applied sciences

Article

A Text Abstraction Summary Model Based on BERT Word Embedding and Reinforcement Learning

Qicai Wang [1], Peiyu Liu [1,*], Zhenfang Zhu [2,*], Hongxia Yin [1], Qiuyue Zhang [1] and Lindong Zhang [1]

[1] School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China; 2018020907@stu.sdnu.edu.cn (Q.W.); 2018020908@stu.sdnu.edu.cn (H.Y.); 2018309063@stu.sdnu.edu.cn (Q.Z.); 2018020898@stu.sdnu.edu.cn (L.Z.)
[2] School of Information Science and Electrical Engineering, Shandong Jiaotong University, Jinan 250357, China
* Correspondence: liupy@sdnu.com.cn (P.L.); zhuzf@sdjtu.edu.cn (Z.Z.); Tel.: +86-131-8889-9297 (P.L.); +86-137-9310-0702 (Z.Z.)

Received: 9 October 2019; Accepted: 29 October 2019; Published: 4 November 2019

Abstract: As a core task of natural language processing and information retrieval, automatic text summarization is widely applied in many fields. There are two existing methods for text summarization task at present: abstractive and extractive. On this basis we propose a novel hybrid model of extractive-abstractive to combine BERT (Bidirectional Encoder Representations from Transformers) word embedding with reinforcement learning. Firstly, we convert the human-written abstractive summaries to the ground truth labels. Secondly, we use BERT word embedding as text representation and pre-train two sub-models respectively. Finally, the extraction network and the abstraction network are bridged by reinforcement learning. To verify the performance of the model, we compare it with the current popular automatic text summary model on the CNN/Daily Mail dataset, and use the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics as the evaluation method. Extensive experimental results show that the accuracy of the model is improved obviously.

Keywords: BERT word embedding; text summary; reinforce learning

1. Introduction

Text summarization is a task of compressing long text into short one meanwhile keeping up the central idea. Automatic text summarization is one of the core tasks in natural language processing (NLP) and information retrieval. As information techniques develop and change rapidly, especially for Mobile Internet, there is a huge amount of data will inevitably be produced day to day. Nowadays, the traditional manual summarization method is difficult to suit the needs of information retrieval in people's daily life. Therefore, automatic summarization is becoming more and more important in the wave of mass information.

The text summary method can be classified into two paradigms: extractive and abstractive. The extractive method extracts the important sentences or a section of text from the original text and combines them to form a summary [1–4], while the abstractive method will generate novel words which do not exist in the source text and while retaining the original meaning [5–8]. Compared with the difference between them, the extraction paradigm is relatively simple and ensures grammatical correctness, but the semantics are inconsistent; while the abstraction paradigm is more concise, but redundant. When summarizing a very long text, the extractive approach is too simple and the readability is poor, and the abstract method of compressing a long input sequence with a single fixed-length vector may cause the information loss, neither of them could perform the long text summary better. At present, some neural network models [9,10] combine the advantages of extractive and abstractive approaches. Firstly, selecting key sentences from the source text by the extractive method, then generating a summary of these sentences by the abstractive method. Reference [11] proposed a new model for the long text summary, which abstracts the summary by using a deep communication agent, first of all, dividing the long input text into multiple agents encoders, and then generating the summary through a unified decoder. Although these methods have achieved good results, due to the limitation of specific data sets and the small amount of data, their word embedding effect is not obvious and the semantic features of a text cannot be fully obtained.

In view of the above problems, we utilize the advantages of the pre-trained language model, BERT (Bidirectional Encoder Representations from Transformers) [12], which is successfully applied in many NLP tasks, and we use the pre-trained representation of BERT as the text representation of all our models. BERT has been pre-trained on a large amount of unlabeled corpus to generate better word embedding. Inspired by [9], we proposed a new method that integrates the extractive network and abstractive network by using reinforcement learning. CNN/Daily Mail was used as the experimental dataset in this paper. Firstly, we convert the human written abstractive summaries to the ground truth labels. Secondly, we pre-train two sub-models respectively, the extractive model is trained according to the generated pseudo-document summary data pairs, namely, the article and the ground truth labels are paired; the abstractive model is trained on the basis of the ground truth labels and the abstractive summaries labels. Finally, in order to train a complete end-to-end model, we use the strategy gradient of reinforcement learning to bridge two well-trained networks.

In addition, in order to obtain better sentence and document vectors in the extractive sub-model, we use the hierarchical self-attention mechanism. As we all know, each word contributes differently to sentence semantics, and so does each sentence to document semantics.

Empirically, on CNN/Daily Mail dataset, we used the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics to evaluate the performance of our method. Our approach was the new state-of-the-art. On the DUC2002 dataset, we only used it as a test dataset due to the small scale. The experimental results show that our model has better generalization ability.

Our major contributions can be summarized as follows:

(1) In this paper, we applied BERT word embedding to the text summarization task, and improved the performance of the task by taking advantage of the rich semantic features of BERT word embedding.
(2) We train a single unified model, which combines the advantages of universal language model BERT, extraction method, and abstraction method.
(3) Our approach is based on the strategy gradient of reinforcement learning and bridges the abstractive model and the extractive model. We have carried experiments on a large dataset to find that our approach achieves state-of-the-art ROUGE scores.

The rest of our paper is organized as follows. In Section 2, the related work is described. In Section 3, the proposed approach is presented in detail. Section 4 describes the related content of the experiments. We show the result and analysis of the result in Section 5. Section 6 presents conclusions and future work.

2. Related Works

Text summarization has been widely studied in recent years. We first introduce the related works of extractive and abstractive summarization and then introduce the related works of reinforcement learning and self-attention. Finally, we introduce a few related works with BERT (Bidirectional Encoder Representations from Transformers) and word embedding.

The previous works [1–4] mainly focused on extractive methods. References [1–3] select sentences using RNN (Recurrent Neural Network) to get the vector representations of sentences and articles. Reference [4] uses RNN and graph convolutional networks to compute the importance of sentences. Although the extractive method can easily achieve a relatively high score, the consistency is usually poor. The sequence-to-sequence (seq2seq) model [13,14] has been successfully applied in various NLP tasks such as NMT (Neural Machine Translate), QA (Question Answering), and Image Captioning. The sequence model can freely read or generate text content that makes abstraction practical. Reference [5] is the first to apply the attention mechanism model based on seq2seq to abstractive text summarization. Compared with traditional methods, this method shows an obvious performance improvement. The network [6] copies words from the source article by pointing or generate new words from the vocabulary. References [7,15] also integrate pointer network [16] into their model to handle the problem of OOV(Out-Of-Vocabulary) words. Other new methods (e.g., [8]) based on the seq2seq framework are proposed, and all of them have achieved effectively result. Although abstractive models are relatively concise by generating new words, due to the input text being too large, they cause the problems of loss of information and high computational cost. The models [9,10] combined the advantages of the extractive method and abstractive method and proposed a hybrid extractive-abstractive architecture. The extractive network is used to extract the sentences with obvious semantics from the input sequence, and then the abstractive network summarizes the selected sentences and generates the final text summary.

There is a common problem of exposure bias in these models which are built on seq2seq framework, namely, the reference abstract and the words generated by the previous time step are used as the input of decoder in the training stage, and only the words generated by the previous time step are used in the test stage; and the cross-entropy loss function was used in the training, but the metrics were used in the test such as ROUGE, BLEU, etc. Previous works [3,7–9,11] used reinforcement learning [17] to mitigate these existing problems. Reference [3] used reinforcement learning for ranking sentences in pure extraction-based summarization. Reference [7] used reinforcement learning policy gradient methods for abstractive summarization. Reference [8] used actor-critic policy methods for abstractive summarization. Reference [11] combined a few independent and cooperative agents to form an end-to-end training model by reinforcement learning. However, none of these methods used reinforcement learning to bridge the non-differentiable computation of two neural networks. Following the previous work [9], we also used reinforcement learning in the model to bridge the pre-trained extractive network and abstractive network.

The attention mechanism has been successfully used in a variety of NLP tasks such as machine translation [18] and text summarization [5,6]. We all know that each element in a sequence contributes differently to the sequence. So self-attention is widely used in language modeling [19], sentiment analysis [20], and other tasks. The hierarchical self-attention is also used to encode sentences and documents for the extractive model in [21]. Inspired by [21], we use self-attention for document representation.

Word embedding representations of the aforementioned model usually exist in two ways: direct learning and pre-trained. Direct learning is the way that gets the word representation in the process of model training, while pre-trained is the way that gets the word embedding initialization by training word2vec on the dataset. For example, direct learning is used in the models [6,9], while models [10,22] used word2vec as word embedding in their models. Although the aforementioned work makes a beneficial exploration in the direction of model structure combination, it does not consider the role of the pre-training universal language model, but applies it to the text summary model. The pre-trained model was trained from large amounts of unlabeled text and has got more complete word representation. The embedded vectors from pre-trained models are richer and more precise, whether in spatial dimensions or semantic features.

BERT is the latest representation of the pre-trained language model, which has recently succeeded in many NLP tasks. BERT is pre-trained for large scale text data, combining word representations and sentence representations in a large transformer [23]. The strategies of applying pre-trained BERT are mainly divided into feature-based and fine-tuning methods. Our method does not follow the tasks of the literature [12], but uses a feature-based strategy, because BERT can generate better contextualized token embeddings, thus our model based on top of them can get better performance.

From the above analysis, the word vectors of the pre-trained models (BERT) are used as task input respectively in this paper, reinforcement learning is used to integrate the extraction network and generation network into a unified model. From a human-written manual summary, first of all, we need to fully understand the main meaning of the article, then select key sentences according to the context information of the article, and then rewrite the selected sentences. The model in this paper adopts the same idea, and the corresponding relationship between them is shown in Figure 1.

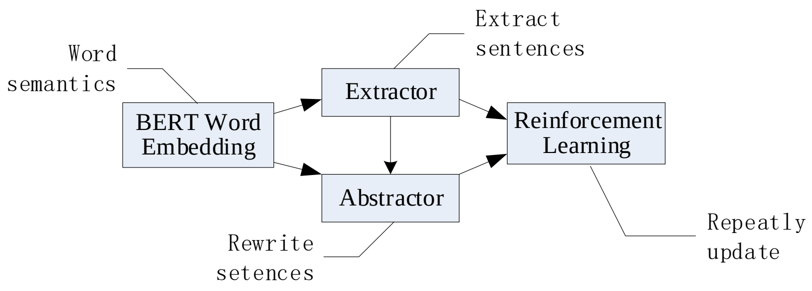

Figure 1. Automatic summary vs. manual summary.

3. Methods

3.1. Problems

The text summarization task can be seen as: in the case of fully understanding the information of the input text, we select the important sentences from the input sequence, and then these sentences are rewritten to the shorter version that do not change the main meaning. Our whole model is consists of two sub-models: the extraction agent and abstraction agent. Formally, the input article is regarded as a sequence of sentences. $s = [s_1, s_2 \cdots s_m]$, m is the index of input sentence sequence, each sentence is a sequence of words. $s_i = [w_1, w_2, \cdots, w_n]$, n is the index of the word sequence. We select the important sentences from the sequence s to make a new sequence: $s' = \left[s'_1, s'_2, \cdots, s'_k\right]$, $K < M$, and then generate the summary s″ by rewriting the sequence s′. In the case of mentioned above, namely, by fitting the training data, firstly find the extraction function: $f_1 : s' = f_1(s)$, then find the abstraction function: $f_1 : s' = f_1(s)$, so the final objective function that will be obtained $f : s'' = f_2(f_1(s))$. The overall flowchart of this model can be seen in Figure 2.

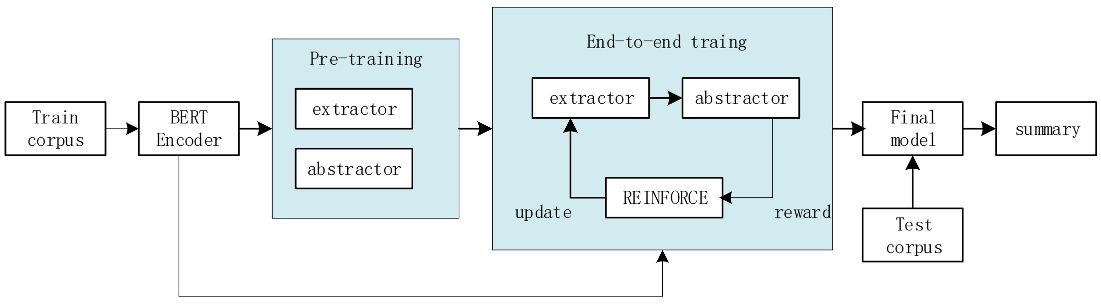

Figure 2. Automatic summary vs. manual summary.

In our works, we used BERT as our encoder for word tokens and sentences. Our main processes include: firstly, we pre-trained our sub-models: abstractor and extractor; secondly, we trained the full end-to-end model with REINFORCE LEARNING, which can bridge the sub-models. The three training processes are mapped to the fitting processes of the aforementioned functions, respectively.

3.2. Word Embedding

Word embedding is based on the distributed hypothesis of word representation. Word embedding represents natural language words as low-dimensional vector representations that computers could understand. The semantic relevance of words can be measured by the similarity between vectors. Word embedding now commonly used in NLP tasks include Word2Vec, Glove, BERT, etc.

There are two existing strategies to apply pre-trained language representations to downstream tasks: feature-based and fine-tuning [12]. Although BERT is mainly used in a fine-tuning mode in most NLP tasks, we use it as a feature-based mode and only use it as our encoder for text representation. As the same as BERT, the WordPiece tokenizer is used for input text sequence. Experiments show that the WordPiece [24] tokenizer is more effective than the natural tokenizer (here, 'natural tokenizer' refers to the method of word segmentation based on space, comma and other punctuation, and CoreNLP toolkits (https://stanfordnlp.github.io/CoreNLP/) are generally used in the experiment.). BERT can express tokenized words as corresponding word embeddings, as well as the sentences in the article are input into the BERT model, and the sentence vector representation of each sentence is obtained. The above process is expressed as Formula (1)

$$r_m = BERT(s_m) \quad s_m \in S, m \in [0, M] \tag{1}$$

where M is the number of sentences, m is the index of a sentence, s_m denotes the text of the mth sentence, S is the set of sentences, r_m is the sentence vector. Next, word embeddings or sentence vectors are used as input in both extractor and abstractor.

3.3. Extraction Model

Our extraction model is motivated by [21,22]. The main difference is that our extractor uses BERT as a sentence encoder and the document encoder adopts the self-attention mechanism. We take a similar computation method and make some changes by adding a unidirectional GRU. Each sentence of the document is visited sequentially to obtain a shortlist of remarkable sentences with high recall to further facilitate the abstractor is our objectives.

The model consists of three components: a sentence encoder, a document encoder, and a sentence extractor. The sentence encoder adopts BERT as the encoder, a bidirectional GRU with self-attention is used to encode document, a unidirectional GRU is used to compute the summary representation. Then, the representation, the document vector and the hidden state of bi-GRU are involved in the computation of the sentence score. The architecture of the model is shown in Figure 3.

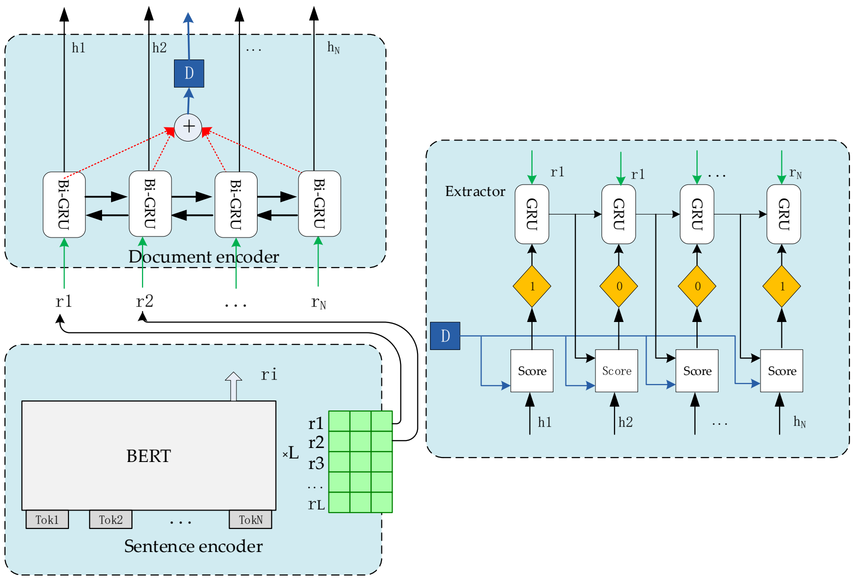

Figure 3. Extraction network model illustration.

After getting the vector representation of sentences by BERT encoder, we can summarize information of documents from both directions. It includes the forward GRU $\overrightarrow{h_t}$ and a backward GRU $\overleftarrow{h_t}$

$$\overrightarrow{h_t} = \overrightarrow{GRU}\left(r_t, \overrightarrow{h_{t-1}}\right) \tag{2}$$

$$\overleftarrow{h_t} = \overleftarrow{GRU}\left(r_t, \overleftarrow{h_{t+1}}\right) \tag{3}$$

where r_t is the sentence vector of tth sentence in the time step t.

Both GRU and LSTM are based on RNN (Recurrent Neural Network), there is no evidence to show which one is the best [25,26]. However, GRU is simpler, more efficient, fewer parameters and easier to implement. Therefore, we use a bidirectional GRU to encode the sentences in the documents and a unidirectional GRU to obtain the summary representation which taking account of decisions made previously. A GRU is a recurrent network with two gates, u_g called the update gate and r_g the reset gate, it can be described by the following equations

$$u_{gj} = \sigma\left(W_{ux}x_j + W_{uh}h_{j-1} + b_u\right) \tag{4}$$

$$r_{gj} = \sigma\left(W_{rx}x_j + W_{rh}h_{j-1} + b_r\right) \tag{5}$$

$$h'_j = tanh\left(W_{hx}x_j + W_{hh}\left(r_{gj} \odot h_{j-1}\right) + b_h\right) \tag{6}$$

$$h_j = \left(1 - u_{gj}\right) \odot h'_j + u_j \odot h_{j-1} \tag{7}$$

where the W's and b's are learnable parameters and h_j is the real valued hidden vector at time step j and x_j is the corresponding input vector namely aforementioned r_i and $\odot$ denotes the Hadamard product.

We concatenated the forward and the backward GRU hidden states to get the vector h_t, which summarizes the information of the sentence s_t and its context, as in Equation (8),

$$h_t = \left[\overrightarrow{h_t}, \overleftarrow{h_t}\right] \tag{8}$$

where d_h denotes the size of hidden vector, M is the number of sentences in the document, so the $h_D \in R^{M*2d_h}$ denotes the whole GRU hidden states, as in Equation (9),

$$H_D = (h_1, h_2, \ldots, h_M) \tag{9}$$

We all know that we pay more or less attention to each sentence according to its contribution to the article. The representation of the whole document is modeled as a weighted sum of the concatenated hidden states of the bidirectional GRU by a self-attention mechanism [26]. We take the concatenated hidden states H_D as input and yield a vector of weights, a_D, as output, calculated as shown in Equation (10),

$$a_D = softmax\left(W_{s_2} \tanh\left(W_{s_1}, H_D^T\right)\right) \tag{10}$$

where W_{s_2} and W_{s_1} are learnable parameters, $W_{s_2} \in R^{k*2d_h}$, $W_{s_2} \in R^k$, k is a hyper-parameter can be set arbitrarily. The softmax() is the normalized function used to normalize the attention weights, which sum up to 1. After getting the attention vector a_D, the document vector is obtained as a weighted sum of the GRU hidden states weighted by a_D, as shown in Figure 4, and Equation (11),

$$d = a_D H_D \tag{11}$$

where $a_D \in R^{1*M}$, d $\in R^{1*2d_h}$, so the document representation is a vector whose dimension is d.

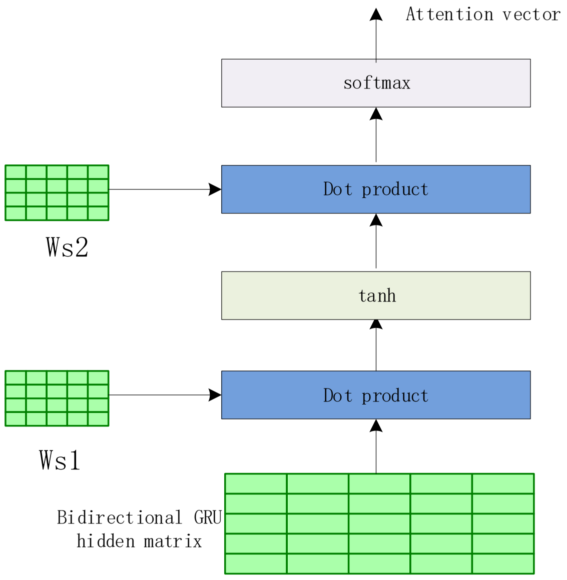

Figure 4. The self-attention unit.

For extractor, each sentence is viewed sequentially again, where a logistic layer makes a binary decision as to whether that sentence belongs to the summary, as shown in Equation (12)

$$\mathrm{P}\left(y_j = 1 \middle| h_j, O_j, d\right) = \sigma\left(W_c h_j + h_j^T W_{sal} d - h_j^T W_{red} \tanh\left(O_{j-1}\right) + W_p p_j + b\right) \tag{12}$$

where y_j is a binary variable indicating whether the jth sentence is included in the summary, h_j is the hidden state of bi-GRU at the jth time step, O_{j-1} is the dynamic representation of the summary before the jth time step, d is the document vector, W_c, W_{sal}, W_{red}, W_p, and b are all learnable parameters. The expression $W_c h_j$ denotes the information content of the jth sentence, $h_j^T W_{sal} d$ represents the salience of the sentence with respect to the article, $h_j^T W_{red} \tanh\left(O_{j-1}\right)$ obtains the redundancy of the sentence with respect to the current representation of the summary, $W_p p_j$ is the position of the sentence with respect to the article. O_{j-1} is calculated using Equation (13)

$$O_{j-1} = GRU\left(Sel_{j-1} r_{j-1}, O_{j-2}\right) \;\; j \geq 2 \tag{13}$$

where $Sel_{j-1\in} \in \{0, 1\}$, O_0 a zero vector.

We do not follow the loss function of the literature [21,22], where they used the negative log-likelihood. We use cross entropy loss as the loss function, as shown in Equation (14)

$$L_{ext} = -\frac{1}{M} \sum_{j=1}^{M} \left(g_j \log \beta_j + \left(1 - g_j\right) \log\left(1 - \beta_j\right)\right) \tag{14}$$

where $g_j \in \{0, 1\}$ is the ground-truth label for the sentence and M is the number of sentences. When $g_j = 1$, it suggests that the jth sentence should be attended to help abstractive summarization. β_j is the normalize attention weights using softmax(), as shown in Equation (15)

$$\beta_j = \frac{\exp\left(P\left(y_j = 1 \middle| h_j, O_j, d\right)\right)}{\sum_{j=1}^{M} \exp\left(P\left(y_j = 1 \middle| h_j, O_{j-1}, d\right)\right)} \tag{15}$$

In the end-to-end training phrase, β_j as the sentence-level attention will be focused on abstract summaries.

Essentially, our extraction model is a binary classifier, which classifies whether the sentences in the input text sequence are important or not.

3.4. Abstraction Model

Another part of our method is an abstraction model that rewrites the previously selected key sentences and then generates a concise and readable summary. We use the pointer-generator network proposed by [6]. The pointer-generator network facilitates copying words from the source text via pointing [16], which improves accuracy and processing ability of OOV words, while retaining the ability to generate new words [6]. The network contains an encoder and a decoder and can be seen as a balance between extractive and abstractive methods. Many similar studies [6,7,11] show that such a model can effectively improve the performance of text summary. More details of the network can be found in the literature [6].

Although we used the pointer-generator network, we made some changes to improve the performance and accuracy of the model. Compared with the vanilla network [6], there are some differences: first, inspired by [10], the new network introduces the updated word attention combined with sentence-level and word-level attentions as same as [10]; second, we replace the LSTM in the network with the GRU, since the GRU is simpler and requires fewer parameters; third, the two models input different amounts of data, when the article reaches 400 tokens the input of the vanilla network is truncated, which causes loss of information, the input of the new network is the key sentences from aforementioned extraction model; fourth, the word embedding of two models are different, the word2vec is used for the vanilla pointer-generator network, the BERT is used in our abstractive network. In addition, the WordPiece tokenizer can help to process the OOV words. The architecture of the updated pointer-generator network is shown in Figure 5.

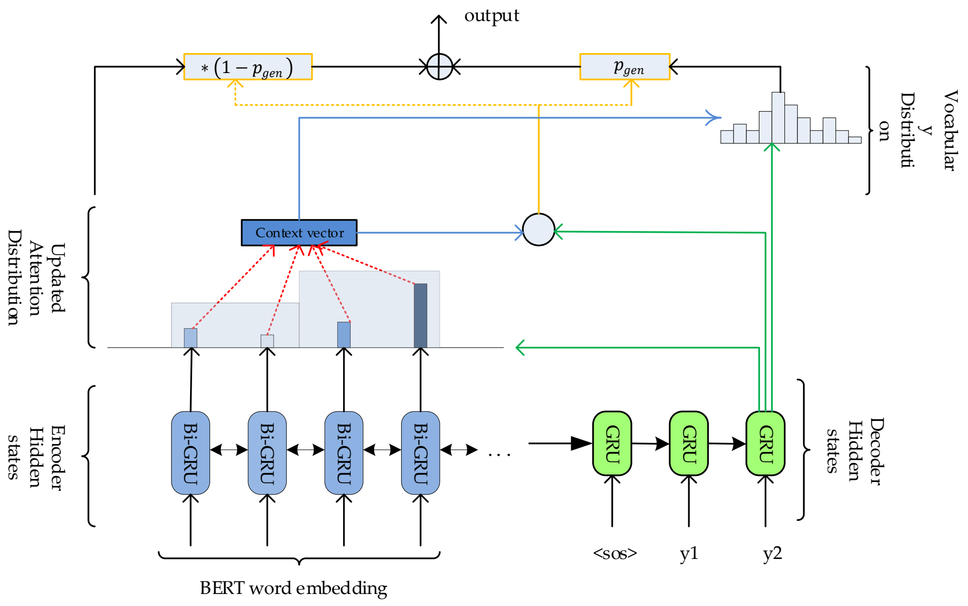

Figure 5. Model of pointer-generator network [10].

There is a lot of evidence that attention mechanism is very important for NLP tasks (e.g., [5,19,23]). We use the sentence-level modulate the word-level attention such that words in less attended sentences are less likely to be generated [10]. Take the simple scalar multiplication of the aforementioned sentence attention β_m in sec 3.3 and the word attention α_n^t of the mth sentence, and then renormalize the result into the new attention. The updated word attention μ_n^t,

$$\mu_n^t = \frac{\alpha_n^t \times \beta_m}{\sum_n \alpha_n^t \times \beta_m}. \tag{16}$$

The final probability distribution of word w is related to the updated word attention μ^t as follows

$$P_{final}(w) = p_{gen}(h^*) P_{vocab}(h^*, w) + \left(1 - p_{gen}(h^*)\right) \sum_{n:w_n = w} \mu_n^t \tag{17}$$

$$h_n^* = \sum_n \mu_n^t h_n \tag{18}$$

where $p_{gen}(h^*) \in [0,1]$ is the generating probability (see Equation (8) in [6]), $P_{vocab}(h^*, w)$ is the probability distribution over word w being decoded, h^* is the context vector, a function of the updated word attention μ^t, h_n is the encoder hidden state for the nth word.

During pre-training, the loss is the negative log likelihood, we minimize the loss as

$$L_{abs} = -\frac{1}{T} \sum_{t=1}^{T} \log P_{final}\left(w_t^*, \mu^t\right) \tag{19}$$

where w_t^* is the target word in the reference abstractive summary at the time step t. The coverage mechanism [6] is also used to prevent the abstractor from repeatedly putting the focus on the same point. In each decoder step t, the coverage vector c^t is calculated as follows, which is the sum of attention over all previous timesteps

$$c^t = \sum_{t'=1}^{t-1} \mu^{t'} \tag{20}$$

Moreover, coverage loss L_{cov} is calculated as

$$L_{cov} = \frac{1}{T} \sum_{t=1}^{T} \sum_{n=1}^{N} \min\left(\mu^t, c_n^t\right) \tag{21}$$

We also apply the inconsistency loss as same as [10], the inconsistency loss is calculated by Equation (22)

$$L_{inc} = \sum_{t=1}^{T} \log\left(\frac{1}{|\kappa|} \sum_{n \in \kappa} \alpha_n^t \times \beta_m\right) \tag{22}$$

where κ is the set of top K attended words and T is the number of words in the summary. In conclusion, the final loss of abstraction model is

$$L_{final\_abs} = L_{abs} + \lambda_1 L_{cov} + \lambda_2 L_{inc}. \tag{23}$$

where λ_1, λ_2 are hyper-parameters.

3.5. Training Procedure

The training process of our method is divided into two phases: (1) pre-training phase, (2) full training phase. Without well-trained extractor, the extractor would often select irrelevant sentences, and without well-trained abstractor, the extractor would get noisy reward. We first pre-train the extractor by minimizing L_{ext} in Equation (14) and the abstractor by minimizing L_{final_abs} in Equation (23), respectively, and then, we apply standard policy gradient methods of reinforcement learning to bridge together these two networks and to train the whole model in an end-to-end fashion.

3.5.1. Pre-Training

The sentences with high informativity are our goal of the extractor, the extracted sentences should contain as much information as possible to generate an abstract summary. In order to train the extractive model, we need ground truth labels for each document, but our train corpus only contains human written abstractive summaries, so we need to convert the abstractive summaries to extractive labels. Similar to the extractive model of [22], we compute the ROUGE-L recall score [27] between sentence and the reference abstractive summary, and measure the informativity of each sentence in the document by score. We sort and select the sentences in order from high to low. We add one sentence at a time if the new sentence can increase the score of all the selected. The selected sentences should be the ones that maximize the ROUGE score with respect to gold summaries. Finally, we obtain the ground truth labels and train our extraction model by minimizing Equation 14. We use the ROUGE scores of the selected sentences as sentence-level attention of the corresponding sentences, respectively.

When pre-training, the abstractor takes ground truth sentences of the previously extracted as input. The sentence-level attention of these input sentences is viewed as hard attention, which involves the calculation of attention consistency. In the pre-training stage, we finally get these two well-trained extractor and abstractor.

3.5.2. End-to-End Training

During full training stage, we employ a hybrid extractive-abstractive architecture, with policy gradient of reinforcement learning to bridge together the aforementioned two pre-trained networks. We first use an extractor agent to select important sentences and then employ an abstractor to paraphrase each of these extracted sentences. In this stage, RL training work is as follows: if a good sentence is selected by the extractor, the ROUGE match would be high after the abstractor paraphrase and thus the action is encouraged. If a bad sentence is selected, the generated sentence would not match the reference summary after rewrites and if the ROUGE score is low, the action is discouraged.

3.6. Reinforcement Learning

The abstraction model is the seq2seq model with copy mechanism, which can compress extracted sentences into short text. We use reinforcement learning to connect the two models to form a unified model for optimal training. We regard the existing training data as the environment, and the extractive and abstractive models as agents of reinforcement learning. The agent observes the state from the environment and then performs the extraction and abstraction. The abstraction summary is more similar to the reference summary in the environment, we will get the higher score of the reward. If the final summary does not match the reference summary, will get a lower score. Our reward function is shown in Formula (24).

$$f_{reward} = ROUGEL_{F1}\Big(s_{gold}, f_2(s')\Big) \tag{24}$$

The corresponding relationship between the agent part and the environment part in reinforcement learning is shown in Figure 6.

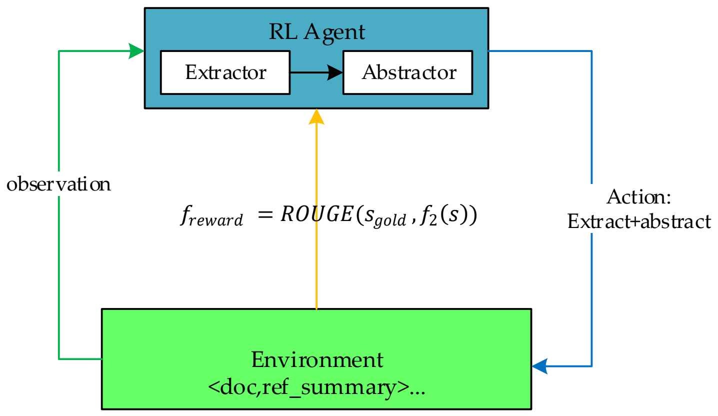
RL Agent
Extractor
Abstractor
observation
$f_{reward} = ROUGE(s_{gold}, f_2(s))$
Action: Extract+abstract
Environment <doc,ref_summary>...

Figure 6. Illustration of reinforcement learning for summary model.

3.7. Redundancy Issue

In our work, we first perform the extraction operation to obtain the extracted sentences semantically independent of each other, which greatly reduces the frequency of common redundancy problems in the abstraction model. As the extracted sentences inescapable have semantic crossover, there is still a little redundancy. For simplicity, we did not use the same coverage mechanism as in the previous work [6] to define a coverage vector, sum up the previous time step's attention, and then use the coverage loss function to avoid repeatedly paying attention to the same position. We employ the same reranking strategy as [7,9]. When conducting beam-search, the same triples are not allowed, and then all combinations of the generated summary sentence bundles are reranked, and the smallest and shortest combination are selected as the final summary.

4. Experiments

4.1. Datasets

The proposed method uses two well-known datasets as our experimental dataset: CNN/Daily Mail and DUC2002. The first dataset proposed by [28] for reading comprehension tasks and then reused for extractive [22] and abstractive text summarization tasks [6]. In recent years, the dataset has been widely used in automatic text summary tasks due to its large data volume and long text content. In this dataset, there are 287,113 data for training, 13,368 for validation, and 11,490 for testing. On average, there are about 28 sentences per document in the training set [22]. The basic statistics of the dataset are shown in Table 1. This dataset includes the anonymous version and the non-anonymous version. The former is that all entity names of the data are replaced by special tag words, while the latter is the original data. We adopt the non-anonymous version. The CNN/Daily Mail data consists of several document summary pairs, each of which corresponds to a few highlighted sentences in manual annotated documents.

Table 1. Basic statistics of the CNN/Daily Mail dataset.

	Train	**Validation**	**Test**
Pairs of data	287,113	13,368	11,490
Article length	749	769	778
Summary length	55	61	58

The second dataset is DUC2002 (http://www-nlpir.nist.gov/projects/duc/guidelines/2002.html), which is only used as a test-only test set since the size of the dataset is small. It contains 567 news articles and the corresponding single-document summarization, or the multi-document summarization generated for the same topics. In our work, we used the single-document summarization task. To verify the generalization ability of models, we use the DUC article as the test input of our trained model. We evaluate the results using the official ROUGE F1 script.

4.2. Detail

We use PyTorch (https://pytorch.org) as our deep learning framework, and then we use the python package PyTorch-Transformers (https://github.com/huggingface/pytorch-transformers) as our BERT encoder, formerly known as pytorch-pretrained-bert. It should be noted that we do not need special tokens (e.g., CLS, SEP) when encoding sentences.

Instead of extracting the list of words from the dataset as previous work ([6,9], etc.), we directly use the vocabulary of the pre-trained BERT model as our vocabulary. The BERT model has two types of vocabulary: the case-sensitive and the case-insensitive, we chose the case-insensitive vocabulary with a total of 30,522 words for the experiment. For comparison, in our experiments, we use the word embedding with 768 dimensions and 1024 dimensions as the word vector representation respectively.

We use the Adam optimizer [29] and apply early stopping based on the validation set. We apply gradient clipping using 2-norm of 2.0. The batch size is 32 for all the training. For all GRU-RNNs, the hidden layer number is set to 1, the hidden state size is set to 256, so the concatenated hidden state size is 512, and the sentence attention context vector also has a dimension of 512. The learning rate of ML (Machine Learning) is set to 3×10^{-4}. The maximum length of the input text sequence sentences is set to 100, the maximum number of sentences is set to 60.

When pre-training the extractor, k is a hyper-parameter of Equation (10), k is set to 256. When pre-training the abstractor, the maximum length of the sentence of the summary is set to 30, λ_1, λ_2 are all set to 1 in Equation (23). When full training end-to-end model, the discount factor of reinforcement learning is set to 0.95, and the early stop factor is set to 3.

4.3. Metrics

Pyrouge package was used to write evaluation scripts for evaluation, and the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) was used as the standard for experimental evaluation. The ROUGE standard was first used in reference [27] and subsequently became the metric for evaluating the generated summary model. The 'similarity' between the generated summary and the reference summary is evaluated by calculating the same number of units. Among them, rouge-1 (unigram), rouge-2 (bi-gram) and rouge-L (the longest common subsequences) are the most widely used in single document abstract.

For all datasets, according to previous works [6,9,10], we use ROUGE-1, ROUGE-2, and ROUGE-L on full-length F1 as evaluation metrics in the reported experimental results.

4.4. Baselines

To further illustrate the superiority of the proposed model over two datasets, we compare it with several baseline models. Note that, most models serve as baselines on the CNN/Daily Mail dataset. All comparison models are described in detail as follows:

- Leading sentences (Lead-3): It directly extracts the first three sentences of the article as a summary. This model as extractive baseline.
- Refresh: The model proposed by [3], takes the reinforcement learning objective as the extraction baseline and optimizes the rouge evaluation index globally.
- SummaRuNNer: It is proposed by [22] to generate the summary by extracting some key subset of the content for the article, as an extractive baseline.
- HSSAS: It is proposed by [21] to employ the self-attention mechanism to create a good sentence and document embeddings, as an extractive baseline.
- NeuSum: It is proposed by [30] to extract the document summarization by jointly learning to score and select sentences, as an extractive baseline.

- Pointer-generator+coverage: It is proposed by [6] to copy words from the source article and retain the ability to generate new words, as an abstractive baseline.
- Inconsistency loss: The method proposed by [10], which uses sentence-level attention to modulate the word-level attention, introduces inconsistency loss function to penalize the inconsistency between two attentions, as an abstractive baseline.
- DCA: The method proposed by [11], which encodes long text with the deep communication agents and then connects to a single decoder to generate a focused and coherent summary through reinforcement learning, is the best abstract model in 2018 and serve as an abstractive baseline.
- RNN-ext+abs+rl+rerank: It is proposed by [9], which first selects salient sentences and then rewrites them abstractly to generate a concise overall summary, as an abstractive baseline.

5. Result and Analysis

Due to our model employing a hybrid approach of extractive and abstractive, we can not only generate the final abstract summaries, but also the extract summaries. So three groups of experiments were conducted on CNN/Daily Mail dataset: the extraction experimental group, the complete experimental group, and the comparison experimental group. In the comparison of the experimental group, we compared the effects of the base one and the large model of BERT, the effects of the tokenizers, and the effects of reinforcement learning. The results of extractive experiments are shown in Table 2, and full experiments are shown in Table 3.

Table 2. Performance comparison of models with respect to the extractive baselines on CNN/Daily Mail.

Model	R-1	R-2	R-L	R-AVG
Lead-3 [6]	40.34	17.70	36.57	31.54
Refresh [3]	40.00	18.20	36.60	31.60
SummaRuNNer [22]	39.9	16.3	35.1	30.43
HSSAS [21]	42.3	17.8	37.6	32.57
NeuSUM [30]	41.59	19.01	**37.98**	32.86
(m1) BEAR (ext+base)	42.43	**20.36**	37.23	33.34
(m2) BEAR (ext +large)	**42.54**	20.35	37.24	**33.38**

Table 3. Performance comparison of models with respect to the abstractive baselines on CNN/Daily Mail.

Model	R-1	R-2	R-L	R-AVG
Pointer Generator + Coverage [6]	39.53	17.28	36.38	31.06
Inconsistency loss [10]	40.68	17.97	37.13	31.93
DCA [11]	41.69	19.47	37.92	33.11
Rnn-ext + abs + RL + rerank [9]	40.88	17.80	38.54	32.41
(m3) BEAR (ext + abs + base)	39.84	18.83	37.35	32.01
(m4) BEAR (ext + abs + RL + base)	40.91	19.88	38.45	33.08
(m5) BEAR (base + nature)	40.95	17.89	38.55	32.80
(m6) BEAR (base + WordPiece)	41.76	20.20	39.39	33.78
(m7) BEAR (large + WordPiece)	**41.95**	**20.26**	**39.49**	**33.9**

5.1. Result

In the extraction experimental group, only sentences are selected without rewriting, in other words, the abstraction fitting function is f2 (s′) = s′, and the rest are the same as the full experiment. As shown in Table 2, the experimental results of m1, m2 have many advantages in the ROUGE-1 standard compare with extractive baselines [3,6,22,30], and are slightly higher than the method in the literature [21]. In the ROUGE-2 standard, our method has an obvious effect. In the ROUGE-L standard, our model is superior to the models Refresh [3] and Lead-3 [6], but lower than the model NeuSUM [30].

In the complete experimental group, the experimental results of our model are significantly improved compared with the earlier work and baseline model. As we all know, we obtain the highest ROUGE score for the current abstraction summary in the CNN/Daily Mail dataset. As shown in Table 3, compared with the baseline model [9], the scores of ROUGE-1, ROUGE-2, ROUGE-L, and R-AVG are respectively improved 1.07%, 2.46%, 0.95%, and 1.46%.Compared with the highest scoring model [11] in 2018, the score of Rouge-1 is only slightly improved, but these two metrics of Rouge-2 and Rouge-L are significantly improved, and these two models we have fully trained are also significantly better than the abstraction methods [6,10].

We also compared our best model with the best extractive baseline model and the best abstractive baseline model. The comparison bar chart is shown in Figure 7.

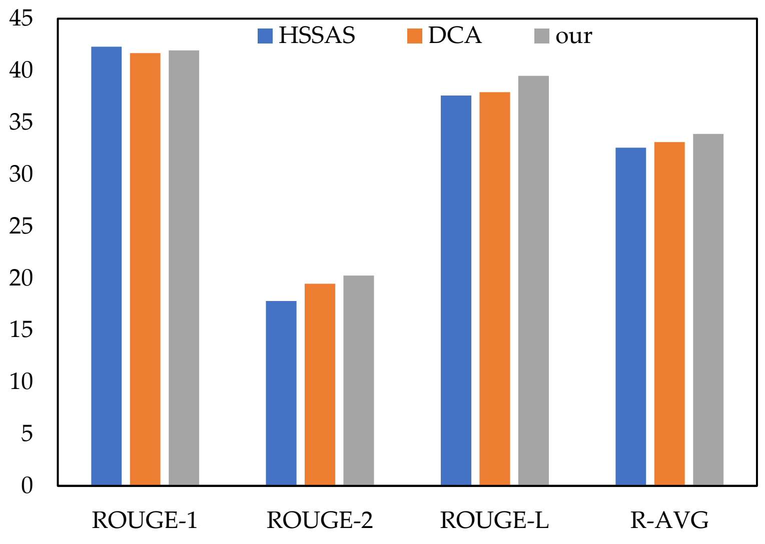

Figure 7. Performance comparison with respect to the best baselines on CNN/Daily Mail.

5.2. *Ablation Study*

In the comparison group, three comparison experiments are analyzed as follows:

Pre-trained BERT model with multiple versions, we used two versions in experiments: the base-uncased and the large-uncased. In terms of pre-training complexity and the dimension of word embeddings, the large-uncased version should be more powerful than the base-uncased. As shown in Figure 8, in our extraction experiment (m1, m2) and the full experiment (m6, m7), the experimental results are very comparable, and the large results are slightly better than the base version results, and the difference in the performance of these two versions was not as much as we expected.

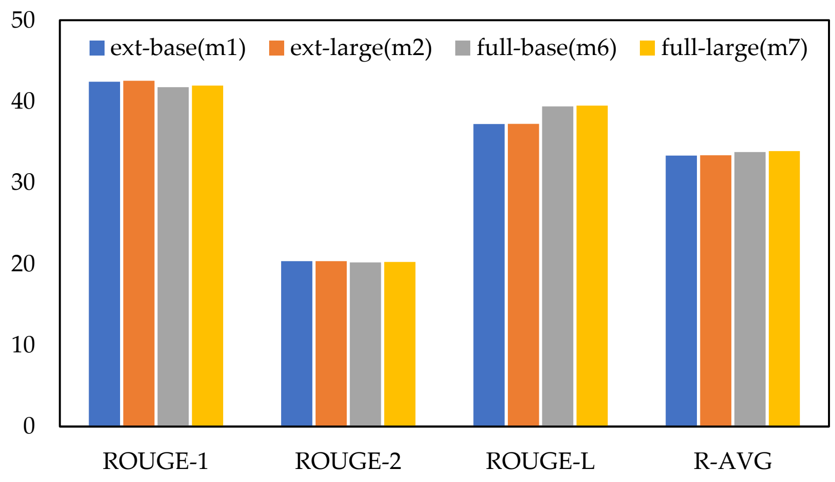

Figure 8. Performance comparison of the $\text{BERT}_{\text{base}}$ mode with respect to the $\text{BERT}_{\text{large}}$ model of BERT.

When comparing the effect of WordPiece (m6) and natural (m5) tokenizer, the improvement of WordPiece tokenizer is more obvious than that of the traditional tokenizer, with an average increase of nearly 1 percentage point. There are two main reasons for this. Firstly, BERT adopts the WordPiece tokenizer in their training process, and the word embedding obtained is more suitable for the tokenizer. Secondly, it shows that the WordPiece tokenizer is more effective than natural tokenizer, and the new tokenizer can capture richer semantic features. The comparison between two tokenizers is shown in Figure 9.

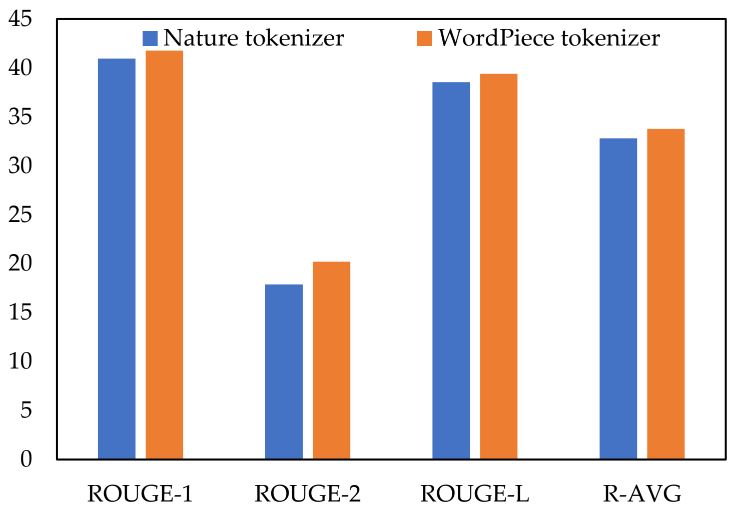

Figure 9. Performance comparison of tokenizer mode.

The reinforcement learning method was adopted in the experiment except model m3. Since many previous works [7,10,11] have proved the effectiveness of reinforcement learning, this paper only used the combination of the base model and WordPiece tokenizer for comparison and did not use other combinations of the experiment for cross-comparison. By comparing the experimental model (m3) with the model (m4), the scores of the model (m4) using reinforcement learning are more than 1 percentage point higher than that of the model (m3), which proves that the effect of reinforcement learning is quite significant in our experiment. The effect of reinforcement learning is shown in Figure 10.

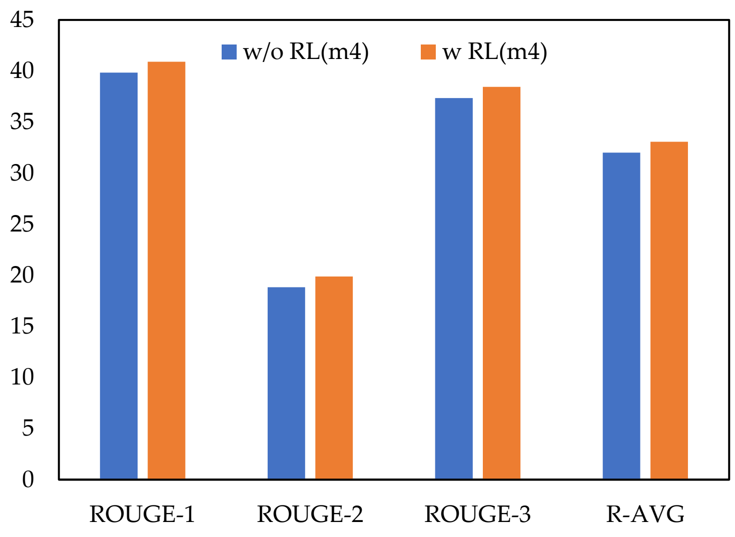

Figure 10. Performance comparison of the effect of reinforcement learning.

Also, by comparing model (m3) with the rnn-ext+abs model in paper [9], all the scores of the model (m3) are more than 1 percentage point higher, which further proves that the word embedding vector of BERT has the same result as reinforcement learning in our experiment and has the same significant effect.

5.3. *Generalization*

Because the DUC2002 dataset is too small to train such a complex model, we only use it as our test dataset. To test the generalization of our model, we directly use the trained model to summarize the DUC article. Table 4 shows the generalization of our method to the DUC2002 dataset. The results show that our model has better generalization ability than the other two models. The results of pointer-generator model and rnn-ext+abs+RL model are obtained from [9].

Table 4. Generalization to DUC-2002(F1).

Models	R-1	R-2	R-L	R-AVG
Pointer-generator	37.22	15.78	33.90	28.97
Rnn-ext + abs + RL	39.46	17.34	36.72	31.17
BEAR (m7)	40.53	19.85	38.37	32.92

5.4. *Redundancy Issue*

Extracted sentences can effectively reduce the redundancy in the abstract model from the source. For there may be some semantic cross, we use the rerank strategy mentioned earlier. As shown in Table 3, our model evaluation score is higher than these two models in [6,10]. As shown in Table 5, the sample sentences are concise and clear. Experiments show that our strategy can effectively reduce redundancy and improve the accuracy of the evaluation.

Table 5. Speed comparison between BEAR, rnn-ext+abs+RL, and pointer-generator.

	Pre-Traing	Training	Total Training	Test	GPU
BEAR	19 h	6 h	22 h	0.67 h	K80
Rnn-ext + abs + RL	4.15 h	15.56 h	19.71 h		K40
Pointer-generator			76 h		K40

5.5. Training Speed

In the experiment, we train on a single Tesla K80 GPU, the extracted pre-training time is about 6 h, the abstracted pre-training time is about 19 h, the mixed training time is about 3 h, and the test time is about 40 min. Since extraction training and abstraction training can be carried out in parallel, the total experimental time is about 23 h. When training the summary model, we use the extracted statement training model instead of the whole document, and we can draw a conclusion that training with a few sentences is faster than training with the whole document. The experiment shows that this paper is faster than the pointer-generator [6] in the calculation, but the speed is not as fast as that of [9], because the high dimension of BERT word embedding increases the computational complexity. The speed comparison results are shown in Table 5.

In addition, the resource consumption of deep learning is increasingly a concern for researchers, and we also estimate the carbon footprint of one training. According to the search result, the power of Tesla K80 GPU is 0.3 kw/h, and the average carbon emission is 0.433 kg/kwh (data from the U.S. EIA). It can be estimated that our carbon footprint is about 3 kg.

5.6. Case Study

We present two examples in Table 6 for comparison. As we can intuitively see from the sample of the final summary, it is more concise, as shown in Table 6. The strategy of eliminating repetition can effectively reduce semantic crossover and make the content of the summary more compact, which not only maintains semantic relevance, but also takes into account the readability and fluency of language.

Table 6. Examples of generated summaries on CNN/Daily Mail dataset.

Reference:
"17 americans were exposed to the ebola virus while in sierra leone in march", "another person was diagnosed with the disease and taken to hospital in maryland", "national institutes of health says the patient is in fair condition after weeks of treatment."
Model (m4):
five americans were monitored for three weeks at an omaha hospital. one of the five had a heart—related issue on saturday and has been discharged but hasn't left the area, taylor wilson wrote. they were exposed to ebola in sierra leone in march but none developed the virus. the others have already gone home.
Model (m7):
five americans were monitored for three weeks at an omaha, nebraska. they all had contact with a colleague who was diagnosed with the disease. the last of 17 patients who were being monitored are expected to be released by thursday. more than 10,000 people have died in a west african epidemic of ebola.

Also, we can find that the first sentence of model m4 and model m7 is basically the same, while the latter three sentences are quite different. Semantically, model m7 is more relevant to the reference than model m4, and the second and fourth sentences of model m4 are somewhat off topic, which is the main reason for the low score. The final summary of model m7, although different from the reference, talks about the same thing.

6. Conclusions and Future Work

At present, BERT has achieved the most advanced performance in many NLP tasks, but few works combine it with the extract model and abstract model for text summarization by the strategy gradient of reinforcement learning. We propose a new method to combine these methods into a unified model that has been validated in summary tasks or other tasks. Experiments show that the model proposed in this paper achieves the best results in the CNN/Daily Mail dataset. In the future, we will select another pre-training model that is more suitable for the generative task and combine the fine-tuning pre-training model with the abstractive summary task.

Author Contributions: Conceptualization, Q.W. and P.L.; Funding acquisition, P.L.; Methodology, Q.W.; Project administration, Z.Z.; Resources, P.L. and Z.Z.; Software, Q.W.; Validation, Q.W.; Visualization, Q.W.; Writing—original draft, Q.W.; Writing—review and editing, H.Y., Q.Z., and L.Z.

Funding: This work was supported by the National Social Science Fund(19BYY076), the Science Foundation of the Ministry of Education of China (no. 14YJC860042), and the Shandong Provincial Social Science Planning Project (no. 19BJCJ51/18CXWJ01/18BJYJ04/16CFXJ18).

Acknowledgments: Thanks to all commenters for their valuable and constructive comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cheng, J.; Lapata, M. Neural summarization by extracting sentences and words. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2016; Volume 1, pp. 484–494.
2. Nallapati, R.; Zhou, B.; Santos, C.N.D.; Gulcehre, C.; Xiang, B. Abstractive text summarization using sequence-to-sequence rnns and beyond. In Proceedings of the SIGNLL Conference on Computational Natural Language Learning, Berlin, Germany, 11–12 August 2016; p. 280.
3. Narayan, S.; Cohen, S.B.; Lapata, M. Ranking sentences for extractive summarization with reinforcement learning. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, USA, 2–4 June 2018; pp. 1747–1759.
4. Yasunaga, M.; Zhang, R.; Meelu, K.; Pareek, A.; Srinivasan, K.; Radev, D. Graph-based neural multi-document summarization. In Proceedings of the 21st Conference on Computational Natural Language Learning, Vancouver, BC, Canada, 3–4 August 2017; pp. 452–462.
5. Rush, A.M.; Chopra, S.; Weston, J. A neural attention model for abstractive sentence summarization. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 379–389.
6. See, A.; Liu, P.J.; Manning, C.D. Get to the point: Summarization with pointer-generator networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 30 July–4 August 2017; pp. 1073–1083.
7. Paulus, R.; Xiong, C.; Socher, R. A deep reinforced model for abstractive summarization. In Proceedings of the Sixth International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
8. Li, P.; Bing, L.; Lam, W. Actor-critic based training framework for abstractive summarization. *arXiv* **2018**, arXiv:1803.11070V2.
9. Chen, Y.C.; Bansal, M. Fast abstractive summarization with reinforce-selected sentence rewriting. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; pp. 675–686.
10. Hsu, W.T.; Lin, C.K.; Lee, M.Y.; Min, K.; Tang, J.; Sun, M. A unified model for extractive and abstractive summarization using inconsistency loss. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; pp. 132–141.
11. Celikyilmaz, A.; Bosselut, A.; He, X.; Choi, Y. Deep communicating agents for abstractive summarization. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, USA, 2–4 June 2018; pp. 1662–1675.
12. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.

13. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 3104–3112.
14. Cho, K.; Van, M.B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1724–1734.
15. Gu, J.; Lu, Z.; Li, H.; Li, V.O.K. Incorporating copying mechanism in sequence-to-sequence learning. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2016; pp. 1631–1640.
16. Vinyals, O.; Fortunato, M.; Jaitly, N. Pointer networks. In Proceedings of the International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–10 December 2015.
17. Sutton, R.S.; Barto, A.G. *Introduction to Reinforcement Learning Cambridge*; MIT Press: Cambridge, MA, USA, 1998; Volume 2.
18. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *Comput. Sci.* **2014**.
19. Lin, Z.; Feng, M.; Santos, C.N.D.; Yu, M.; Xiang, B.; Zhou, B. A Structured Self-Attentive Sentence Embedding. *arXiv* **2017**, arXiv:1703.03130.
20. Li, Z.; Wei, Y.; Zhang, Y.; Yang, Q. Hierarchical attention transfer network for cross-domain sentiment classification. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
21. Al-Sabahi, K.; Zuping, Z.; Nadher, M. A hierarchical structured self-attentive model for extractive document summarization (hssas). *IEEE Access* **2018**. [CrossRef]
22. Nallapati, R.; Zhai, F.; Zhou, B. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 3075–3081.
23. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, 5998–6008.
24. Wu, Y.; Schuster, M.; Chen, Z.; Le, Q.V.; Norouzi, M.; Macherey, W. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv* **2016**, arXiv:1609.08144V2.
25. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
26. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. Lstm: A search space odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *28*, 2222–2232. [CrossRef] [PubMed]
27. Lin, C.Y. ROUGE: A Package for Automatic Evaluation of summaries. In Proceedings of the Workshop on Text Summarization Branches Out, Barcelona, Spain, 25–26 July 2004; pp. 74–81.
28. Hermann, K.M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; Blunsom, P. Teaching Machines to Read and Comprehend. In Proceedings of the 28th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 1693–1701.
29. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *Comput. Sci.* **2014**.
30. Zhou, Q.; Yang, N.; Wei, F.; Huang, S.; Zhou, M.; Zhao, T. Neural document summarization by jointly learning to score and select sentences. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; Volume 1, pp. 654–663.