder adopted in this paper. As the popular structure adopted by many subsequent models, some variants of this RNN-based encoder include the adjustment of the pooling method[20] and the use of Bi-LSTM to replace Bi-GRU[16]. With the introduction of pre-trained models, the deformation of the sentence encoder is also diversified, such as (Zhong et al,2019[19]) using BERT's token embedding to replace the embedding matrix in RNN-based encoder, and (Liu,2019[8]) using BERT-base to replace the whole sentence encoder.

The document encoder is used to encode the global semantic information of the document. Early models use LSTM/GRU as the document encoder[11]. The main improvement is to change from uni-directional RNN to bidirectional RNN or from single-layer RNN to multi-layer RNNs. This paper adopts single-layer bidirectional RNN encoder as one choice of document encoder. In addition, the popular practice is to use the transformer to replace the document encoder. For example, (Wang et al,2019[16]) use the 5-layer, 4-head transformer to replace the RNN-based document encoder, which is also adopted in this paper when constructing our model. It is worth mentioning that the article also proposed three pre-training tasks suitable for summarization tasks to pre-train the model. While (Liu,2019[8]) comprehensively used the combination of BERT sentence encoder and transformer document encoder, which has gradually become the standard encoder for the current pre-trained extractive summarization model. Considering that there is no support from a powerful computing platform, we gave up using BERT.

The basic structure of Decoder is proposed by (Cheng and Lapata,2016[1]). The hidden state output by the document encoder is classified, by going through Multilayer perceptron(MLP) and Sigmoid function, to whether the current sentence belongs to the summary. This paper also uses this decoder as the baseline. Subsequent improvements to this type of decoder mainly focus on two aspects: one is to interact with more effective information before hidden states being sent to the classifier, and the other is the adjustment of interaction methods. For example, it is proposed in (Narayan et al,2018[12]) that when decompressing a document, in addition to using the text content of the document itself, side information (i.e. title, image caption, table caption and other information) can also be used. Another example is that when interacting, the statistical information of sentences can be introduced. (Nallapati et al,2017[11]) proposed a classifier layer that integrates 5 types of information (content, salience, novelty, absolute position, relative position), which is also the main decoder adopted in this paper. This type of decoder classifies each sentence separately when decoding, and whether other sentences have been selected as summary has no effect on the classification of the current sentence. Especially after the emergence of the pre-trained model, the output of the transformer passed as input to such a classifier layer forms the currently popular decoder.

The extractive summarization generation task is usually regarded as a sequence labeling task. This paper also adopts this traditional task formulation, that is, using the decoder to classify each sentence one by one in the compressed document. However, such a model faces a problem in the prediction process, that is, as the decoder decodes sentence by sentence, the accuracy of the decoder gradually decreases due to various reasons; when the important information in the document is distributed throughout the document or in the latter position, the accuracy of the extractive summary is much lower than that when the important information is distributed in the front position of the document[20]. To this end, (Zhou,2018[20]) proposed a new task formulaization (global selection task). At each step of decoding, the probability distribution on the sentences that have not been selected as summary is calculated and the sentence with the highest probability is selected to be added to the summary; then the selected sentence is used as the input of the next step to compute the probability distribution over the remaining sentences; do so until the threshold is reached. Considering the huge amount of work and many experiments, we did not try this task mode, hoping to continue to explore in the future.

2 Task Formulation

We expect a summary contains less words than the article without losing most of its information. As abstractive summarization method might suffer from failure in factual consistency, we prefer to use extractive summarization to generate summaries. The key feature of extractive summarization is that the generated summary only uses sentences which appear in the article. Thus, we could model the problem of extractive summarization as a sequential labeling problem. Suppose the article contains n sentences $s_1, s_2, ..., s_n$. We define the label for the

summary as $y_1, y_2, ..., y_n$ each taking values either equal to 0 or 1. If y_i equals to 1, it means the i-th sentence is used in the generated summary.

2.1 CNN/DailyMail Dataset

CNN/Dailymail dataset, proposed by Hermann et al.[3], is a large dataset designed for the task of summarization, which contains an article and an abstract in each sample. The article is news which consists of some sentences. The abstract is human-written summary for news. A direct way to get the dataset is to download it from Tensorflow Dataset[1]. We attempt to clean the data by ourselves first, which requires performing sentence segmentation and dealing with special characters. However, we finally turn to another source of CNN/Dailymail dataset, since Zhong et al.[19] has provided a cleaned version of CNN/Dailymail dataset on their Github[2]. This dataset provides three items for a sample: The first item is the segmented paragraph, presented in the form of a list containing many sentences, with sentences being lower-cased. The second item is the segmented abstract, also presented in the form of a list of sentences. The third item is the gold label for the article, which We will explain in the following part. This dataset is pre-splited into training set, validation set and test set. The training set has 254363 samples, the validation set has 13367 samples, and the test set has 11489 samples. In the downloaded file, each sample is stored in a single file, and this size of dataset causes failure to unzip or directly upload so many files into Google Drive. We turn to merging 16 articles in a file, and use the merged data in our research. Following the practice of Zhong et al.[19], we truncate the document to 70 sentences and sentence to 100 words, then pad the documents and sentences to their maximum length.

2.2 ROC&AUC Metric

As described previously, the task is actually a binary classification problem over sentences of a document. A receiver operating characteristic curve, or ROC curve, can be applied to evaluate the performance of our models. Since we have more than 10000 documents in the testing dataset and each document consists of 70 sentences with padding in our settings. If we obtain the ROC curve with all the padding sentences it will lead to a very high AUC score because the padding parts of input and label are actually identical, and positive(label 1) probability of padding sentences will be extremely low. So we only select non-trivial sentences in each document. In the testing set, there are in total 306280 non-trivial sentences. For each model, we concatenate all the labels and corresponding probability of these sentences and plot the ROC curve.

2.3 ROUGE Metric

Recall-Oriented Understudy for Gisting Evaluation, or ROUGE is a package originally proposed by Lin[7] which evaluates the quality of a summary compared to a referenced summary. It is very widely used in evaluating generated summaries in relavant papers[9][11][19]. The calculation of $ROUGE_N$ metric relies on the concept of n-gram, so we introduce n-gram first. n-gram is a contiguous sequence of n words from a given sample of text. An example for the calculation of n-gram is illustrated in Table 1. Suppose a text has N words, the n-gram for this text should include $N - n + 1$ elements. Thus we could calculate the n-grams for model summary generated by our model, as well as n-grams for the referenced summary. The ROUGE-N metric in the original

Original text	"Recall-Oriented Understudy for Gisting Evaluation"
Unigrams(1-gram)	["Recall-Oriented","Understudy","for","Gisting","Evaluation"]
Bigrams(2-gram)	["Recall-Oriented Understudy","Understudy for","for Gisting","Gisting Evaluation"]
Trigrams(3-gram)	["Recall-Oriented Understudy for","Understudy for Gisting","for Gisting Evaluation"]

Table 1: Article, Summary and Gold Summary

paper counts the number of overlapping elements both found in the n-grams of the model summary and n-grams of the reference summary, and divides this number by the total number of n-grams in the reference summary to yield the desired metric. Since it is a recall-based metric, we call it Recall ROUGE-N. Using calculation similar to precision and F1-score in machine learning, we can also derive the formula of Precision $ROUGE_N$, F1 ROUGE-N.

$$Recall\ ROUGE - N = \frac{number\ of\ overlapping\ elements}{total\ elements\ in\ referenced\ summary}$$

[1]https://www.tensorflow.org/datasets/catalog/cnn_dailymail
[2]https://github.com/maszhongming/Effective_Extractive_Summarization

$$Precision\,ROUGE-N = \frac{number\,of\,overlapping\,elements}{total\,elements\,in\,model\,summary}$$

$$F1\,ROUGE-N = 2 \times \frac{Precision\,ROUGE-N \times Recall\,ROUGE-N}{Precision\,ROUGE-N + Recall\,ROUGE-N}$$

Another ROUGE metric in the paper[7] is called $ROUGE_L$. It first counts length of word-level Longest Common Subsequence for two texts, and then calculate precision, recall and F1-score using the number of words in model and reference summary similarly. At the sentence level, the length is calculated between the model sentence and reference sentence. At the summary level, the length is calculated as the sum of the lengths of Longest Common Subsequences between the model summary and every sentence in the referenced summary. ROUGE-L captures sequence-level information for two texts, while ROUGE-N cannot.

ROUGE metric cannot tackle the case when two summaries are using different words with the same meaning. In our case, the generation of gold summaries and evaluation of model summaries could be affected by this drawback, but it is still an effective evaluation metric. We use a Python library naming rouge [3]to calculate the ROUGE metric in our experiment.

ROUGE-1 and ROUGE-2 reflects informativeness, and ROUGE-L reflects fluency. Most papers[9][18][19][11] adopt F1- ROUGE as the evaluation metric. We will report F1 ROUGE-1, F1 ROUGE-2 and F1 ROUGE-L as the evaluation metric for each model.

2.4 Gold Summaries and Gold Labels

We should not expect the original sentence in the news to appear in the summary, since this is not a common way for a human to write summaries by copying sentences from the article. So problem arises that how can we use human-written summary to train our model? One direct way is to maximize the ROUGE metrics between our selected sentences and the human-written abstract. However, though ROUGE metrics truly grant us a way to evaluate the quality of the summary, it is not differentiable with respect to those parameters in our model. So we ought to find other ways. As a common practice in literature[5][9][11], the problem is solved by generating a gold summary full of the sentences in the news by such a procedure:

The initial gold summary is empty. We form our gold summary with a greedy approach. For each step, our sentence set contains all sentences in the article, and also new to the gold summary in this step. We try every sentence in our sentence set, put it into the gold summary at its order in the original text and check the performance of the new gold summary. Among all sentences in the sentence set, the sentence which gives the maximum ROUGE metric comparing to the human-written summary will be put formally into our gold summary. We repeat this procedure until the introduction of new sentence won't increase the ROUGE metric of the gold summary comparing to the human-written one. The summary produced by this algorithm is our desired gold summary.

We usually document the gold summary as a vector labeling the position of its sentences in the article. The i^th sentence in the article is included in the gold summary if the i^th element in the vector is 1. The vector is thus called gold labels, which is directly the target value for a sequential labeling problem, and thus for our extractive summarization. Luckily, the generated gold labels are included in our dataset, which is generated using the algorithm mentioned above. This saves a lot of time for us. We will use these gold labels to train our model as well as generate the gold summaries when evaluating the quality of our generated summaries. A sample of article, summary and gold summary is listed in Table 2.

Article
-lrb- cnn -rrb- a duke student has admitted to hanging a noose made of rope from a tree near a student union , university officials said thursday . the prestigious private school did n't identify the student , citing federal privacy laws . in a news release , it said the student was no longer on campus and will face student conduct review. the student was identified during an investigation by campus police and the office of student affairs and admitted to placing the noose on the tree early wednesday , the university said. officials are still trying to determine if other people were involved . criminal investigations into the incident are ongoing as well . students and faculty members marched wednesday afternoon chanting " we are not afraid . we stand together , " after pictures of the noose were passed around on social media . at a forum held on the steps of duke chapel , close to where the noose was discovered at 2 a.m. , hundreds of people gathered . " you came here for the reason that you want to say with me , ' this is no duke we will accept . this is no duke we want . this is not the duke we 're here to experience .

[3]https://pypi.org/project/rouge/

and this is not the duke we 're here to create , ' " duke president richard brodhead told the crowd. the incident is one of several recent racist events to affect college students . last month a fraternity at the university of oklahoma had its charter removed after a video surfaced showing members using the n-word and referring to lynching in a chant . two students were expelled . in february , a noose was hung around the neck of a statue of a famous civil rights figure at the university of mississippi . a statement issued by duke said there was a previous report of hate speech directed at students on campus . in the news release , the vice president for student affairs called the noose incident a " cowardly act . " " to whomever committed this hateful and stupid act , i just want to say that if your intent was to create fear , it will have the opposite effect , " larry moneta said wednesday . duke university is a private college with about 15,000 students in durham , north carolina . cnn 's dave alsup contributed to this report .
Human-written Summary
walter mondale was released from the mayo clinic on saturday , hospital spokeswoman said . the former vice president , 87 , was treated for cold and flu symptoms .
Generated Summary
-lrb- cnn -rrb- a duke student has admitted to hanging a noose made of rope from a tree near a student union , university officials said thursday . the prestigious private school did n't identify the student , citing federal privacy laws . in a news release , it said the student was no longer on campus and will face student conduct review.
Gold Summary
-lrb- cnn -rrb- a duke student has admitted to hanging a noose made of rope from a tree near a student union , university officials said thursday . the prestigious private school did n't identify the student , citing federal privacy laws . in a news release , it said the student was no longer on campus and will face student conduct review . the student was identified during an investigation by campus police and the office of student affairs and admitted to placing the noose on the tree early wednesday , the university said

Table 2: A sample for Article, Summary, Generated Summary and Gold Summary. The summary is generated by Model 4 with threshold 0.3.

3 Architecture

Our model architecture can be decomposed into four consecutive parts: **Word Embedding, Sentence Encoder, Document Encoder, Document Decoder** (See Figure 1).

A Word Embedding layer first encodes the text into a vector representation according to a (pre-trained or fine-tuned) embedding layer. We then aggregate sentence-level information using a Sentence Encoder. The next step is to encode the information into a document-level representation using a Document Encoder. Finally we get a probability for each sentence which decides whether it should be included in the model summary after a Document Decoder. One should notice that the document-level representation should have a dimension equal to length of document to satisfy the need of Document Decoder. The detail of these blocks will be discussed in the following parts.

3.1 Pre-trained Word Embedding

In tasks where we use word-level embedding, it will be difficult to train the model starting from randomness for the embedding layer. Transferring the pre-trained embedding to some specific application is prevalent in industry to improve training performance. This method also provides a good solution for the task of extractive text summarization. The commonality of these pre-trained embedding models is that they are all trained on large data sets that might not be directly related to specific applications, but then yield knowledge that can be transferred to other applications. Just to list a few, Word2Vec[10] and Glove[13] provide vector representations for some specific vocabularies, while BERT[2] model provides a tokenizer and a network for a given text with huge number of parameters. We will use pre-trained Word2Vec embedding directly in our model instead of going into details in Word2Vec. Since Keras provides a embedding layer which can take Word2Vec Embedding matrix as initial value, and can also be trainable or not, we test the case when the embedding layer is trainable

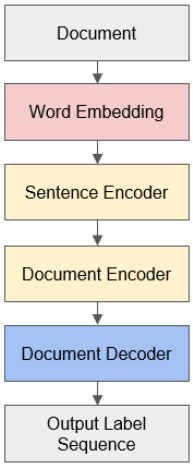

Figure 1: Architecture of our model

or not, which can be viewed as fine-tuning. We choose top 30000 popular words in the dataset as the vocabulary, use the downloaded pre-trained dictionary to implement the embedding from a Python library called gensim[4], and employ a random embedding for those unknown words.

3.2 Sentence Encoder

3.2.1 CNN Sentence Encoder

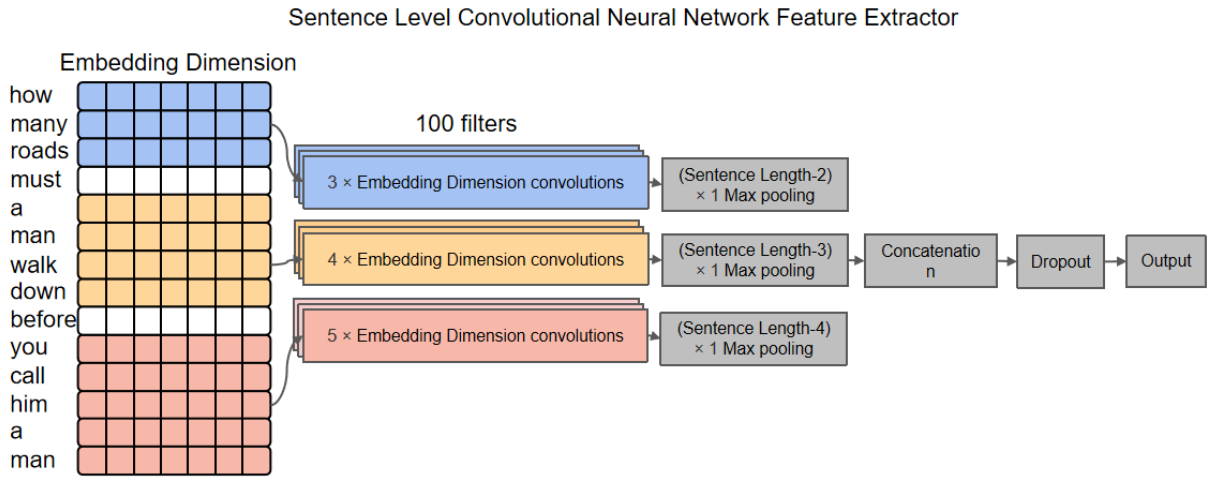

Figure 2: CNN Feature Extractor. The input is embedded sentence vectors, and the output is a vector with 300 dimension to represent the whole sentence.

Every word is converted to a high dimensional vector representation as the output of word embedding. Sentence Encoder aims to get a sentence representation from those word embedding. The first attempt is to use a layer based on Convolutional Neural Network(CNN), proposed by Kim[6] originally for text classification task, to aggregate the information contained in words and form the vector representing the meaning of sentence.

[4]https://radimrehurek.com/gensim/

The structure of this layer is illustrated in Figure 2. We use three CNN layers, all with 100 filters but different filter size to create a "moving average" of the words. Then each layer is linked to a max pooling layer to get the sentence-level representation for each filter. Finally, all values are concatenated together, go through a dropout layer and output is produced.

We are interested in this CNN Sentence Encoder since it is stated to have similar performance as the RNN Sentence Encoder [5][19] . We will evaluate the performance of this sentence encoder in our experiment in contrast to RNN encoder.

3.2.2 RNN Sentence Encoder

Another attempt is to apply a bidirectional RNN-based encoder[11]. Taking word embedding as input, the first layer of the RNN runs at the word level and computes hidden state representations at each word position sequentially. Another layer of RNN runs backwards from the last word to the first and we refer to the pair of such forward and backward RNNs as a bidirectional RNN. Then the final output of the sentence encoder is the average pooling of the concatenated hidden states of the bi-directional sentence-level RNN, as shown in the blue and gray part in Figure 3.

3.3 Document Encoder

3.3.1 RNN-Based Encoder

We firstly apply a bidirectional RNN-based document encoder which has the same structure with RNN-based sentence encoder but runs at the sentence level and takes the sentence embedding (the output of the sentence encoder) as input. Then the representation of the entire document is then modeled as a non-linear transformation of the average-pooled, concatenated hidden states of the bidirectional sentence-level RNN, as shown in the yellow and red part in Figure 3.

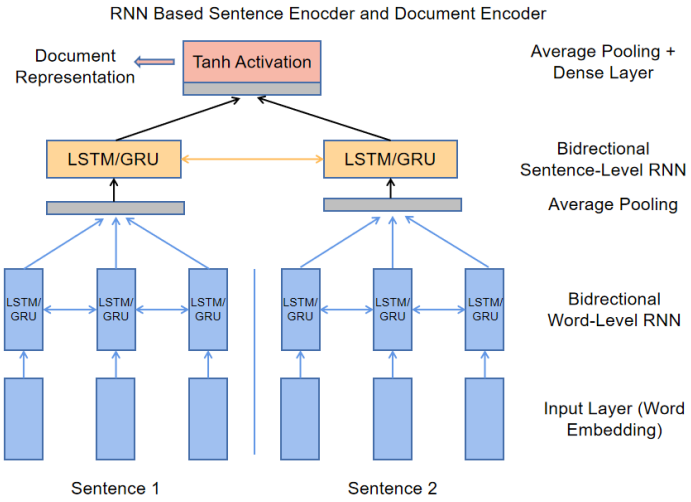

Figure 3: RNN-Based Encoder. The input is the Word2Vec word embedding, and the output is a vector with 2*200(hidden dimension) dimension to represent the whole document.

3.3.2 Transformer Encoder

As an alternative for document decoder, transformer [15] has ruled the NLP world in many application areas by its attention mechanism. We will only make use of the self-attention layer which allows pairwise interaction of the information within the document level. According to [16], we design a similar architecture that deploys self-attention modules as document encoder, with an innovative scoring decoder layer. Our model utilises Bidirectional-RNN and self-attention module to encode the sentence and document respectively. As in Fig 4, X_i represents the word embedding for sentence i, S_i is the sentence representation after word embedding being processed by the sentence encoder and D_i is the document representation after sentences S_i being processed by document encoder. All document level representation is then put into the Information Interactive Decoder where outputs a score for each sentence, which will further be interpreted as a probability denoting the probability of including this sentence in the summary.

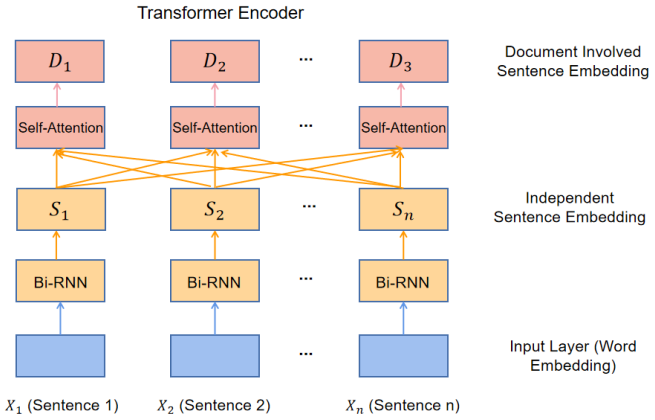

Figure 4: An architecture designed with self-attention layer as document encoder

3.4 Document Decoder

3.4.1 Notation

We use h_j to denote the concatenated hidden states at the j^{th} time step of the bidirectional sentence-level RNN, which can be obtained from RNN-based document encoder. And we use l_j to denote the document involved sentence embedding for j^{th} sentence, which is the output of the transformer we introduced above. Besides, N_d is the number of sentences in the document and N is the number of documents.

3.4.2 Simple Linear Decoder(MLP)

The basic structure of the decoder is that the hidden state representing the current sentence is classified, by going through MLP + sigmoid activation, to whether that sentence belongs to the summary, as shown below.

$$P(y_j = 1|h_j) = \sigma(Wh_j + b), \tag{1}$$

where y_j is a binary variable indicating whether the j^{th} sentence is part of the summary.

3.4.3 Information Interactive Decoder(IID)

We adopt another popular decoder which makes the following improvements: before inputting to the sigmoid classifier, the representation of each sentence is processed to include some statistical information of the sentence. For example, the position information of a sentence in a document is usually the key information of whether a sentence is selected as a summary, and important sentences often appear in the position of the first few sentences or the last few sentences in the document. As shown below,

$$\begin{aligned} P(y_j = 1|v_j, s_j, d) = \sigma(&W_c v_j && (content) \\ &+ v_j^T W_s d && (salience) \\ &- v_j^T W_r tanh(s_j) && (novelty) \\ &+ W_{ap} p_j^a && (abs.\ pos.\ imp.) \\ &+ W_{rp} p_j^r && (rel.\ pos.\ imp.) \\ &+ b), && (bias\ term) \end{aligned} \tag{2}$$

where y_j is a binary variable indicating whether the j^{th} sentence is part of the summary, v_j , the representation of the sentence is given by a non-linear transformation of h_j or l_j(i.e. $v_j = relu(Wh_j + b)$) for RNN-based document encoder or transformer respectively. d is the representation of the entire document and is given by:

$$\begin{aligned} d &= tanh(W_d \frac{1}{N_d} \sum_{j=1}^{N_d} h_j + b), && (RNN - based\ document\ encoder) \\ &= tanh(W_d \frac{1}{N_d} \sum_{j=1}^{N_d} l_j + b), && (Transformer) \end{aligned} \tag{3}$$

and s_j is the dynamic representation of the summary at the j^{th} sentence position, given by:

$$s_j = \sum_{i=1}^{j-1} h_i P(y_i = 1|v_i, s_i, d). \tag{4}$$

In other words, the summary representation is simply a running weighted summation of all the sentences representation till sentence j, where the weights are given by their respective probabilities of summary membership.

In Eqn.(2), the term $W_c h_j$ represents the information content of the j^{th} sentence, $h_j^T W_s d$ denotes the salience of the sentence with respect to the document, $h_j^T W_r tanh(s_j)$ captures the redundancy of the sentence with respect to the current state of the summary, while the next two terms model the notion of the importance of the absolute and relative position of the sentence with respect to the document[1]. We consider p_a and p_r , the absolute and relative positional embeddings respectively, as model parameters as well.

Accordingly,we minimize the negative log-likelihood of the observed labels at training time.

$$\begin{aligned} l(W, b) = -\sum_{d=1}^{N}\sum_{j=1}^{N_d}(y_j^d log P(y_j^d = 1|v_j^d, s_j^d, d_d) \\ + (1 - y_j^d) log(1 - P(y_j^d = 1|v_j^d, s_j^d, d_d))) \end{aligned} \tag{5}$$

4 Experimental Setup

4.1 Implementation Details

Due to the size of the dataset and the intrinsic complexity of the model architecture required for the task, training models consumes a huge amount of computation. Since neither our own laptops nor Colab platform could provide us the needed computation, we rent a **RTX 3080 GPU** to finish our task[5]. Even though, it still takes around 12 hours to finish training one model, we have also searched into some more advanced settings such as Fine-tune BERT [8], Pre-training Tranformers[18], but it takes longer time so we did not obtain the results from them.

During training process, we will utilise binary cross entropy loss and Adam optimizer with $lr = 1e^{-4}$. Batch size equals to 32 and checkpoints are saved and evaluated on the validation set at every epoch, with 8975 steps at each epoch. We will train 20 epochs for each model. Here the gold labels (in 2.3) will be used and the trained models will tested to produce Rouge metrics.

	Word Embedding	Sentence Encoder	Document Encoder	Document Decoder
Model 1	Pre-train	CNN	BiLSTM	MLP
Model 2	Pre-train	BiGRU	BiLSTM	MLP
Model 3	Pre-train	CNN	BiLSTM	IID
Model 4	Pre-train	BiGRU	BiLSTM	IID
Model 5	Pre-train	BiGRU	Transformer	IID
Model 6	Fine-tune	BiGRU	Transformer	IID

Table 3: Models architecture

Baseline

We will use a simple Lead-3 model, which simply produces the leading three sentences of the document as the summary as a baseline.

More advanced models

We want to explore which combination of different blocks in different parts gets the best performance. Due to limited computational resources, we were not able to try all combinations. For this reason, we adopted a control variables approach, starting with a basic model and gradually replacing blocks with more complex and advanced ones to see if these new blocks led to performance improvements. We evaluate 6 models according to this principle, which are listed in Table 33. For these models, a pre-trained embedding layer with dimension 128 will be applied, and various encoder&decoder pairs will be deployed. For CNN sentence encoder, we follow the setting in the original paper[6], with the number of filters for each kernel equal to 100, l2 penalty parameter

[1]The absolute position denotes the actual sentence number, whereas the relative position refers to a quantized representation that divides each document into a fixed number of segments and computes the segment ID of a given sentence.

[5]`https://featurize.cn/vm/mine`, which is a famous online lab(in Chinese)

equal to 3, and dropout rate equal to 0.5. The hidden dimension for BiGRU sentence encoder is 128. The hidden dimension for BiLSTM document encoder is 512. In the Information Interactive layer, the positional embedding dimension is 64, the segmentation number for relative positional embedding is 10, and the hidden units in this layer is 100. Additionally, for model 6 the embedding layer will also be fine-tuned during training.

Training Loss and Validation Loss

We will show the training and validation loss during training process of all 6 models, although the binary cross-entropy loss may not provide insightful information for text summarisation problems, but it somewhat reflects the goodness of our models performing on classification.

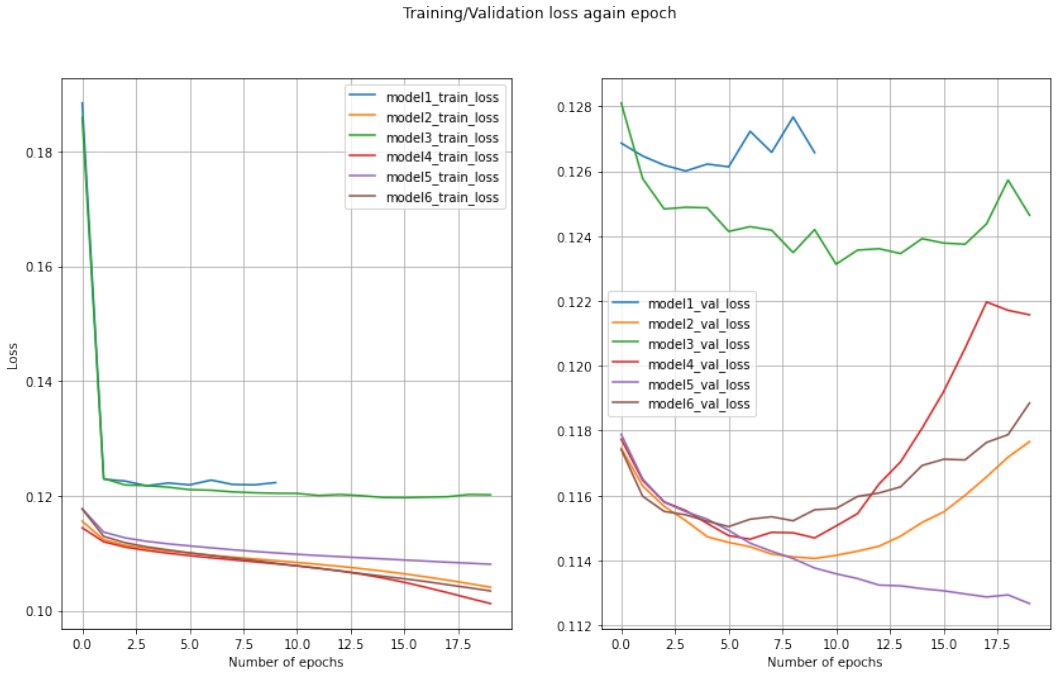

Figure 5: Train/Validation loss

For model 1, it is the first model that we developed and was treated as an experiment, we forgot to train again for 20 epochs, and we did not expect it to outperform others. From the figures we can find that Model 5 has the lowest validation loss and is the only one that did not appear over-fitting trend over the 20 epochs.

4.2 Evaluation

4.2.1 ROC curve

With the settings demonstrated in (2.2), we could find that the ROC curves are smooth for all models, and all of them have good AUC values. From the perspective of ROC metric, model 1 and model 3 have the lowest figures. And these two models both use CNN as the sentence encoder, also we can find that the two models have the highest validation loss during training. The model with the best performance in terms of AUC score is model 5, which is equipped with Word2Vec+BiGRU+BiLSTM+Transformer+IID. This model also has the lowest validation loss. However, to assess a model in text summarization problems, the loss and ROC curve may not tell the true story, and we will incorporate the evaluation on ROUGE metric in the following parts.

4.2.2 ROUGE

Out of all three metrics of each ROUGE-N, we only provide the results from F1-score, because we are using full-length variants and F1-score gives an overall assessment of the model[11]. However, since the core of the task is classification, the threshold of classifying positive label matters when doing prediction. At test time, selecting all sentences with $P(y = 1) > 0.5$ seems not to be a good boundary because our training dataset is very imbalanced in terms of summary-membership of sentences. As calculated in the notebook, the number of negative labels is more than 8 times than that of positive labels, with $\frac{\#Negative\ labels}{\#Postive\ labels} = \frac{6860319}{854206} = 8.03$. Hence we did not expect good performances with it. Instead of keeping threshold $\tau = 0.5$, we will try three values of threshold: $\tau = \{0.2, 0.3, 0.4\}$. The results are given in Table 4, Table 5 and Table 6.

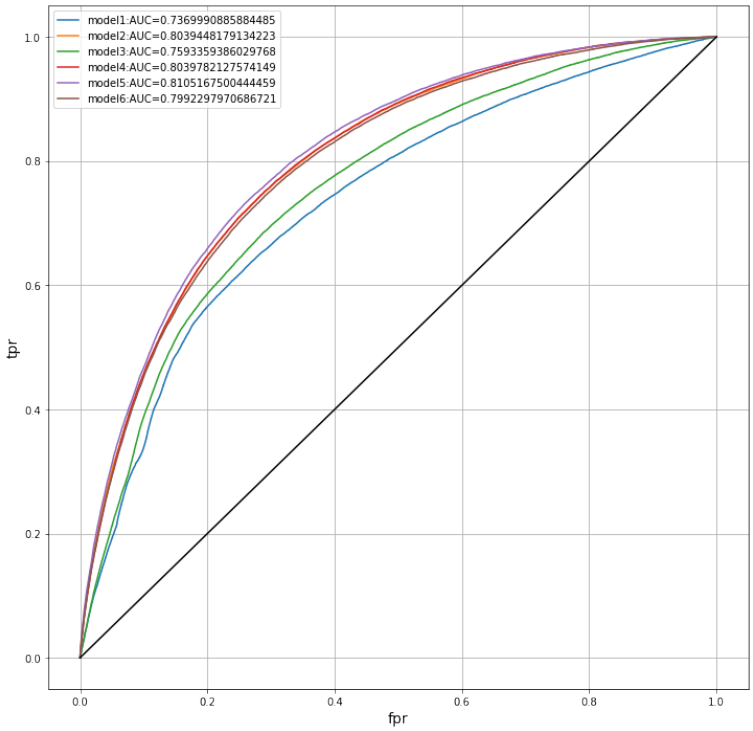

Figure 6: ROC curves of 6 models

	ROUGE-1	ROUGE-2	ROUGE-L
Lead-3	40.71	17.05	37.07
Model 1	40.32	14.62	37.18
Model 2	42.03	18.58	39.11
Model 3	40.83	17.59	37.83
Model 4	42.32	**18.71**	39.24
Model 5	**42.50**	18.65	**39.47**
Model 6	41.65	18.32	38.77

Table 4: Performance comparison of 6 models on the entire CNN Daily Mail test set using full-length F1 variants of Rouge. Threshold is set at $\tau = 0.2$. Model 4 and Model 5 are performing great.

	ROUGE-1	ROUGE-2	ROUGE-L
Lead-3	40.71	17.05	37.07
Model 1	30.89	12.56	27.66
Model 2	39.90	17.66	36.47
Model 3	34.60	14.27	31.23
Model 4	39.84	17.51	36.32
Model 5	40.50	17.93	37.01
Model 6	**40.74**	**18.03**	**37.37**

Table 5: Performance comparison of 6 models on the entire CNN Daily Mail test set using full-length F1 variants of Rouge. Threshold is set at $\tau = 0.3$. Model 6 has the best performance over all three metrics

	ROUGE-1	ROUGE-2	ROUGE-L
Lead-3	**40.71**	**17.05**	**37.07**
Model 1	24.89	10.32	22.14
Model 2	29.75	13.22	26.90
Model 3	25.12	10.86	22.79
Model 4	30.20	13.24	27.22
Model 5	31.48	14.02	28.76
Model 6	31.94	14.20	29.09

Table 6: Performance comparison of 6 models on the entire CNN Daily Mail test set using full-length F1 variants of Rouge. Threshold is set at $\tau = 0.4$. Few sentences are selected in generated summary for the high threshold so the ROUGE scores are low, and Lead-3 bench mark significantly beat these models. This maybe be caused by the imbalance of training set

	ROUGE-1	ROUGE-2	ROUGE-L
SummaRuNNer(2017)[11]	39.60	16.20	35.30
SWAP-NET(2018)[4]	41.6	18.3	37.7
Model 5 with threshold 0.2	42.50	18.65	39.47

Table 7: Comparison of our best model with previous SOTA-level models on CNN/DailyMail. We beat the old baseline for extractive text summarization on this dataset

4.2.3 Comments

We conclude our observations as following:
1. Only when we use threshold equal to 0.2 or 0.3, our model can be as competitive as Lead-3. Most models outperform Lead-3 when the threshold is 0.2.
2. In terms of Sentence Encoders, the performance of CNN is significantly worse than BiGRU, which contradicts to the previous findings[5].
3. We find that in terms of Document Decoder, the performance of IID Layer is slightly stronger than that of MLP Layer, which proves that the additional features is truly effective in this task.
4. Models using Transformer as Document Encoder are performing quite well. The fine-tuned model performs better when the threshold is higher, and the pre-trained model performs better when the threshold is lower.
We also select our best model and make comparison with previous SOTA models, see 7. The result indicates that our model is very competitive on this dataset.

5 Conclusion

In this report, we construct a number of extractive summarization models and conduct experiments under a certain architecture, and seek to better understand how different combinations blocks would affect the performance. We improve our model by gradually replacing blocks, and Model 5 and Model 6 are shown to be competitive against the baseline and previous SOTA models.
We were not able to use BERT in our model due to limitation on computational resources. In the future, we will try to use more advanced transferable learning methods such as applying BERT, fine-tuned pre-trained model and using unsupervised learning schemas in order to achieve better results.

Acknowledgment
Statement: The contributions of our group members to this project are equal.
Specifically, 25882(ID number) is responsible for the whole preparation of the data, the code of the ROUGE metrics, the word embedding, the baseline model and the CNN-based sentence encoder. He also takes charge of the construction of the basic content of the python notebook and writing down his work correspondingly in the final report.
26725(ID number) is responsible for the code of the RNN-based sentence and document encoder and the document decoder. He also writes down his work correspondingly and plus the abstract and the introduction part in the final report.
39290(ID number) is responsible for the code of the transformer encoder. He also takes charge of the training of all 6 models and outputs all the prediction results including ROUGE metrics and ROC curves. He also writes

down his work correspondingly in the final report.

References

[1] Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany, August 2016. Association for Computational Linguistics.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[3] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend, 2015.

[4] Aishwarya Jadhav and Vaibhav Rajan. Extractive summarization with SWAP-NET: Sentences and words from alternating pointer networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 142–151, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[5] Chris Kedzie, Kathleen McKeown, and Hal Daume. Content selection in deep learning models of summarization, 2018.

[6] Yoon Kim. Convolutional neural networks for sentence classification, 2014.

[7] 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.

[8] Yang Liu. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*, 2019.

[9] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders, 2019.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[11] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents, 2016.

[12] Shashi Narayan, Nikos Papasarantopoulos, Shay B Cohen, and Mirella Lapata. Neural extractive summarization with side information. *arXiv preprint arXiv:1704.04530*, 2017.

[13] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[14] Abhishek Singh, Manish Gupta, and Vasudeva Varma. Unity in diversity: Learning distributed heterogeneous sentence representation for extractive summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[16] Hong Wang, Xin Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. Self-supervised learning for contextualized extractive summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2221–2227, Florence, Italy, July 2019. Association for Computational Linguistics.

[17] Wenpeng Yin and Yulong Pei. Optimizing sentence modeling and selection for document summarization. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[18] Xingxing Zhang, Furu Wei, and Ming Zhou. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy, July 2019. Association for Computational Linguistics.

[19] Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Searching for effective neural extractive summarization: What works and what's next, 2019.

[20] Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. Neural document summarization by jointly learning to score and select sentences. *arXiv preprint arXiv:1807.02305*, 2018.

A GitHub link for the code

Here we provide the link for the Github repository that contains the code for the report. `https://github.com/lse-st456/project-2022-group-16`