applied sciences

MDPI

Article

An Abstractive Summarization Model Based on Joint-Attention Mechanism and a Priori Knowledge

Yuanyuan Li *, Yuan Huang, Weijian Huang, Junhao Yu and Zheng Huang

School of Information and Electrical Engineering, Hebei University of Engineering, Handan 056038, China
* Correspondence: lyuanyuanm@163.com

Abstract: An abstractive summarization model based on the joint-attention mechanism and a priori knowledge is proposed to address the problems of the inadequate semantic understanding of text and summaries that do not conform to human language habits in abstractive summary models. Word vectors that are most relevant to the original text should be selected first. Second, the original text is represented in two dimensions—word-level and sentence-level, as word vectors and sentence vectors, respectively. After this processing, there will be not only a relationship between word-level vectors but also a relationship between sentence-level vectors, and the decoder discriminates between word-level and sentence-level vectors based on their relationship with the hidden state of the decoder. Then, the pointer generation network is improved using a priori knowledge. Finally, reinforcement learning is used to improve the quality of the generated summaries. Experiments on two classical datasets, CNN/DailyMail and DUC 2004, show that the model has good performance and effectively improves the quality of generated summaries.

Keywords: abstractive summarization; joint-attention mechanism; prior knowledge; reinforcement learning

Citation: Li, Y.; Huang, Y.; Huang, W.; Yu, J.; Huang, Z. An Abstractive Summarization Model Based on Joint-Attention Mechanism and a Priori Knowledge. *Appl. Sci.* **2023**, *13*, 4610. https://doi.org/10.3390/app13074610

Academic Editors: Ahmed Rafea, Julian Szymanski and Krzysztof Koszela

Received: 1 February 2023
Revised: 20 March 2023
Accepted: 4 April 2023
Published: 5 April 2023

1. Introduction

Before the rapid development of the Internet and common electronic devices, people obtained news and other information from newspapers and TV. Now, network platforms have become an important way for people to obtain and share information. These trends have led to an exponential growth of information on the Internet. With such a wealth of information, people are looking for ways to quickly access available information. Automatic text summarization has emerged at this historic moment, effectively alleviating the problem of information overload. Automatic text summarization is designed to convert text or collections of text into short summaries containing key information. Text summaries can be divided into single-document summaries and multi-document summaries according to the input type. Single-document abstracts [1] generate abstracts from a given document, and multi-document abstracts [2] generate abstracts from a given set of subject-related documents. According to the output type, the text summary method can be divided into extractive summaries [3] and abstractive summaries [4–10]. Extractive abstracts extract keywords and key phrases from the source text, thus forming the abstract. Abstractive abstracts allow the generation of new words and phrases to make up the abstract based on the original text. Compared with the extractive summary, the abstractive summary is more in line with the human habit of generating an abstract, although it is complicated to implement. First off, generative text summarizing reduces reading time and increases reading effectiveness by avoiding the extraction of extensive content from the source text, instead creating brief and to-the-point summaries. Second, rather than choosing specific sentences from the original text, people who read an article to learn its main points typically summarize it using the original text as a guide. In conclusion, generative text summarizing is more adaptable in its information-gathering and is closer to how humans produce summaries. In addition, generative summarization requires machines to imitate

humans and understand more semantic information in the original text, which is even more challenging. The significance of examining it has also been reinforced by the growing rise of generative text summaries in the mainstream of research. Therefore, this article mainly studies generative text summarization.

An automatic text summarization model was first designed by Luhn [11] in 1958, which kicked off automatic text summarization. Subsequently, graph-based abstractive text summarization [4], linear programming-based abstractive text summarization [5,6], semantic-based abstractive text summarization [7,8], and template-based abstractive text summarization [9,10] followed. With the rise of deep learning in the fields of natural language processing, more and more abstractive text summary models based on deep learning have emerged. Deep-learning-based methods surpass traditional generation methods in multiple languages and evaluation metrics [12]. Deep-learning-based automatic text summarization emulates how people summarize by remembering abstracts based on their comprehension of the original material. As a result, deep-learning-based text summarization technology has received considerable attention from both academia and industry due to its significant research significance and wide range of potential applications.

Among many automatic abstract generation models based on deep learning, the most popular one is the sequence-to-sequence (Seq2seq) [13] model, which is presented in Figure 1. The model generally consists of two parts: the first part is the encoder part, which is used to characterize the input sequence of length n. The second part is the decoder part, which is used to build the representation extracted from the encoder into a mapping of sequence with an output length of m. Most of the existing models are based on this model for improvement. Rush et al. [14] proposed an abstractive text summarization model ABS based on the Seq2seq model and incorporated the attention mechanism into the model. The Seq2seq model incorporating attention is shown in Figure 2. Based on the Seq2seq model, the researchers studied the different effects of different neural networks used for encoder and decoder. Chopra et al. [15] constructed decoders based on recurrent neural networks(RNN). Nallapati et al. [16] changed the encoder to a recurrent neural network as well, making it a standard recurrent neural network-based encoder–decoder model. Gehring et al. [17] replaced the recurrent neural network used in the sequence-to-sequence model with a convolutional neural network, thus proposing the ConvS2S model. and the model has achieved good results in automatic text summarization tasks. The ConvS2S model not only realizes parallel computation at different positions in the sequence, but also can capture the long-range dependencies between words in a sentence through the hierarchical structure, which makes it possible to handle complex sentence information.

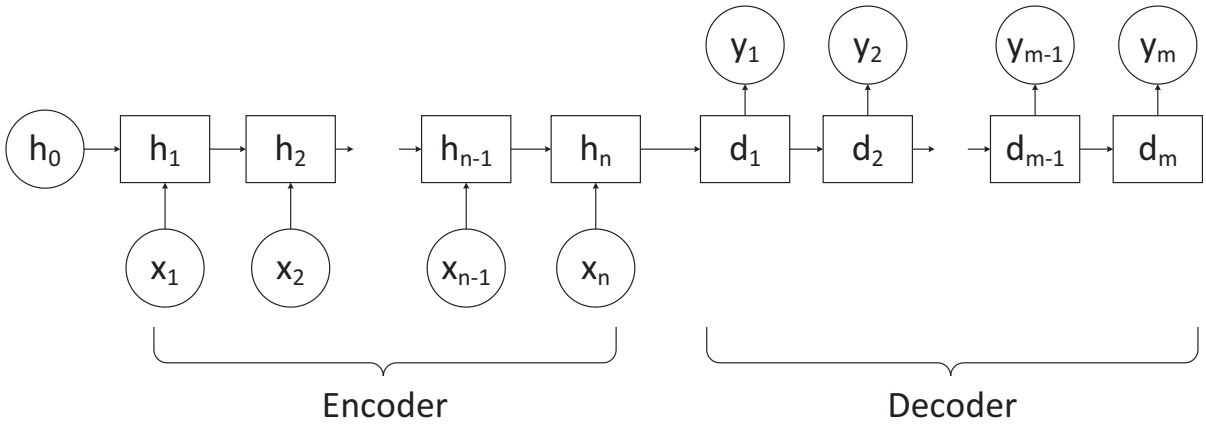

Figure 1. Seq2seq model.

The word list of the model is composed of high-frequency words, so there will be some words that do not appear in the word list, and we call such words out-of-vocabulary (OOV).To solve the problem of OOV, Nallapati et al. [16] also put forward the Generator-

Pointer model. The model via pointer P_{gen} determines whether the summary is copied from the source text or whether the generator generates a new word. During encoder–decoder training, the decoder inputs the correct reference summary, while during prediction, the decoder inputs the output from the previous time step. Once the output from the previous step is incorrect, a series of deviations will occur, resulting in poor summary quality. This is the issue of exposure bias. During the research, we also noticed this issue. Paulus et al. [18] first proposed to address the problem of exposure bias based on reinforcement learning. The maximum-likelihood cross-entropy loss function and a strategy gradient for reinforcement learning are combined in Paulus's paper to propose new learning objectives. This reduces bias in the reinforcement learning process.

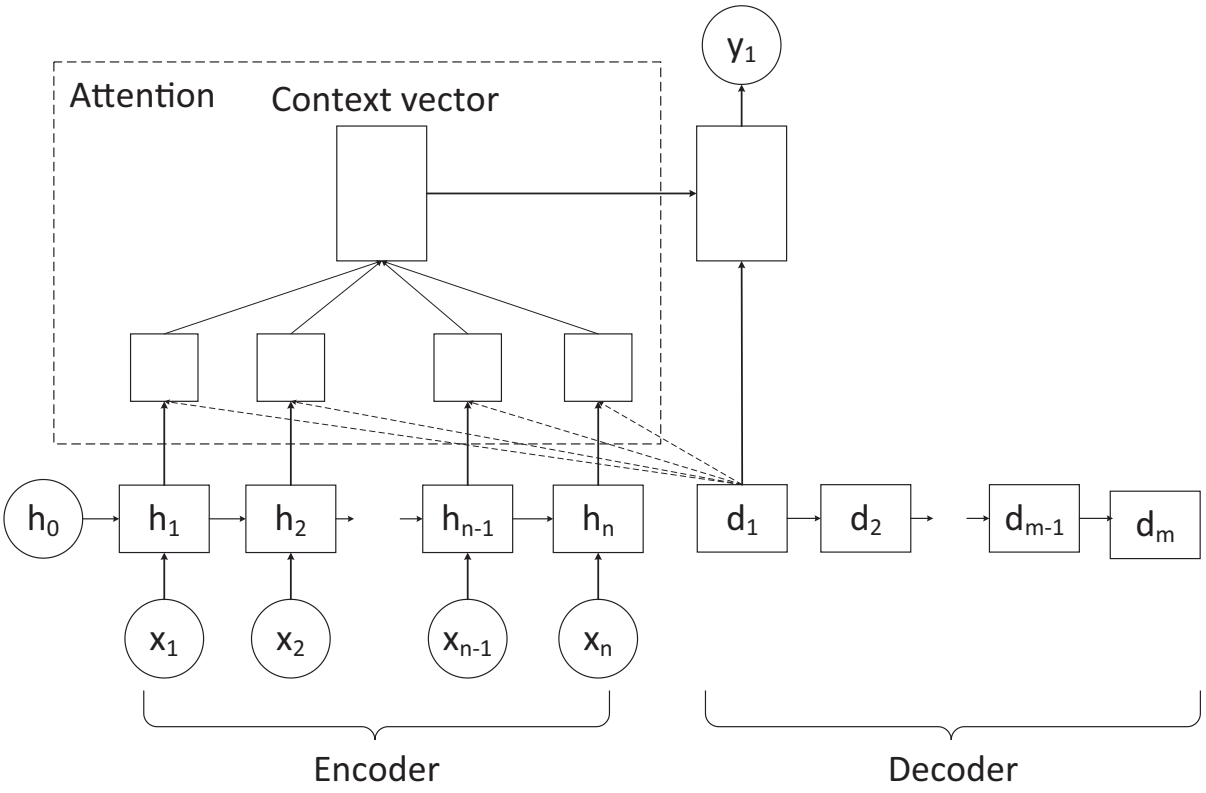

Figure 2. Seq2seq model with Attention.

We identified several issues with the generative summary model based on the aforementioned research findings: (1) Researchers cannot truly determine the level of correlation between the vectors stored by neural networks and the original text throughout the study process because of the black box nature of neural networks. (2) It is challenging to provide coherent summaries since abstract generation models often only focus on the link between single-word vectors. (3) Humans typically combine the text's prior information when creating abstractions, which results in abstracts that are more semantically rich. Current models frequently lack historical context. We will conduct our research from the perspective of introducing prior knowledge and richer semantic information. The main contributions of this paper are as follows:

1. We improve the encoder part of the original codec model by adding a weighted summation (WS) process, so that the encoder can determine how well the vectors generated by the neural network match the original text and assign higher weights to the important parts of the original text.
2. To improve the quality of the generated summaries, we use the word–sentence-level joint-attention mechanism (JAM) to force the decoder to focus not only on the keywords in the source text but also on the sentences where the keywords are located.

Furthermore, we combine prior knowledge (PK) to enable the decoder to obtain the original information of the source text on the one hand, avoiding information loss caused by layer-by-layer transmission in the abstract generation process; on the other hand, we transmit higher-level abstract semantic information to the decoder, enabling the generation of abstracts more in line with human reading habits.

3. We introduce reinforcement learning (RL) for optimization to address the problems of exposure bias and inconsistency between evaluation metrics and loss functions. The readability and fluency of the summary are ensured.
4. The abstractive summary model based on the joint-attention mechanism and prior knowledge (JPATS) was used on two datasets, CNN/DailyMail and DUC 2004, and the results showed that our model outperformed the existing baseline model.

2. Related Work

2.1. Attentional Mechanism

An attention mechanism involves the idea of placing attention on something at a certain moment and ignoring something else. In the text summarization task, it is decided which part of the input should be focused on in the text summarization task. Rush et al. [13] first proposed an attention-based mechanism for encoders and achieved significant performance improvements. The Bahdanau attention mechanism [19] is commonly used in text summary tasks, which is a micro-attention model without strict one-way alignment restrictions. The self-attention mechanism [20] is a variant of the attention mechanism that allows the machine to notice correlations between different parts of the whole input. Later, researchers conducted research based on the attention mechanism, resulting in the generation of numerous text summarization models based on the attention mechanism. Zheng et al. [21] proposed an unsupervised extractive summary model with dual-attention mechanism, named DAS for news text summary extraction. The model contains the News-Tweet follow module and the News-self follow module to capture the correlation between news and linked tweets. DAS achieves better results than state-of-the-art unsupervised models.

Hakami et al. [22] concluded that the traditional attention mechanism may lead to off-topic generated abstracts, and thus proposed a dual-attention mechanism as a way to generate accurate and reasonable abstracts, achieving 96.7 % accuracy and high performance. Qiu et al. [23] improved the multi-headed attention mechanism at the model's encoding end, giving more weight to correct information and improving the coverage mechanism to improve the summary model's quality. Liu et al. [24] proposed a SEASum model, which consists of two encoders: a semantic encoder and a syntactic encoder. Cascading or parallel assembly is used to assemble the two encoders, and the encoder outputs are fused and fed to the decoder to obtain the digest. The validity of the model is demonstrated experimentally. The model fuses the features of two encoders using a multi-headed attention mechanism, with the semantic features as query values and the sum of the two features as key and value values. Key and value essentially represent two independent matrices, which are the results of different linear transformations of the original sequence.The ability to use multiple attention mechanisms to capture information and assign attentional weights rationally.

2.2. Reinforcement Learning

Reinforcement learning is a learning mechanism that learns how to map from states to behaviors to maximize the reward obtained. Naraya et al. [25] defined the task of extractive single-document summarization as a sentence ranking task, and combined the maximum-likelihood cross-entropy loss and the reward function of the RL policy gradient during training to learn a direct global optimization of the Rouge metric, allowing the model to learn to perform sentence ranking at the time of summary generation. Chen et al. [26] combined extractive and abstractive approaches to improve accuracy while using parallel decoding to allow the speed of model training enhancement. Moreover, a

gradient-based technique for reinforcement learning is applied to go for joint training. Reinforcement learning was used by Chen et al. to enhance the performance of extracting and producing associative summaries, Paulus et al. to lessen the bias of the generative summary model, and Naraya et al. to enhance the performance of extracted summaries. The efficacy of extracted summaries, generative summaries, and combined extracted and produced summaries can all be improved via reinforcement learning. Liang et al. [27] added selection gates to remove redundant information after the encoder and optimized the evaluation metrics by reinforcement learning strategy to obtain better results. Liao et al. [28] fused long-range dependencies into training based on reinforcement learning, and the proposed model can capture long-range dependencies and generate summaries containing more important information. Kryscinski et al. [29] divided the decoder into two parts, one for the contextual network, which is responsible for extracting information from the source document, and the other for optimizing the evaluation metrics using reinforcement learning to encourage the generation of words and phrases that are not present in the source text.

3. Model

In this paper, we present a model where the source text is first processed with word2vec to obtain the word-embedding w_n representation of each word. In addition, then input to the neural network, in turn, to obtain the hidden vector h_n for each word. Unlike machine translation tasks that need to retain all the information in the original text, text summaries need to retain only the most important information. Thus, inspired by the idea of a self-attention mechanism, we match the trained hidden vectors with the original text and assign higher weights to the vectors that are more relevant to the information in the original text. After this operation, further relationships between words h'_n are obtained, enabling the decoder to obtain source text information with focus. The sentence-level vectors are obtained by averaging the word vectors, forcing the decoder to focus on the keywords while focusing on the sentences in which the keywords are located. The word-level and sentence-level vectors are used to calculate the score weights with the hidden states under each moment of the decoder, respectively, and this is used to improve the pointer network to generate more coherent and human-readable summaries. Moreover, this paper uses the original word vector without any training to obtain the sentence embedding as the prior knowledge to guide the copy generation of the pointer network. Finally, reinforcement learning is used to improve the evaluation metric Rouge and alleviate the problem of inconsistency between the evaluation metric and the loss function.The model is shown in Figure 3.

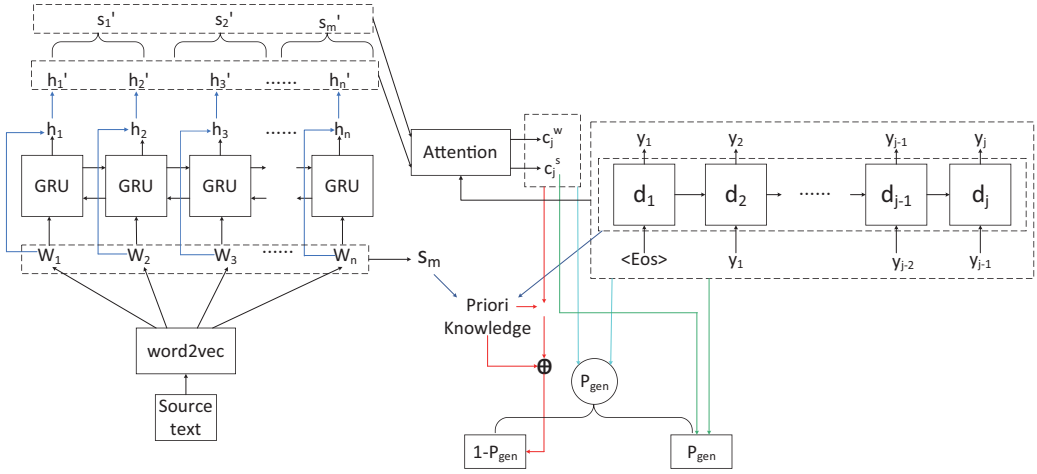

Figure 3. An abstractive summarization model based on joint-attention mechanism and a priori knowledge.

3.1. Problem Formulation

Given the source text $X = \{x_1, x_2, x_3, ..., x_l\}$, l is the sentence length, $X \in V_s$, V_s is the source text vocabulary; generate the summary $Y' = \{y'_1, y'_2, y'_3, ..., y'_m\}$, m is the summary length, $Y' \in V_t$, V_t is the target summary. Generating a summary needs to satisfy $m \ll l$. If $Y' \subseteq X$, it means that all summaries are derived from the source text, and we define this case as an extractive summary. If $Y' \nsubseteq X$, it means that not all the summary words are from the source text, and this case is defined as an abstractive summary. In the model training stage, the input of the decoder part is the correct abstract, that is, the learning function $f : X \rightarrow Y$, and Y is the reference abstract extracted manually. The purpose of designing the model is to make the abstract Y' generated by the model close to the standard abstract Y. In the test phase, the trained function f is used to generate a summary from the source text.

3.2. Encoding Side

To solve the RNN long-range dependency problem, a deformed long short-term memory (LSTM) or gated recurrent unit (GRU) of RNN is usually used in the codec model. Compared with LSTM, GRU is easier to train and can improve the training rate to a great extent. Therefore, GRU is used as the encoder in this paper. GRU is defined as follows.

$$u_t = \sigma(W_u[w_t, h_{t-1}]) \tag{1}$$

$$r_t = \sigma(W_r[w_t, h_{t-1}]) \tag{2}$$

$$\dot{h}_t = \tanh(W_h w_t + U_h(r_t \odot h_{t-1})) \tag{3}$$

$$h_t = (1 - u_t) \odot h_{t-1} + u_t \odot \dot{h}_t \tag{4}$$

where W_u, W_r, and W_h are the weight matrices, w_t is the current input, and $h_(t-1)$ is the hidden state vector passed in from the previous time node, and this hidden state vector contains the relevant information of the previous node.

In this paper, we use a bidirectional GRU(Bi-GRU), as shown in Figure 4, to retain both forward and backward time step information. The forward network reads the word-embedding vector from left to right to obtain the forward GRU hidden state $(\overrightarrow{h}_1, \overrightarrow{h}_2, ..., \overrightarrow{h}_n)$, and the backward network reads the word-embedding vector from right to left to obtain the backward GRU hidden state $(\overleftarrow{h}_1, \overleftarrow{h}_2, ..., \overleftarrow{h}_n)$. The two hidden state formulas are as follows.

$$\overrightarrow{h}_t = GRU(w_t, \overrightarrow{h}_{t-1}) \tag{5}$$

$$\overleftarrow{h}_t = GRU(w_t, \overleftarrow{h}_{t+1}) \tag{6}$$

The initial state of Bi-GRU is set as a zero vector, i.e., $\overrightarrow{h}_1 = 0$, $\overleftarrow{h}_n = 0$. The forward and backward hidden states are cascaded to obtain the final hidden state. Expressed as: $h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$.

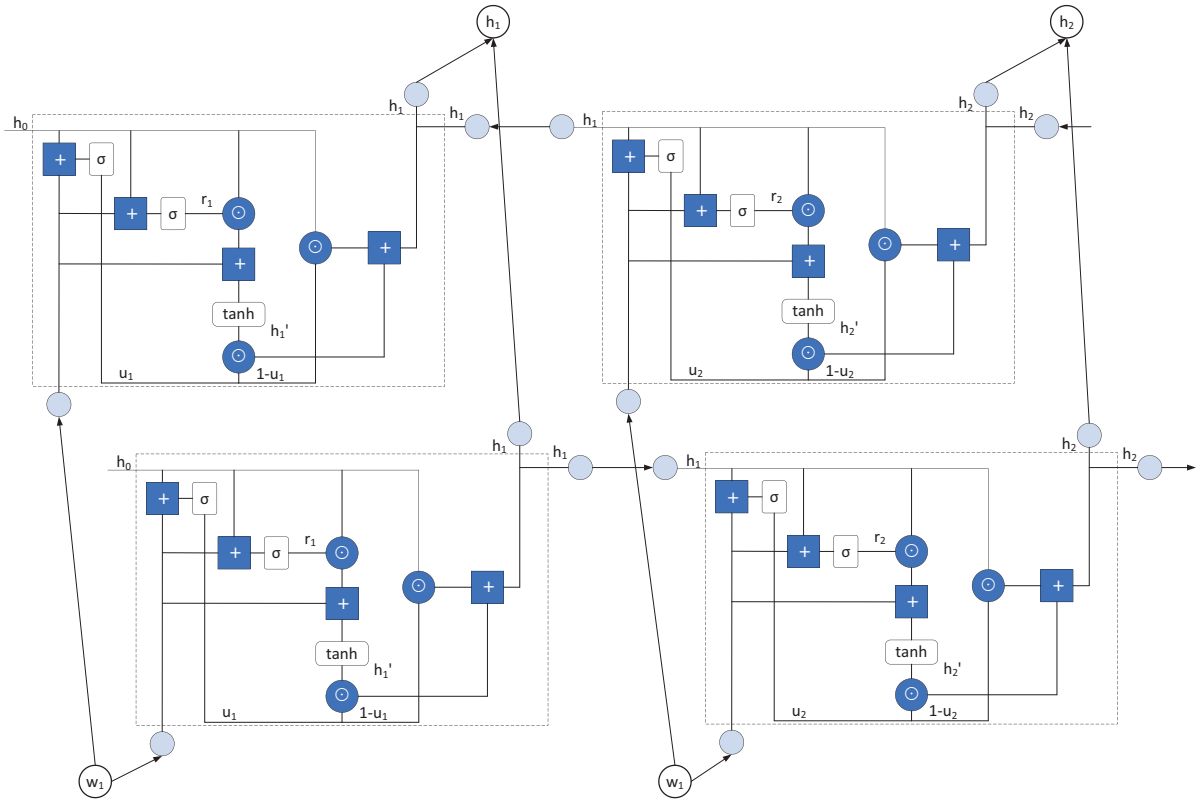

Figure 4. Bi-GRU.

The self-attention mechanism can determine how relevant the current word is to other words in the sentence. Less reliance on external information and better at capturing the internal relevance of data or features. Inspired by this, we calculate the correlation between the hidden vectors trained by the neural network and the source text word vectors to avoid deviating from the original meaning. In addition, it is also possible to give higher weight to keywords, making the decoder focus on important information in the decoding process. The process is as follows.

1. First, obtain the similarity matrix, as shown in Figure 5:

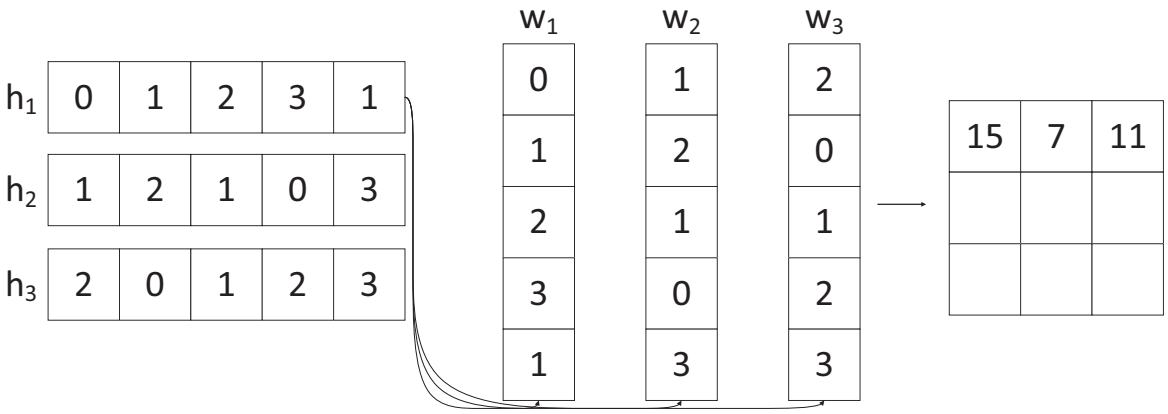

Figure 5. Similarity matrix.

2. Normalization, as shown in Figure 6:

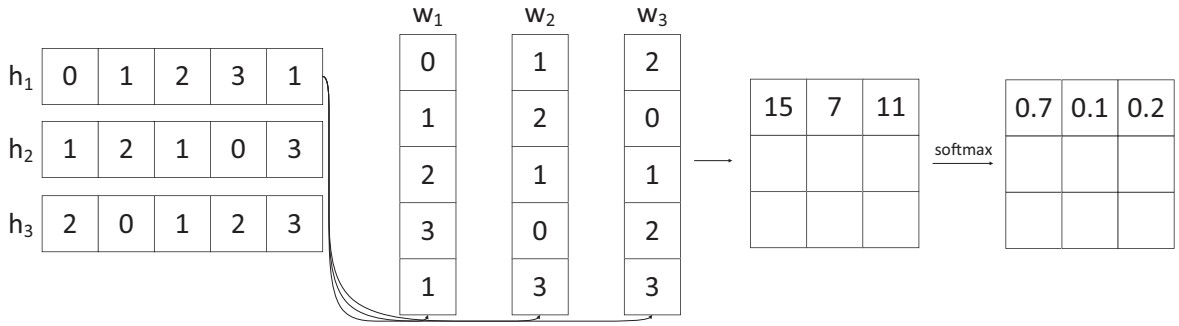

Figure 6. Normalization.

3. Weighted summation, as shown in Figure 7:

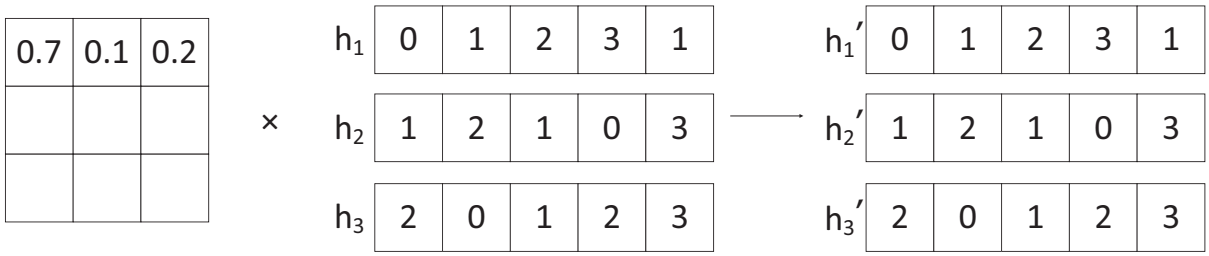

Figure 7. Weighted summation.

The final result is:

$$H^{'} = softmax(\frac{HW^{T}}{\sqrt{d_w}})H \tag{7}$$

H is the matrix corresponding to the hidden vector and W is the matrix corresponding to the word vector. To prevent the gradient from disappearing, divide the inner product by the square root of the dimensions.

The $h^{'}$ vector after assigning weights is used for the representation of sentence vectors, meaning that to provide high-quality summaries, the decoder pays attention to both the source text's keywords and the sentences in which they appear. The word-embedding averaging method by Kedzie et al. [30] works well for sentence embedding across different domains and summarizer architectures. We obtain the sentence embedding, i.e., $S_m^{'}$, by this simple method. As shown in Equation (8).

$$S_m^{'} = \frac{1}{n}\sum_{t=1}^{n} h_t^{'} \tag{8}$$

3.3. Joint-Attention Mechanism for Word-Sentence-Level Fusion

In this paper, each word is used as input to the encoder, and after GRU training, the relationship between word-level vectors is obtained. Distinguish from the relationship between sentence-level vectors mentioned later. GRU suffers from the long-range dependency problem, i.e., when the input is too long, the later input words will forget the previously input words, which in turn causes incomplete information between words, resulting in incomplete features received by the decoder. In this paper, each sentence in the paragraph is subjected to sentence embedding processing, as follows: The vector obtained by summing and averaging the word-embedding of all the words in the sentence is used as the final sentence embedding. After this processing, there will be relationships not only between word-level vectors, but also between sentence-level vectors, and the decoder discriminates

the word-level sentence-level vectors based on their relationship with the hidden state of the decoder. The specific formula is as follows.

$$d_j^w = GRU(y_{j-1}, d_{j-1}^w, c_j^w) \tag{9}$$

$$e_{jn} = V_a^T tanh(W_a d_{j-1}^w + U_a h_n^{'}) \tag{10}$$

$$\alpha_{jn} = \frac{exp(e_{jn})}{\sum_{k=1}^{n} exp(e_{jk})} \tag{11}$$

$$c_j^w = \sum_{p=1}^{n} \alpha_{jn} h_p^{'} \tag{12}$$

$$d_j^s = GRU(y_{j-1}, d_{j-1}^s, c_j^s) \tag{13}$$

$$e_{jm} = V_a^T tanh(W_a d_{j-1}^s + U_a S_m) \tag{14}$$

$$\alpha_{jm} = \frac{exp(e_{jm})}{\sum_{k=1}^{m} exp(e_{jk})} \tag{15}$$

$$c_j^s = \sum_{q=1}^{m} \alpha_{jm} S_q \tag{16}$$

Equations (9)–(12) is the process of obtaining word-level context vectors c_j^w, and Equations (13)–(16) is the process of obtaining sentence-level context vectors c_j^s.

Taking the word-level context vector c_j^w calculation process as an example: First, the decoder generates a hidden vector d_j^w based on the hidden vector of the previous neuron d_{j-1}^w, the output of the previous neuron y_{j-1}, and the context vector of the current neuron c_j^w. Second, the j^{th} neuron of the decoder calculates the correlation e_{jn} between the current state and each neuron in the encoder $h_n^{'}$ based on the hidden vector of the previous neuron d_{j-1}^w. Third, obtain the generated word weight score α_{jn}; Finally, the word-level context vector c_j^w is obtained. Similarly, the sentence-level context vector c_j^s is obtained.

Where V_a, U_a are the parameter vectors. n means that there are n words in the paragraph and m means that there are m sentences in the paragraph (i.e., assuming that each input text has m sentences). d_j is the hidden vector generated by the decoder based on the hidden vector d_{j-1} of the previous neuron, the output y_{j-1} of the previous neuron and the context vector c_j of the current neuron. d_j^w is the hidden vector in word-level dimension and d_j^s is the hidden vector in sentence-level dimension. h_n is the output of each neuron of the encoder, and S_m is the sentence-level representation computed from each word-level output. The jth neuron of the decoder has to calculate the correlation e_{jn} of the current state with each neuron in the encoder based on the hidden vector d_{j-1} of the previous neuron. The sentence-level relevance vector is denoted as e_{jm}. If the n^{th} dimension of e_{jn} is larger, it means that the current node is more correlated with the nth neuron of the encoder. α_{jn} is the generated word weight score and α_{jm} is the generated sentence weight score. As a result, the word-level context vector c_j^w and the sentence-level context vector c_j^s are obtained.

Integrating the above content of the article, the semantic vectors (word-level semantic vectors h_1 to h_n) output from the source text input encoder are obtained after the weighted summation process (word-level semantic vectors $h_1^{'}$ to $h_n^{'}$), and the correlation is calculated with the hidden state of the decoder at each moment to obtain the generated word weight score first and then the word-level context vector c_j^w on the one hand, and the sentence embedding is obtained by the averaging method on the other hand. The word-embedding

vectors (w_1 to w_n) obtained by word2vec enter the encoder training to obtain the output semantic vectors (h_1 to h_n) on the one hand, and obtain the a priori knowledge by averaging on the other hand.

3.4. Decoding Side

3.4.1. Improvements to the Pointer Network

Since the summaries may contain words that do not exist in the word list, we use a pointer network between the encoder and decoder. The pointer generator network helps to copy words from the source text via pointers, which improves the accuracy and processing of out-of-vocabulary (OOV) words, while retaining the ability to generate new words. The pointer network is equivalent to a soft switch, and P_{gen} serves as the probability of the soft switch, by which P_{gen} determines whether the summary is copied from the original text or generated from a fixed vocabulary, and P_{gen} is expressed as follows:

$$P_{gen} = \sigma(w_c^T c_i + w_s^T S_t + w_y^T y_i + b_{ptr}) \tag{17}$$

In this paper, we improve the pointer network with a joint-attention mechanism for the problem of unregistered words and duplication of generated summaries, as follows:

$$P_{gen} = \sigma(w_w^T c_w^j + w_s^T c_s^j + w_S^T d_t + w_y^T y_j b_{ptr}) \tag{18}$$

Among them, w_c, w_w, w_s, w_S, w_y, and b_ptr are learnable parameters, c_j is the context vector, S_t is the decoder hidden state, y_j is the output of the decoder, σ is the sigmoid function, and $P_{gen} \subseteq [0, 1]$.

Humans tend to combine background knowledge when reading the central content of overview articles, and the model-generated summaries often do not match human reading habits. We combine a priori knowledge into the pointer network to guide the coverage mechanism of the pointer network by compressing the source text semantic information through sentence embedding to assist the summary generation. A priori knowledge is that which can be possessed not through acquired experience. In the abstract, we interpret it as the original vector without any training. Therefore, we use the initial word vector obtained by word2vec to represent the sentence vector as a basis for a priori knowledge.

We still use word-embedding averaging to obtain the sentence embedding after compressing the source text, i.e., S_m. As shown in Equation (19), S_m represents a higher-level semantic representation.

$$S_m = \frac{1}{n} \sum_{t=1}^{n} w_t \tag{19}$$

After obtaining the sentence embedding, the weight fraction of each hidden state of the sentence embedding and decoder is found by the same method as calculating α_{jm}, as independent of the empirical knowledge. That is, the matching relationship between the words to be output and the individual sentences of the source text is obtained, and then the model prefers the words in the original sentence when decoding the output.

$$P(w) = P_{gen} P_{vocab}(w) + (1 - P_{gen})(\sum \alpha_i^t + \sum \alpha_i^t S_{argmax}) \tag{20}$$

where S_{argmax} is the highest probability corresponding to the sentence embedding. α_i^t has two meanings, α_{jn}^t and α_{jm}^t, respectively. In this way, the model generates summaries with the flexibility to consider not only word-level sentence-level vectors to convey inter-word relationships, but also to preserve the guidance of the source text to the summary through a priori knowledge. $P_{vocab}(w)$ is a pre-constructed word list that does not contain the OOV words in the input text. The improved pointer network is shown in Figure 8.

$$P_{vocab}(w) = softmax(V^{'}(V[d_t^w, h_t] + b) + b^{'}) \tag{21}$$

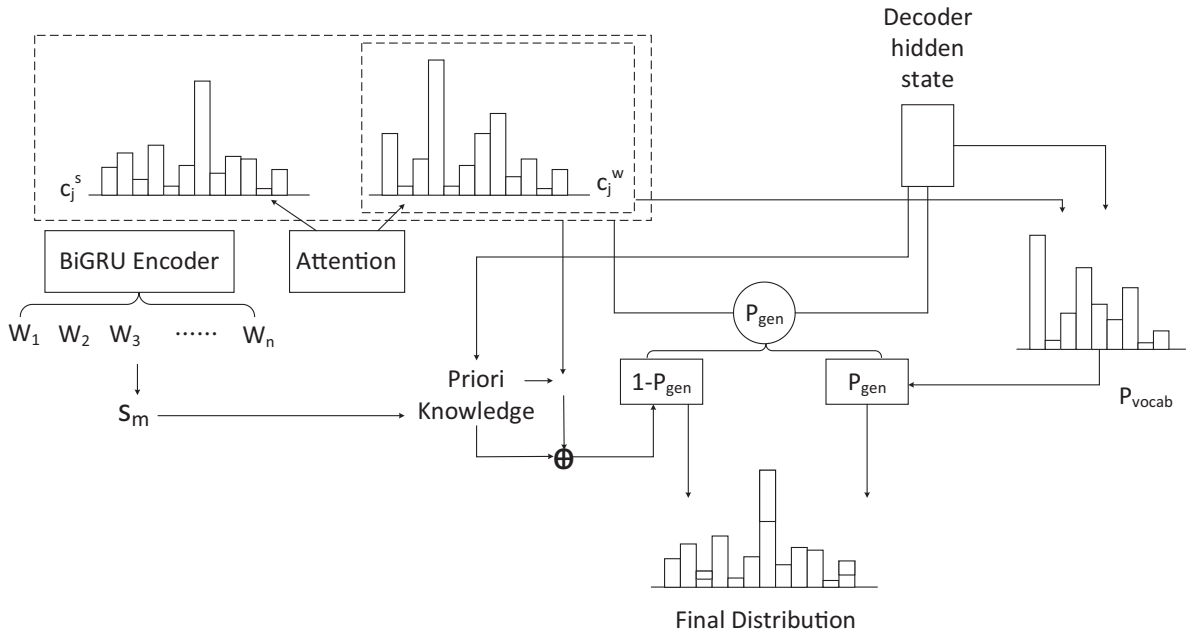

Figure 8. Improved pointer network.

In our experiments, we found that the summaries generated by the model have recurring problems, such as "I like you like you like you like you like you". The coverage mechanism can add the influence of previous decisions to the current decision, avoiding the attention mechanism from continuously focusing on the same position and thus avoiding the generation of duplicate texts. The specific formula is as follows.

$$c^t = \sum_{t'=0}^{t-1} \alpha_{jn}^{t'} \tag{22}$$

c^t is the distribution over the original words, indicating the coverage these words have received from the attention mechanism so far. Notice that c^0 is a zero vector because no words are covered in the original text at the initial moment. Since it is often the words that present the duplication problem, we only use the previous decisions of the word-level attention mechanism for the coverage vector.

The coverage vector acts as an additional input to the attention mechanism, and the formula is:

$$e_{jn}^t = V_T^a tanh(W_a d_{j-1}^w + W_c c_j^t + U_a h_n + b_{attn}) \tag{23}$$

W_a, W_c, U_a, and V_a are learnable vectors.

3.4.2. Reinforcement Learning Based on Improved Pointer Networks

During the training process of the codec, the decoder part inputs the correct reference summary, while during the prediction process, the encoder inputs the output of the previous time step, and once the output of the previous step is wrong, it will cause a series of subsequent deviations, leading to the problem of poor summary quality. This is the exposure bias problem. In addition, the model-generated summaries usually suffer from incoherence and semantic irrelevance, and such problems are caused by the non-uniformity of evaluation metrics and loss functions. For the above problems, the solution in this article is as follows.

Reinforcement learning is used to describe and solve the problem of learning strategies to maximize the reward or achieve a specific goal by an intelligence agent during its interaction with the environment. The training method incorporating the reinforcement learning approach considers the supervised learning objective (maximum-likelihood method) and

the RL strategy gradient objective for generating relationships between summary sentences and standard summary sentences under the Rouge criterion. The maximum-likelihood loss is the loss function used in the conventional supervised learning strategy; training by decreasing the loss function is comparable to increasing the likelihood that the real value will occur. According to the reference summary $y^*\{y_1^*, y_2^*, ..., y_n^*\}$, minimize the negative logarithmic likelihood function L_{ML}.

$$L_{ML} = -\sum_{t=1}^{n} logP(y_t^*|y_1^*, y_2^*, ..., y_{t-1}^*, w) \tag{24}$$

The maximum likelihood only allows the generative model to learn similar representation patterns to the standard summary, while the Rouge metric allows for flexibility in sentence alignment. Therefore, we modify the loss function so that the loss function can be optimized for Rouge. Specifically, two independent output sequences y^s and $\widehat{y}$ are generated in each training iteration. y^s is obtained by sampling from the $P(y_t^s|y_1^s, y_2^s, ..., y_{t-1}^s, w)$conditional probability distribution at each decoding time step, and $\widehat{y}$ is obtained by maximizing the output probability distribution, which is also the model output result. We use the Rouge score as the reward function $r(y)$. The generated abstracts are compared with the reference abstracts, compared by Rouge function values, and rewards are calculated. $r(y)$ is calculated as the similarity evaluation Rouge score between the generated abstract y and the reference abstract y^*.

$$L_{RL} = (r(\widehat{y}) - r(y^s)) \sum_{t=1}^{n} logP(y_t^*|y_1^*, y_2^*, ..., y_{t-1}^*, w) \tag{25}$$

Minimizing L_{RL} is equivalent to maximizing the conditional likelihood of the sampled sequence y^s, thus increasing the payoff expectation of the model. We finally mix the loss functions of the two components to obtain the overall objective function, with the following equation.

$$L_{mixed} = \lambda L_{RL} + (1 - \lambda) L_{ML} \tag{26}$$

λ is used to balance L_{RL} and L_{ML}. The loss function is introduced through reinforcement learning to effectively alleviate the exposure bias problem on the one hand, and to optimize the evaluation metrics and improve the evaluation metric scores of the model on the other hand. At this point, we have presented all parts of the generative summary model based on the joint-attention mechanism and a priori knowledge. Next, the experimental part is elaborated.

4. Experiment

To investigate the performance of the model in this paper, we evaluate our model using two benchmark datasets CNN/DailyMail [31] and DUC 2004 [32] for single-text generative summarization. In addition, we compare the model in this paper with the publicly available model for analysis.

4.1. Dataset

The CNN/DailyMail dataset began as a machine reading comprehension corpus with about 1 million news data collected from CNN and the DailyMail. Later, simple changes were made to form a corpus for single-text generative abstracts. The dataset includes 286,817 training samples, 13,368 verification samples and 11,487 test samples. The training set has an average of 3–4 summary sentences, and each document has an average of 28 sentences. In addition, the average number of tags in the input and output abstracts is 781 and 56, respectively. This dataset is used to train and test the proposed model.

Meanwhile, this paper uses DUC 2004 dataset as the test set to further evaluate our model. DUC contains the 2001–2007 corpus, where the DUC2001–2004 corpus is suitable

for both single-text summarization and multi-text summarization, and DUC2005–2007 is only suitable for multi-text summarization. This article proposes a model for single-text summaries, and therefore uses the DUC 2004 dataset. DUC 2004 is a small dataset for testing only. DUC 2004 contains 500 documents, and each news article contains a reference summary of the corresponding four different manually generated intercepts of 75B.

4.2. Evaluation Metrics

To demonstrate the model's viability and efficacy after receiving the summary it provided, we must assess the model's performance. Both manual and automatic evaluations were used to gauge the abstracts' quality.

Automatic evaluation methods for text summarization include Rouge [33], BLEU [34], etc., and the commonly used one is Rouge, which is also used as an evaluation metric in this paper. Rouge measures the quality of the model-generated summaries by calculating the degree of overlap of the basic units (n-grams) between the generated summaries and the reference summaries. Rouge typically contains the following metrics.

1. Rouge-N: Statistics based on the co-occurrence degree of N-gram words.

$$Rouge-1 = \frac{\sum_{S \in Y} \sum_{1-gram \in S} Count_{match}(1-gram)}{\sum_{S \in Y} \sum_{1-gram \in S} Count(1-gram)} \tag{27}$$

$$Rouge-2 = \frac{\sum_{S \in Y} \sum_{2-gram \in S} Count_{match}(2-gram)}{\sum_{S \in Y} \sum_{2-gram \in S} Count(2-gram)} \tag{28}$$

   where Y is the reference summary and S is the sentence in the reference summary. The numerator indicates the number of simultaneous $N-gram$ in the generated summary and the standard summary. The denominator indicates the number of $N-gram$ occurrences in the reference summary.
2. Rouge-L: refers to the longest common subsequence (LCS) in the generated abstracts and reference abstracts.

$$Rouge-L = \frac{(1+\beta^2)R_{LCS}P_{LCS}}{R_{LCS}+\beta^2 P_{LCS}} \tag{29}$$

$$R_{LCS} = \frac{LCS(X,Y)}{n} \tag{30}$$

$$P_{LCS} = \frac{LCS(X,Y)}{m} \tag{31}$$

$$\frac{1}{Rouge-L} = \frac{1}{(1+\beta^2)P_{LCS}} + \frac{\beta^2}{(1+\beta^2)R_{LCS}} \tag{32}$$

   where R_{LCS} denotes the recall rate in Equation (30), P_{LCS} denotes the precision rate in Equation (31), $LCS(X,Y)$ denotes the length of the longest common subsequence between the generated summary and the reference summary, n is the length of the reference summary, and m is the length of the generated summary. In general, β is set to a large value, so $Rouge-L$ usually considers only R_{LCS} (recall) in this case. As shown in Equation (32), if β is large, the inclusion of the first half of Equation P_{LCS} can be neglected.

When performing a manual evaluation, we primarily look at readability and whether or not the text retains important details from the original (generalization). We manually select 100 examples from the CNN/DailyMail dataset and give the generated summaries and reference summaries generated by the different models to the evaluators. The five evaluators scored the abstracts of the different models based on the above two aspects, from 1 to 5, with higher scores representing the more the resulting abstracts met the two

evaluation metrics. Finally, the average value is taken as the final score of the manual evaluation of the model.

The model is evaluated comprehensively using both automatic and manual evaluations. to ensure the quality of the model-generated abstracts.

4.3. Comparison with Public Models

4.3.1. Results on the Dataset CNN/DailyMail

On the CNN/DailyMail dataset, we compare the model with the following publicly available models.

- TextRank [35]: a graph-based ranking algorithm for keyword extraction and document summarization, improved from the PageRank algorithm. It extracts keywords using co-occurrence information (semantics) between words within a document, and can extract keywords, key phrases of a given text from that text, and key sentences of that text using an extractive automatic digest method.
- Seq2seq + attention [19]: a standard sequence-to-sequence model with an attention mechanism for generating summaries.
- PGN [36]: to solve the problem of words that do not appear in the word list, a pointer generation network is proposed.
- PGN + Cov [36]: an overlay mechanism is added to the pointer generation network to avoid duplication of the generated summaries.
- FASum + FC [37]: a knowledge graph is used to represent the factual information extracted from the article and a factual corrector model is used to correct factual errors.
- ML + RL, intra-attention [18]: introduces a new internal attention mechanism that combines the standard supervised word prediction and reinforcement learning training methods to reduce bias.
- ML + RL Rouge + Novel, with LM [29]: the decoder is divided into a contextual network and a pre-trained language model, and reinforcement learning is used to encourage the generation of new phrases.
- Lightweight Meta-Learning [38]: a low-resource generative summary model using meta-learning and lightweight modules. Effectively mitigates domain transfer and overfitting problems.
- DEATS [39]: proposes a dual-coding model that mutually benefits from the basic model and achieves state-of-the-art results compared to existing methods.
- BERTSumABS [40]: the pre-trained BERTSum is used as the encoder and the 6-layer Transformer [20] is used as the decoder. In addition, a new fine-tuning scheme is proposed to separate the optimizers of the codecs.
- TransformerS2S + CS + RL [41]: an FCSF-TABS-based model is proposed, where Bert is fine-tuned for content selection in the first stage and fed into the Transformer-based summary model in the second stage to generate summary sentences.

Table 1 shows the results of comparing the model in this paper with existing models. It can be seen that JPATS outperforms most publicly available models. Compared with ML + RL, intra-attention and ML + RL Rouge + Novel, with LM, which introduced reinforcement learning, JPATS improved in all three Rouge metrics, which we attribute to the joint-attention mechanism and a priori knowledge, and we will further demonstrate our idea with ablation experiments later. It is worth noting that JPATS is slightly less effective than the models BERTSumABS, TransformerS2S + CS + RL, which incorporate Bert and Transformer, reflecting the significant effect of the pre-trained models. We will further focus on the pre-trained model and migrate the methods in this paper to the pre-trained model to obtain better results.

Table 2 shows the results of the manual evaluation of each model. Our model achieved the highest scores on both readability and general. The presence of a priori knowledge in the JPATS model is shown to help the decoder generate summaries that conform to human reading habits and improve the readability of the summaries. In addition, the model not only requires the decoder to focus on the keywords in the source text, but also forces the

decoder to focus on the sentences where the keywords are located, and the combination of the two drives the model to generate more comprehensive and specific summaries. Thus, our model is also superior in the two manual evaluation metrics of readability and general. Combining Rouge evaluation metrics and manual evaluation, our model achieves good results.

Table 1. Rouge scores of JPATS on the dataset CNN/DailyMail.

Models	Rouge-1	Rouge-2	Rouge-L
TextRank	34.11	12.78	22.5
Seq2seq + attention	31.34	11.79	28.10
PGN	36.44	15.66	33.42
PGN	36.44	15.66	33.42
PGN + Cov	39.53	17.28	36.38
FASum + FC	40.38	17.67	37.23
ML + RL, intra-attention	39.87	15.82	36.9
ML + RL Rouge + Novel, with LM	40.19	17.38	37.52
Lightweight Meta-Learning	39.94	16.96	26.09
DEATS	40.85	18.08	37.13
BERTSumABS	41.72	19.39	38.76
TransformerS2S + CS + RL	42.71	19.94	39.31
JPATS	41.25	18.81	38.34

Table 2. Manual scoring of JPATS on the dataset CNN/DailyMail.

Models	Readability	General
Seq2seq + attention	3.52	3.44
PGN + Cov	3.66	3.39
FASum + FC	3.70	3.69
DEATS	3.73	3.71
ML + RL, intra-attention	3.79	3.76
ML + RL Rouge + Novel, with LM	3.85	3.80
BERTSumABS	3.97	3.92
TransformerS2S + CS + RL	4.03	4.03
JPATS	4.12	4.09

4.3.2. Results on the Dataset DUC 2004

On the DUC 2004 dataset, we compared the model with the following publicly available models.

- ABS [14]: a CNN encoder and an NNLM decoder are used to accomplish the summarization task.
- ABS + [14]: an enhanced version of the ABS model, which relies on a series of individually extracted summary features that are added as log-linear features in the secondary learning step.
- RAS-Elman [15]: using an attentive encoder and RNN-based decoder.
- SEASS [12]: a selection mechanism is used to control the information input to the decoder to remove redundant information.
- DEATS [39]: a dual-coding model is proposed that mutually benefits from the basic model and achieves state-of-the-art results compared to existing methods.

Table 3 shows the results of comparing the model in this paper with existing models. As shown in the table, our models all outperformed all baseline models. It reached 29.98, 9.80, and 26.12 on Rouge-1, Rouge-2, and Rouge-L, respectively. Experiments on this dataset further demonstrate the performance of the model in this paper.

Table 3. Rouge scores of JPATS on dataset DUC 2004.

Models	Rouge-1	Rouge-2	Rouge-L
ABS	26.55	7.06	22.05
ABS+	28.18	8.49	23.81
RAS-Elman	28.97	8.26	24.06
SEASS	29.21	9.56	25.51
DEATS	29.91	9.61	25.95
JPATS	29.98	9.80	26.12

5. Discussion

To examine the function of each module in the overall model, we ran corresponding ablation experiments on the model. One or two modules are removed based on the experiment. After the WS, PK, JAM, and RL modules are removed, the fundamental model is JPATS _w/o_ALL. The PGN + Cov model in Table 1 is now identical to JPATS _w/o_ALL.

The WS module of this model is intended to determine whether the hidden vectors generated by Bi-GRU deviate from the meaning of the original text and can assign higher weights to the important parts of the original text. In Table 4, we obtain the scores of the model without the WS module, and the results show that the scores are slightly worse than the original JPATS model. Again, the model with the JAM, PK, and RL modules removed scored slightly higher compared to the model with all modules removed. After comparing the two data, it is proved that the WS module plays the role of judging the matching and assigning weights in the model.

Table 4. JPATS ablation experiment scores on dataset CNN/DailyMail.

Models	Rouge-1	Rouge-2	Rouge-L
JPATS	41.25	18.81	38.34
JPATS_w/o_ WS	40.97	18.65	38.09
JPATS_w/o_ JAM	40.37	18.53	37.95
JPATS_w/o_ PK	40.64	18.47	37.96
JPATS_w/o_ RL	39.66	17.35	36.41
JPATS_w/o_ JAM + PK	40.19	18.21	37.93
JPATS_w/o_ JAM + PK + RL	39.86	17.79	37.88
JPATS_w/o_ ALL	39.53	17.28	36.38

The JAM module uses a joint-attention mechanism that is a fusion of the word-level attention mechanism and sentence-level attention mechanism. The joint-attention mechanism is computed separately with the state of the decoder at each moment, so that the model pays attention to both the keyword and the sentence in which the keyword is located. The captured word-level features and sentence-level features help the decoder to improve the quality of the generated summaries. In the absence of the JAM module, the Rouge score decreases by 0.88, 0.28, and 0.39, respectively. This shows that the JAM module helps the model to consider richer information and improve model performance. In addition, the Rouge score decreases more compared to the score after removing the WS module, indicating a more pronounced effect of the JAM module.

The PK module mimics the human habit of combining background knowledge when reading, and uses the original semantic information without any training to give the model more original information, prompting the generation of summaries that are more in line with human reading habits. After removing the PK model, the Rouge score of the model showed a decrease. The scores still decreased compared to the model with the WS module removed, indicating that the PK module outperformed the WS module for the summary generation task.

Based on previous studies on reinforcement learning, it is effective in improving Rouge scores. For this experiment, we also remove the RL module separately to study it. Removing the RL module resulted in a larger decrease in model scores than the model

with the remaining modules removed, and we obtained the same conclusion as in previous studies. The effectiveness of the RL module for improving Rouge scores can also be derived from the scores of the model with JAM + PK removed and the model with JAM + PK + RL removed.

6. Conclusions

In this paper, we propose an abstractive summarization model based on the joint-attention mechanism and a priori knowledge. On the encoding side, we incorporate a weighted summation part to determine how well the encoder produces vectors that match the original text on the one hand, and assign higher weights to important parts on the other. On the decoding side, we first improve the pointer network using the word–sentence-level joint-attention mechanism to make the decoder focus on the keywords and the sentences where the keywords are located. The original text vector without any training is then used as a priori knowledge to further improve the pointer network to generate a summary that matches human reading habits. Finally, reinforcement learning is introduced to alleviate the exposure bias problem and optimize the evaluation metrics. After experimenting on two publicly available datasets, we find that the model in this paper shows good results compared with existing publicly available models.

Although the model in this paper achieved good results, some other problems were found in the text summary exploration phase: (1) Rouge, a commonly used evaluation metric in the field of text summarization, itself judges the degree of matching between the generated summary and the reference summary, ignoring the evaluation of coherence, conciseness, etc. The introduction of manual evaluation can be time-consuming and labor-intensive for large amounts of data. Therefore, effective improvement of evaluation indicators is crucial for both Chinese and English abstract studies. (2) The uninterpretability of deep learning also hinders the development of the text summarization field to some extent. (3) Whether the introduction of pre-trained models will further improve the model quality. In the future, we will further explore research from the above three points.

Author Contributions: Writing—original draft preparation, Y.L.; writing—review and editing, Y.H.; visualization, J.Y.; supervision, W.H.; project administration, Z.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Ji, X.; Zhao, W. SKGSUM: Abstractive Document Summarization with Semantic Knowledge Graphs. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021. [CrossRef]
2. Li, W.; Zhuge, H. Abstractive Multi-Document Summarization Based on Semantic Link Network. *IEEE Trans. Knowl. Data Eng.* **2021**, *33*, 43–54. [CrossRef]
3. Zhu, T.; Hua, W.; Qu, J.; Hosseini, S.; Zhou, X. Auto-regressive extractive summarization with replacement. *World Wide Web* **2022**. [CrossRef]
4. Li, W.; Xiao, X.; Liu, J.; Wu, H.; Du, J. Leveraging Graph to Improve Abstractive Multi-Document Summarization. *arXiv* **2020**, arXiv:2005.10043.
5. Banerjee, S.; Mitra, P.; Sugiyama, K. Multi-Document Abstractive Summarization Using ILP Based Multi-Sentence Compression. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI), Buenos Aires, Argentina, 25–31 July 2015; pp. 1208–1214.
6. Durrett, G.; Berg-Kirkpatrick, T.; Klein, D. Learning-Based Single-Document Summarization with Compression and Anaphoricity Constraints. *arXiv* **2016**, arXiv:1603.08887.

7. Dohare, S.; Gupta, V.; Karnick, H. Unsupervised Semantic Abstractive Summarization. In Proceedings of the Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018.
8. Wei, L. Abstractive Multi-document Summarization with Semantic Information Extraction. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015.
9. Cao, Z.; Li, W.; Wei, F.; Li, S. Retrieve, Rerank and Rewrite: Soft Template Based Neural Summarization. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), Melbourne, Australia, 15–20 July 2018; pp. 152–161.
10. Makino, T.; Iwakura, T.; Takamura, H.; Okumura, M. Global Optimization under Length Constraint for Neural Text Summarization. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019), Florence, Italy, 28 July–2 August 2019; pp. 1039–1048.
11. Luhn, H.P. The automatic creation of literature abstracts. *IBM J. Res. Dev.* **1958**, *2*, 159–165. [CrossRef]
12. Zhou, Q.; Yang, N.; Wei, F.; Zhou, M. Selective Encoding for Abstractive Sentence Summarization. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017), Vancouver, BC, Canada, 30 July–4 August 2017; pp. 1095–1104. [CrossRef]
13. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems 27, Montreal, QC, Canada, 8–13 December 2014; Volume 2.
14. Rush, A.M.; Chopra, S.; Weston, J. A Neural Attention Model for Abstractive Sentence Summarization. *Comput. Sci.* **2015**. [CrossRef]
15. Chopra, S.; Auli, M.; Rush, A.M. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016.
16. Nallapati, R.; Zhou, B.; Santos, C.; Gulcehre, C.; Bing, X. Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. In Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, Berlin, Germany, 11–12 August 2016.
17. Gehring, J.; Auil, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. Convolutional Sequence to Sequence Learning. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 70.
18. Paulus, R.; Xiong, C.; Socher, R. A Deep Reinforced Model for Abstractive Summarization. *arXiv* **2017**, arXiv:1705.04304.
19. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *Comput. Sci.* **2014**. [CrossRef]
20. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; Volume 30.
21. Zheng, X.; Sun, A.; Muthuswamy, K. Tweet-aware News Summarization with Dual-Attention Mechanism. In Proceedings of the Web Conference 2021: Companion of the World Wide Web Conference, New York, NY, USA, 12–16 April 2021; pp. 473–480.
22. Hakami, N.A.; Mahmoud, H.A.H. A Dual Attention Encoder-Decoder Text Summarization Model. *CMC Comput. Mater. Contin.* **2023**, *74*, 3697–3710. [CrossRef]
23. Qiu, D.; Yang, B. Text summarization based on multi-head self-attention mechanism and pointer network. *Complex Intell. Syst.* **2022**, *8*, 555–567. [CrossRef]
24. Liu, S.; Yang, L.; Cai, X. SEASum: Syntax-Enriched Abstractive Summarization. *Expert Syst. Appl.* **2022**, *199*, 116819. [CrossRef]
25. Narayan, S.; Cohen, S.B.; Lapata, M. Ranking Sentences for Extractive Summarization with Reinforcement Learning. *arXiv* **2018**, arXiv:1802.08636
26. 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 (ACL), Melbourne, Australia, 15–20 July 2018; pp. 675–686.
27. Liang, Z.; Du, J.; Li, C. Abstractive social media text summarization using selective reinforced Seq2Seq attention model. *Neurocomputing* **2020**, *410*, 432–440. [CrossRef]
28. Liao, W.; Ma, Y.; Yin, Y.; Ye, G.; Zuo, D. Improving abstractive summarization based on dynamic residual network with reinforce dependency. *Neurocomputing* **2021**, *448*, 228–237. [CrossRef]
29. Kryscinski, W.; Paulus, R.; Xiong, C.; Socher, R. Improving Abstraction in Text Summarization. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2018), Brussels, Belgium, 31 October–4 November 2018; pp. 1808–1817.
30. Kedzie, C.; Mckeown, K.; Hal, D., III. Content Selection in Deep Learning Models of Summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018.
31. Hermann, K.M.; Koisk, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; Blunsom, P. *Teaching Machines to Read and Comprehend*; MIT Press: Cambridge, MA, USA, 2015.
32. Over, P.; Dang, H.; Harman, D. DUC in context. *Inf. Process. Manag.* **2007**, *43*, 1506–1520. [CrossRef]
33. Lin, C.Y. ROUGE: A Package for Automatic Evaluation of summaries. In Proceedings of the In Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004), Barcelona, Spain, 25–26 July 2004.

34. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W. BLEU: A method for automatic evaluation of machine translation. In Proceedings of the 40TH Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, USA, 7–12 July 2002; pp. 311–318. [CrossRef]
35. Mihalcea, R.; Tarau, P. TextRank: Bringing Order into Texts. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 25–26 July 2004.
36. See, A.; Liu, P.J.; Manning, C.D. Get To The Point: Summarization with Pointer-Generator Networks. *arXiv* **2017**, arXiv:1704.04368.
37. Zhu, C.; Hinthorn, W.; Xu, R.; Zeng, Q.; Zeng, M.; Huang, X.; Jiang, M. Enhancing Factual Consistency of Abstractive Summarization. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021.
38. Huh, T.; Ko, Y. Lightweight Meta-Learning for Low-Resource Abstractive Summarization. In Proceedings of the 45th International ACM Sigir Conference on Research and Development in Information Retrieval, Madrid, Spain, 11–15 July 2022; pp. 2629–2633. [CrossRef]
39. Yao, K.; Zhang, L.; Du, D.; Luo, T.; Tao, L.; Wu, Y. Dual Encoding for Abstractive Text Summarization. *IEEE Trans. Cybern.* **2020**, *50*, 985–996. [CrossRef] [PubMed]
40. Liu, Y.; Lapata, M. Text Summarization with Pretrained Encoders. *arXiv* **2019**, arXiv:1908.08345.
41. Zhang, M.; Zhou, G.; Yu, W.; Liu, W.; Huang, N.; Yu, Z. FCSF-TABS: Two-stage abstractive summarization with fact-aware reinforced content selection and fusion. *Neural Comput. Appl.* **2022**, *34*, 10547–10560. [CrossRef]