Advances in Computer, Signals and Systems (2024)
Clausius Scientific Press, Canada

DOI: 10.23977/acss.2024.080603
ISSN 2371-8838 Vol. 8 Num. 6

Research on Graph-based Text Summarization Extraction Algorithm

Junhong Chen[1,2,a,*], Kaihui Peng[3,b]

[1]*School of Software Engineering, South China University of Technology, Guangzhou, China*
[2]*LeiHuo Studio, NetEase, Hangzhou, China*
[3]*Faculty of Business and Economics, University of Malaya, Kuala Lumpur, Malaysia*
[a]*jupyterchen@163.com*, [b]*pengkaihui66@163.com*
[*]*Corresponding author*

Keywords: Text Summarization; Keyword Extraction; Pre-trained Model

Abstract: This paper proposes a graph-based text summarization extraction algorithm. The algorithm is based on directed graphs and can incorporate the position information of sentences into the computational scope. When calculating the edge weights of nodes in the directed graph, a pre-trained model after negative sampling is used, which not only can extract deeper semantic features but also enable higher relevance between the contextual sentences in the article. The algorithm also introduces a weighting mechanism to adjust the extraction priority of the sentences according to the article's theme, resulting in a higher quality of extracted summary sentences that can represent the key information of the text as much as possible. The algorithm can capture the key information in the text, reduce the impact of irrelevant information on semantics, and play a role in text compression.

1. Introduction

This paper will propose a graph-based text summarization extraction algorithm. Unlike the TextRank[1] based on undirected graphs, this algorithm is based on directed graphs and can take the position information of sentences into consideration. When calculating the edge weights, a pre-trained model is used to extract deep semantic features and generate dynamic word vectors. In addition, for better Chinese support, the text uses the full-word masked version of BERT[2], known as Bert-wwm[3], with the pre-training dataset being the Chinese Wikipedia. In the text, to further improve the quality of semantic extraction, negative sampling is used to pre-train Bert-wwm, thereby enhancing the relevance of the contextual sentences in the article. This algorithm also refers to the article's theme and introduces a weighting mechanism to adjust the priority of sentence output to some extent. The extracted summary text of this algorithm has higher quality, richer information, and better represents the meaning of the text.

2. Design of Graph-based Text Summarization Extraction Algorithm

2.1 Algorithm Overview

The graph-based text summarization extraction algorithm proposed in this paper, named EMW-Sum, primarily consists of two major components: Firstly, the first part involves the pre-training of Bert-wwm, during which negative sampling methods are adopted to obtain the pre-trained Bert-wwm model, designated as N-Bert-wwm. Following this, the N-Bert-wwm model is utilized to generate sentence representation vectors. These vectors are then employed to compute the similarity between sentences, which in turn is used to construct a similarity matrix between the sentences[4]. This part generates a text similarity matrix that serves as the foundation for the subsequent calculations. Next, the second part uses the directed graph based on the sentence similarity matrix obtained from the previous step to compute the scores of the sentences. During the scoring process, certain methods are also applied to optimize the final scores of the sentences. Afterward, sentences are sorted according to their final scores, and a certain number of sentences are selected until the set length requirement is met[5]. Lastly, when generating the summary set, sentences are extracted in the order they appear in the original text. Figure 1 illustrates the overall structural diagram of the EMW-Sum algorithm.

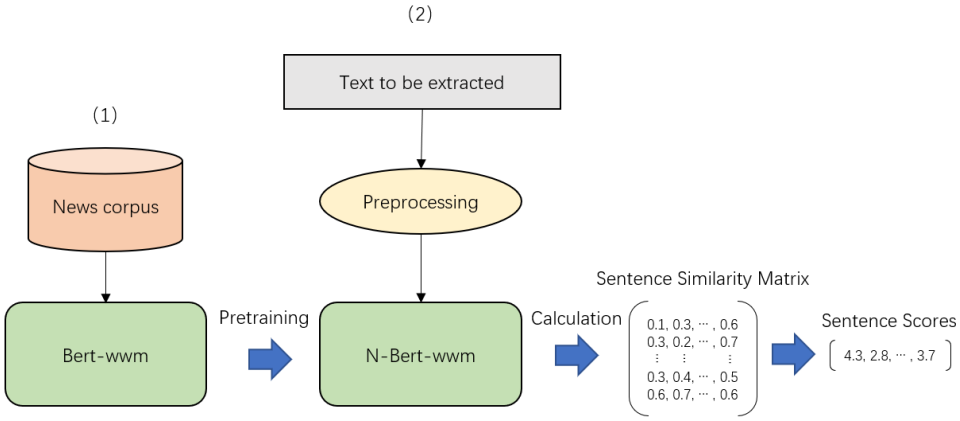

Figure 1: EMW-Sum Overall Structure Diagram

2.2 Input Layer

Whether it is the pre-training of Bert-wwm or the use of Bert-wwm to extract sentence semantics, the sentences must be input into the Bert-wwm. This section will introduce the input layer of Bert-wwm.

The input of Bert-wwm is consistent with that of BERT, which requires that sentences be segmented at the character level, inserting [CLS] marks at the beginning of the sentence, and [SEP] marks at the end, and extending them to a fixed length. Let $S_a = \{x_{a_1}, x_{a_2}, \cdots, x_{a_n}\}$, $S_b = \{x_{b_1}, x_{b_2}, \cdots, x_{b_n}\}$, represent the original input sentences, where n is the length of the sentence. S_a, S_b need to be added [CLS] and [SEP] marks at the beginning and end, respectively, before they can be input into Bert-wwm to capture sentence semantics. Formulas (1) and (2) represent the specific adjustment methods:

$$S_a = \{[CLS], x_{a_1}, x_{a_2}, \cdots, x_{a_n}, [SEP]\} \quad (1)$$

$$S_b = \{[CLS], x_{b_1}, x_{b_2}, \cdots, x_{b_n}, [SEP]\} \quad (2)$$

During the input process of text into Bert-wwm and the output process, it is necessary to pass through token embedding, position embedding, and segment embedding to handle the input. The three embeddings are added together to obtain the final output. This output is the encoding vector obtained from capturing the semantics of the sentence, as expressed below:

$$h_a = \{h_{a_0}, h_{a_1}, \cdots, h_{a_{l-1}}\} = \text{BERT} - \text{WWM}(S_a) \quad (3)$$

$$h_b = \{h_{b_0}, h_{b_1}, \cdots, h_{b_{l-1}}\} = \text{BERT} - \text{WWM}(S_b) \quad (4)$$

Where $h \in \mathrm{R}^{\mathrm{l}\times\mathrm{d}}$, l is the length of the sequence, d is the dimension of the word vectors generated by Bert-wwm, $h_{\mathrm{a_i}}$ represents the semantic representation of the i-th character in sequence a, and $h_{\mathrm{b_i}}$ represents the semantic representation of the i-th character in sequence b[6]. A simple method for calculating text similarity is to use the vector with the [CLS] mark as a representation of the entire sentence to directly calculate the similarity. However, if only the vector with the [CLS] location is used to represent the entire sentence, some information from other positions may be lost, leading to poor performance in practical applications. Therefore, when using Bert-wwm to calculate sentence similarity, it is usually the full output of Bert-wwm that is used. In order to extract more rich context semantic information and make the relevance between the context sentences of the article higher, it is necessary to re-train them. In this paper, the BERT models used are all Bert-wwm, base version models. It contains 12 layers and generates each character vector with a dimension of 768, totaling 110M parameters.

2.3 Calculation of Sentence Similarity Matrix

The sentence similarity matrix refers to the matrix composed of the mutual similarity between sentences in an article. The calculation of sentence similarity can be done in various ways. For instance, the classic word2vec model can be used to generate distributed vectors, from which similarity can be derived, or lexical text similarity can be used to directly calculate the similarity between sentences[7]. However, with the introduction of pre-trained models like BERT, using pre-trained models to calculate text similarity has gradually become a mainstream approach. Since pre-trained models have undergone large-scale unsupervised learning on datasets, they can generate higher-quality dynamic word vectors. This paper uses Bert-wwm to extract word vectors. Although the result of directly using Bert-wwm to extract word vectors is already excellent, to further enhance the quality of semantic extraction, this paper employs a negative sampling method to perform further pre-training on Bert-wwm, and then uses this re-trained Bert-wwm to extract word vectors, thereby ensuring that the word vectors generated by the context sentences of the article are more relevant. The overall structure of the negative sampling model is depicted in Figure 2.

Based on the sentence-level distributed hypothesis theory, two sentences with similar context semantic environments share similar semantics. Following this theory, the negative sampling method used in this paper involves taking the preceding and following sentences of a sentence as positive samples, and using other randomly selected sentences from the corpus as negative samples for training. Let O_c represent the encoding vector obtained for the current sentence through Bert-wwm, O_p represent the encoding vector obtained for the positive sample through Bert-wwm, and O_n represent the encoding vector obtained for the negative sample through Bert-wwm.

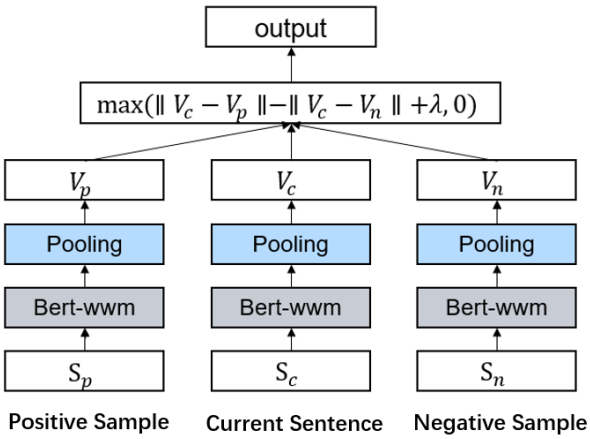

Figure 2: BERT Pretraining Based on Negative Sampling

The encoding vector is transformed into a sentence vector that can represent the entire sentence through pooling operations. In this paper, the pooling layer used is the average pooling layer[8]. The calculation is as follows:

$$V_{c,avg} = \sum_{i=1}^{l_c} \frac{O_{c_i}}{l_c} \tag{5}$$

$$V_{p,avg} = \sum_{i=1}^{l_p} \frac{O_{p_i}}{l_p} \tag{6}$$

$$V_{n,avg} = \sum_{i=1}^{l_n} \frac{O_{n_i}}{l_n} \tag{7}$$

Where $V_{c,avg}$, $V_{p,avg}$, $V_{n,avg}$ are the sentence vectors of the current sentence, positive sample, and negative sample, respectively. l is the length of the sequence, and the optimized loss function is as follows:

$$loss = \max(\| V_{c,avg} - V_{p,avg} \| - \| V_{c,avg} - V_{p,avg} \| + \lambda, 0) \tag{8}$$

Here, $\|\cdot\|$ denotes the distance measure, and this paper uses the Euclidean distance to measure distance. λ represents the margin between edges, and the model optimizes the model by making the distance between the current sentence and the positive sample closer than the distance from the negative sample by λ.

After the model training is complete, the sentence vectors produced by the pre-trained model need to use a similarity calculation method to determine the similarity between them, such as cosine similarity, or dot product[9]. During the experimental process, it was found that using vector dot product to calculate the similarity between vectors is more effective. The dot product calculation formula is as follows:

$$\alpha = [a_1, a_2, \cdots, a_n] \tag{9}$$

$$\beta = [b_1, b_2, \cdots, b_n] \tag{10}$$

$$\alpha \cdot \beta = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \tag{11}$$

Let E_{ij} represent the similarity between the i-th sentence and the j-th sentence in the text, and V_i, V_j be the sentence vectors of the i-th and j-th sentence text, respectively, through the pre-trained model, then the calculation of the sentence similarity matrix is as follows.

$$E_{ij} = V_i \cdot V_j \tag{12}$$

2.4 Calculation of Sentence Scores

After obtaining the sentence similarity matrix using the pre-trained Bert-wwm model, the matrix will be used to calculate the score of each sentence, with higher scores indicating a higher priority in the extraction process[10]. To this end, this paper proposes a graph-based text summarization extraction method. Each sentence in the text is converted into a node in the graph, and nodes are connected by directed edges, with edge weights representing the similarity between two sentences. Before obtaining the final score of a sentence, the article's theme is considered, and a weighting mechanism is introduced to adjust and optimize the scores. Compared to algorithms like TextRank, which are based on undirected weighted graphs, this algorithm uses directed weighted graphs, and the position information between sentences can affect the extraction results.

Assuming the preprocessed document is denoted as $D = \{s_1, s_2, \cdots, s_n\}$, and e_{ij} represents the similarity between the edges (s_i,s_j), a sentence score can be calculated based on the following method:

$$score_i = \sum_{j\in\{1,\cdots,i-1,i+1,\cdots,n\}} e_{ij} \tag{13}$$

However, this method treats the graph as an undirected graph and does not introduce position information. In reality, there is a pair of directed edges between sentences s_i and s_j, pointing from s_i to s_j and from $s_j$to s_i.Considering the direction of the edges, the score calculation for a sentence is as follows:

$$score_i = \sum_{j<i} e_{ij} + \sum_{j>i} e_{ij} \tag{14}$$

For the edge (s_i,s_j), when $j < i$, the edge is a forward edge, indicating that the edge points from s_i to a node located before s_i in the article. When $j > i$, the edge is a backward edge, indicating that the edge points from s_i to a node located after s_i. Although formula (14) distinguishes between forward edges and backward edges, the calculated result is the same as that of formula (13). To address this, two hyperparameters λ_1 and λ_2 are introduced, representing the weight of forward edges and backward edges, respectively. Thus, the sentence score calculation is as follows:

$$score_i = \lambda_1 \sum_{j<i} e_{ij} + \lambda_2 \sum_{j>i} e_{ij} \tag{15}$$

The introduction of hyperparameters allows for different weights for forward and backward edges of a sentence. When λ_1 and λ_2 are both equal to 1, the result is equivalent to the calculation method using undirected edges. Subsequent experiments show that λ_1 tends to be negative, which means that when a node is similar to previous nodes, it will reduce the score of that node, making it easier for nodes that are further along to have lower scores and harder to achieve high scores that can be extracted.

2.5 Adjustment of Sentence Scores

However, the aforementioned calculation method treats sentences as independent individuals without considering the relationship between the sentences and the overall theme of the article. Key words in an article often represent the overall theme, and the more keywords appear in a specific sentence, the closer it is to the theme, making the sentence more important. To count the keywords in an article, text preprocessing is required[11].

Firstly, there is tokenization. After a comprehensive evaluation of various tokenization tools, it is decided to use the Jieba tokenization tool as the tokenization tool in this paper. Next is the removal of stopwords, which are words that have no actual meaning in practical applications, such as "a", "the", "or", and so on. In addition to removing these meaningless words, special symbols need to be removed as well.

The calculation of keywords uses the TF-IDF algorithm. The TF-IDF algorithm gives priority to words that have a high frequency in the paper but a low frequency in the entire corpus[12], as these keywords are more representative of the entire article's meaning. Among them, IDF can be obtained in advance through statistics, which is the inverse document frequency, indicating the frequency of a word appearing in all documents in the entire corpus. TF is the term frequency and needs to be dynamically obtained based on the article input. In this algorithm, the TF-IDF algorithm is used to calculate the top 10 most frequent words in the paper as keywords. Through these keywords, the keyword coverage rate of each sentence can be calculated, which will affect the final score of the sentence. Let $K(s_i)$ represent the keyword coverage rate of s_i, then the calculation method of $K(s_i)$ is as follows.

$$K(s_i) = \frac{len(keywords(s_i))}{len(s_i)} \tag{16}$$

where $len(s_i)$ represents the number of words after s_i is tokenized and filtered, and $len(keywords(s_i))$ represents the number of keywords contained in s_i. When calculating sentence scores, a node weight is introduced to optimize the node scores. Let w_i represent the weight of s_i, then w_i is calculated as follows:

$$w_i = 1 + h \cdot K(s_i) \tag{17}$$

Here, h is a hyperparameter that controls the extent of the influence of the weight on the calculation. When h is 0, it indicates that no weight optimization is used.The mechanism of introducing the weight optimizes the calculation after which the sentence score calculation process is as follows:

$$score_i = w_i\left(\lambda_1 \sum_{j<i}(e_{ij} \cdot w_j) + \lambda_2 \sum_{j>i}(e_{ij} \cdot w_j)\right) \tag{18}$$

The weight of a node not only affects its own score but also has an impact on the scores of other nodes. Then, sentences are sorted according to the final score, and a certain number of sentences are selected until the set length requirement is met. Finally, when generating the summary set, sentences are extracted in the order they appear in the original text[13].

3. Experimental Results and Analysis

3.1 Experimental Dataset

This chapter's experiments use the test set from NLPCC 2017[14], which contains data from headline news, including the full text of the news and the corresponding abstracts, totaling 50,000 data items. From the full text of the news and the corresponding benchmark abstract data, 10,000 pairs of samples were randomly selected for testing based on the text length distribution. The selected news texts cover news from various fields such as healthcare and current events.

3.2 Baseline Models

The baseline models compared in this experiment are primarily TextRank, LexRank, and some improved methods of these models. For example, TextRank-word2vec indicates the use of the word2vec model as the calculation model for edge weights and TextRank to compute text summaries. TextRank-Bert-wwm indicates using Bert-wwm to extract text feature vectors. Similarly, LexRank-word2vec signifies using the word2vec model to calculate edge weights, while LexRank is used to compute text summaries. LexRank-Bert-wwm indicates using Bert-wwm to generate edge weights.

3.3 Evaluation Metrics

The algorithm uses the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) evaluation metric. The principle of the ROUGE evaluation metric is to compare the text summaries extracted by the algorithm with a set of benchmark text summaries and obtain corresponding scores. Then, through these scores, measure the similarity between the automatically generated text summaries and the benchmark text summaries.

3.4 Analysis of Experimental Results

To verify the impact of pretraining on the results, the pretrained Bert-wwm model and the unpretrained Bert-wwm model will be used to extract sentence semantic information, with the weight adjustment mechanism not being used and the weights for both forward and backward edges both set to 0.5. It is also necessary to verify the impact of the number of negative samples during pretraining on the results. Therefore, a comparative experiment was designed, including scenarios with one positive sample corresponding to one negative sample, one positive sample corresponding to two negative samples, and one positive sample corresponding to three negative samples. The experimental results are shown in Table 1.

Table 1: Results of Pretraining Experiments

	ROUGE-1	ROUGE-2	ROUGE-L
No pretraining	0.19769	0.08530	0.16234
One positive sample, one negative sample	0.27753	0.12672	0.21517
One positive sample, two negative samples	0.27207	0.12242	0.21133
One positive sample, three negative samples	0.26976	0.11953	0.20875

From the results, it can be seen that pretraining Bert-wwm can significantly improve the accuracy of extraction. Additionally, when one positive sample is sampled with one negative sample, the effect is the best, with ROUGE-1 of 0.27753, ROUGE-2 of 0.12672, and ROUGE-L of 0.21517. Therefore, in the subsequent experimental calculations, the pretrained model used is the model obtained from pretraining with one positive sample corresponding to one negative sample.

In the process of text summarization extraction, there are multiple hyperparameters involved, including the parameter that controls the weight of keyword proportion h, and the parameters that control the forward edge weight λ_1 and the backward edge weight λ_2. To simplify the calculation and facilitate the control of variables, it is assumed that $\lambda_1+\lambda_2=1$, and the tuning results are optimized with reference to the ROUGE-L results. λ_1、λ_2, and h are changed with a step size of 0.1. Figure 3 shows the analysis of the parameter tuning results.

The image displays some typical curves during the parameter tuning process, including the curve with the best performance, as well as curves around the optimal value of h, and also a curve with h set to 0. From the figure, it can be seen that when h is 0.4, λ_1 is -0.7, and λ_2 is 1.7, the model's extraction performance is the best. When λ_1 is in the interval [-1, 0] and [0.5, 1], the weight mechanism has a significant promotion effect on the results. When λ_1 is in the interval [-1, 0], the algorithm's performance gradually improves as h increases, and the optimal performance is achieved when h is 0.4. After that, as h increases, the effect gradually decreases. When λ_1 is in the interval [0.5, 1], the results increase with the increase of h, but the overall performance of this interval is significantly worse than that when λ_1 is negative.

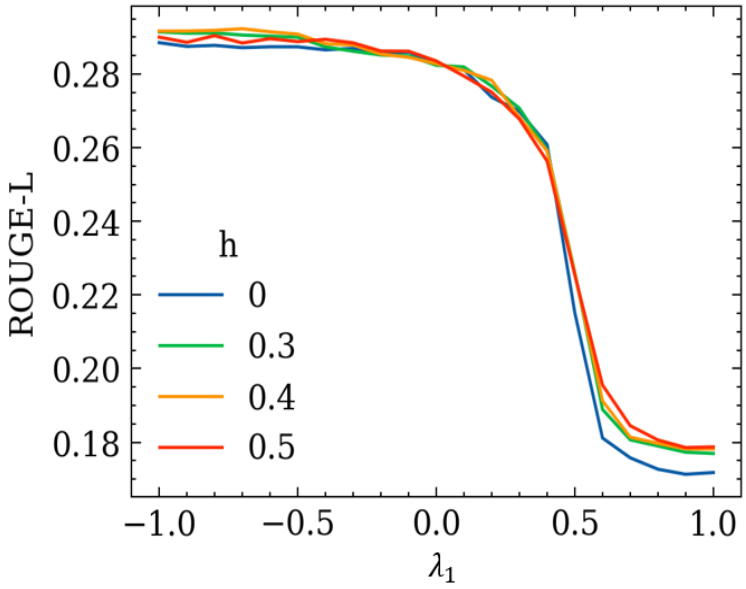

Figure 3: Parameter Tuning Change Trend

Through comparative experiments, it can be observed that the use of a weighting mechanism to adjust and optimize the extraction of sentences effectively increases the accuracy of summary extraction, making the extracted summaries more in line with the article's theme and more representative. By utilizing different weights for forward and backward edges, the sentence position information can be employed to assign different extraction priorities to sentences at different positions in the article, making the summary extraction strategy of the algorithm more consistent with the stylistic characteristics of news, thereby extracting higher-quality text summaries.

When computing the similarity matrix, a method is needed to measure the similarity between various vectors. To date, the experiments have used dot products to calculate the similarity between vectors. To compare the impact of different calculation methods on the results, various methods for calculating vector similarity have been used to measure the similarity between vectors. At this point, the weighting adjustment mechanism is introduced, and the optimal parameters are used. ROUGE-1, ROUGE-2, and ROUGE-L are used as reference metrics for the experiments. The experimental results are shown in Table 2.

Table 2: Experimental Results of Different Similarity Calculation Methods

Similarity Calculation Method	ROUGE-1	ROUGE-2	ROUGE-L
Dot Product	0.35781	0.19690	0.29227
Cosine Similarity	0.34998	0.19013	0.28616
Eclidean Distance	0.29052	0.15360	0.24013
Manhattan Distance	0.29007	0.15316	0.24030

From the results, it can be seen that using dot product for similarity calculation yields the best results, with the highest values for ROUGE-1, ROUGE-2, and ROUGE-L. Therefore, in the EMW-Sum model, dot product is used to calculate sentence similarity.

To test the performance of the EMW-Sum algorithm compared to TextRank and LexRank-based algorithms, this paper set up a comparative experiment with different algorithms. The comparison objects are the TextRank algorithm and the relevant improved algorithms of TextRank. In addition, to verify the effectiveness of using Bert-wwm, a comparison group using word2vec to calculate the similarity matrix and EMW-Sum to solve is added. Table 3 shows the comparison results of various models.

Table 3: Comparison Results of Different Models

Model	ROUGE-1	ROUGE-2	ROUGE-L
TextRank	0.29208	0.13600	0.22930
TextRank-word2vec	0.30292	0.14177	0.23287
TextRank-Bert-wwm	0.30043	0.13915	0.23040
LexRank-word2vec	0.29923	0.13902	0.22882
LexRank-Bert-wwm	0.29122	0.13496	0.22809
EMW-Sum-word2vec	0.31554	0.16707	0.25808
EMW-Sum	**0.35781**	**0.19690**	**0.29227**

Through comparative experiments, the performance of TextRank, LexRank, and their improvements are compared with the model proposed in this paper. From the above table, it can be seen that the method EMW-Sum proposed in this paper has significantly higher scores in all evaluation metrics compared to the scores of other algorithms. Compared to the EMW-Sum algorithm that uses word2vec, the algorithm based on pre-trained Bert-wwm is superior, indicating that sentence vectors generated by Bert-wwm are more suitable for summarization extraction.

4. Conclusion

This paper first introduces the steps of the EMW-Sum algorithm for summary extraction, including text preprocessing and input, pretraining of Bert-wwm, and the specific steps for summary extraction. Then, it describes the dataset used in the experiment and the relevant environmental configuration, followed by the evaluation metrics used to assess the effectiveness of summary extraction. Subsequently, the optimization of multiple parameters and calculation methods within the model is discussed to obtain the optimal calculation method. Finally, the effectiveness of the EMW-Sum algorithm is verified by comparing it with some classic and commonly used baseline models on a news summarization data set.

References

[1] Mihalcea R, Tarau P. Textrank: Bringing order into text[C]//Proceedings of the 2004 conference on empirical methods in natural language processing. 2004: 404-411.

[2] Devlin J. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[3] Cui Y, Che W, Liu T, et al. Pre-training with whole word masking for chinese bert[J]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2021, 29: 3504-3514.

[4] Abdi A, Shamsuddin S M, Idris N, et al. A linguistic treatment for automatic external plagiarism detection[J]. Knowledge-Based Systems, 2017, 135: 135-146.

[5] Nápoles G, Dikopoulou Z, Papageorgiou E, et al. Prototypes construction from partial rankings to characterize the attractiveness of companies in Belgium[J]. Applied Soft Computing, 2016, 42: 276-289.

[6] Goyal R, Dymetman M, Gaussier E. Natural language generation through character-based rnns with finite-state prior knowledge[C]//Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. 2016: 1083-1092.

[7] Yu D, Wang H, Chen P, et al. Mixed pooling for convolutional neural networks[C]//Rough Sets and Knowledge Technology: 9th International Conference, RSKT 2014, Shanghai, China, October 24-26, 2014, Proceedings 9. Springer International Publishing, 2014: 364-375.

[8] Li B, Zhou H, He J, et al. On the sentence embeddings from pre-trained language models[J]. arXiv preprint arXiv:2011.05864, 2020.

[9] Yu Y, Wang Y, Mu J, et al. Chinese mineral named entity recognition based on BERT model[J]. Expert Systems with Applications, 2022, 206: 117727.

[10] Mahata D, Kuriakose J, Shah R, et al. Key2vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings[C]//Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). 2018: 634-639.

[11] Yao L, Pengzhou Z, Chi Z. Research on news keyword extraction technology based on TF-IDF and TextRank[C]// 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS). IEEE, 2019: 452-455.
[12] Genest P E, Lapalme G. Framework for abstractive summarization using text-to-text generation[C]//Proceedings of the workshop on monolingual text-to-text generation. 2011: 64-73.
[13] Hua L, Wan X, Li L. Overview of the NLPCC 2017 shared task: single document summarization[C]//Natural Language Processing and Chinese Computing: 6th CCF International Conference, NLPCC 2017, Dalian, China, November 8–12, 2017, Proceedings 6. Springer International Publishing, 2018: 942-947.
[14] Yuan W, Neubig G, Liu P. Bartscore: Evaluating generated text as text generation[J]. Advances in Neural Information Processing Systems, 2021, 34: 27263-27277.