Keywords extraction algorithm based on attention mechanism of BERT model

Yanfen Luo*
Information Engineering College, Nanchang Hangkong University, Nanchang 330038, Jiangxi, China

ABSTRACT

Text keyword extraction refers to finding the words or phrases from an article that best represent its topic or content. Text keyword extraction is not only helpful to quickly understand the content of the text, but also can be used in NLP tasks such as information retrieval, automatic summarization, text classification, etc. In order to improve the performance of keyword extraction algorithm, this paper proposes a keyword extraction model AttentionRank based on the self-attention mechanism of BERT model by using BERT pre-trained language model. The pre-trained BERT language model can recognize keywords through self-attention and cross-attention, and enhance the ability of keyword extraction algorithm to understand context. The experimental results show that AttentionRank has obvious advantages in keyword extraction compared with LDA and LSA algorithms.

Keywords: BERT model, AttentionRank, keyword extraction

1. INTRODUCTION

Text keyword extraction is a key technology in the field of Natural Language Processing (NLP). In recent years, many text keyword extraction algorithms have emerged, including those based on text itself, such as Term Frequency-Inverse Term Frequency (TF-IDF) and TextRank algorithm. There is also keyword extraction algorithms based on topic model, such as Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA) algorithm, etc.[1]. The keyword extraction algorithm based on topic model is superior to the traditional TF-IDF and TextRank algorithms based on text itself, and its performance is more obvious. However, the key word extraction algorithm still faces many challenges, including polysemy, long text processing and cross-language key word extraction. In response to these challenges, many related technical research directions have been derived, including the use of deep learning architecture semantic networks, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), Transformer, etc., to improve the performance of keyword extraction[2]. The entity information in the knowledge graph is combined with the relationship information between entities through the knowledge graph to improve the accuracy and ease of keyword extraction. In this paper, a keyword extraction model based on the self-attention mechanism of Bidirectional Encoder Representations from Transformers (BERT) model, is proposed by using BERT pre-trained language model. This is because the self-attention mechanism of the BERT model is able to capture the dependencies between different positions in the input sequence and generate high-quality text representations based on these dependencies[3]. AttentionRank keyword extraction algorithm based on BERT model uses attention mechanism to enhance the recognition ability of keywords, and shows obvious advantages in keyword extraction tasks.

2. BERT MODEL INFRASTRUCTURE

BERT is a pre-trained language representation model based on the Transformer model, proposed by Google in 2018. The infrastructure of the BERT model is shown in Figure 1[4].

*729035457@qq.com

International Conference on Optics, Electronics, and Communication Engineering (OECE 2024), edited by Yang Yue, Proc. of SPIE Vol. 13395, 133954H · © 2024 SPIE · 0277-786X · Published under a Creative Commons Attribution CC-BY 3.0 License · doi: 10.1117/12.3049066

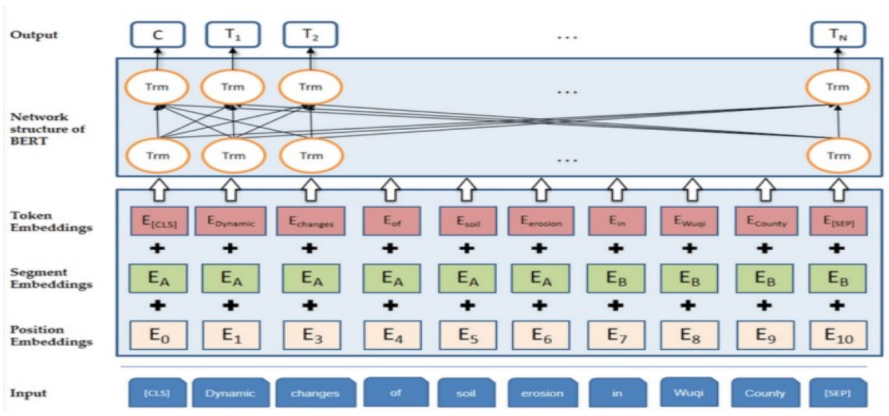

Figure 1. BERT model architecture diagram.

(1) Input layer

BERT's input is a sequence of raw text, which can be a single sentence or two sentences (for example, the question and answer in a question-and-answer task). These texts need to go through specific pre-processing steps before being input into the model.

(2) Tokenization and Embeddings

Tokenization: Input text is first split into tokens using a Tokenizer. This step usually involves converting text to lower case, removing punctuation, word segmentation, and so on. BERT uses the WordPiece method to further divide words into subwords to optimize the size of the vocabulary and the generalization power of the model.

Token Embeddings: The Token after the participle is mapped to a high-dimensional space to form Token Embeddings. This is done by looking for a pre-trained embedded matrix that provides a fixed-size vector representation for each Token.

Segment Embeddings: Since BERT is capable of processing two sentences as input (for example, in a sentence pair classification task), a way to distinguish between two sentences is needed. Segment Embeddings are used for this purpose, adding an additional embed to each Token to indicate which sentence it belongs to (usually "A" or "B").

Position Embeddings: Since the Transformer model itself does not have the ability to process the Token location information in the sequence, position embeddings are required to provide this information. Each position has a unique embedding vector that is learned during training.

Token Embeddings, Segment Embeddings, and Position Embeddings are added together to get the final input embeddings for each Token.

(3) Network Structure of BERT

Its core component is a multi-layered Transformer Encoder, with each layer containing self-attention mechanisms and feedforward networks, a structure capable of capturing long distance dependencies and generating high-quality text representations.

Self-attention mechanism: Allows the model to focus on tokens in different locations while processing the sequence and calculate the attention weights between tokens, thus capturing dependencies in the input sequence. The Self-Attention Mechanism in BERT, a core component of the Transformer architecture, allows the model to focus on information at different points in the input sequence and assign different weights based on how important that information is.

Feedforward neural network: The output of the self-attention mechanism is further transformed to extract higher-level features.

Residual connection and layer normalization: Used to improve the training stability and effectiveness of the model, helping to mitigate gradient disappearance and gradient explosion problems.

(4) Output

The output of BERT depends on the specific task. In the pre-training phase, BERT employed two tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP).

MLM: In this task, BERT predicts tokens that are randomly masked in the input sequence. The output of the model is the probability distribution for each masked Token, obtained through the Softmax layer.

NSP: This task requires BERT to predict whether two sentences are consecutive or not. The output of the model is a probability distribution of a binary classification problem.

3. HOW THE SELF-ATTENTION MECHANISM WORKS IN BERT

First, BERT converts the input text into embedding vectors. These embeddings include word embeddings, position embeddings, and (in some tasks) segment embeddings[5].

For each position in the input sequence, BERT generates three vectors: a query vector, a key vector, and a value vector. These three vectors are obtained by multiplying the embedding vector by three different weight matrices (learned during training)[6].

For each position in the input sequence, BERT calculates the dot product between the query vector for that position and the key vectors for all positions in the sequence. These dot product values are then divided by a scaling factor (usually the square root of the key vector dimension) to obtain the attention scores[7].

The attention score is entered into the softmax function to calculate a probability distribution that describes the importance of each position in the sequence to the current position[8].

Using the probability distribution obtained in the previous step as the weights, the value vectors of all positions in the sequence are weighted and summed to obtain the output vector of the current position. This output vector is a weighted combination of all the position information in the sequence, reflecting how important the different positions are to the current position[9].

To enhance the representation of the model, BERT uses a multi-head attention mechanism[9]. This means that BERT calculates several different self-attention weights for each input position and combines these weights to generate the final output vector. Each “head” independently learns different attention weights to capture different aspects of the input sequence.

Through the above process, BERT’s self-attention mechanism is able to capture dependencies between different locations in the input sequence and generate high-quality text representations based on these dependencies. This makes BERT excellent at a variety of natural language processing tasks.

4. KEYWORDS EXTRACTION BASED ON ATTENTION MECHANISM OF BERT MODEL

4.1 Keyword extraction model based on BERT model self-attention mechanism

Keyword extraction is the identification of words or phrases that represent the main topic of a document. In this section, AttentionRank, a keyword extraction model based on the self-attention mechanism of BERT model, is proposed. Pre-trained BERT language models can identify keywords through self-attention and cross-attention. The working process of AttentionRank model to extract text keywords is shown in Figure 2.

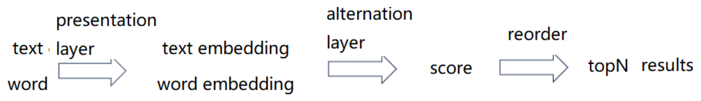

Figure 2. Keyword extraction process of AttentionRank model.

First, the text is divided into words, and BERT is used to calculate the embedding value of the text and the embedding value of the word. Then, the cosine similarity between the embedding vector of the word and the embedding vector of the text is calculated, and the text keywords are sorted according to the similarity.

To obtain self-attention values for individual words in a sentence, we extract word-word self-attention weights from a pre-trained BERT model, and then sum up the attention that the word receives from other words in the same sentence. Next, we add up the attention of the member words of this candidate word to calculate the self-attention value of candidate c in the sentence. The document-level cumulative self-attention value of candidate word c is calculated as the sum of all self-attention values of word c in each sentence in document d. This method helps us to understand the importance of each part of the text and the relationship between them, and has important application value for natural language processing tasks such as keyword extraction and text classification.

4.2 Specific implementation of AttentionRank model

AttentionRank integrates a cumulative self-attention component with a cross-attention component to calculate a candidate's final score. The structure of AttentionRank model is shown in Figure 3.

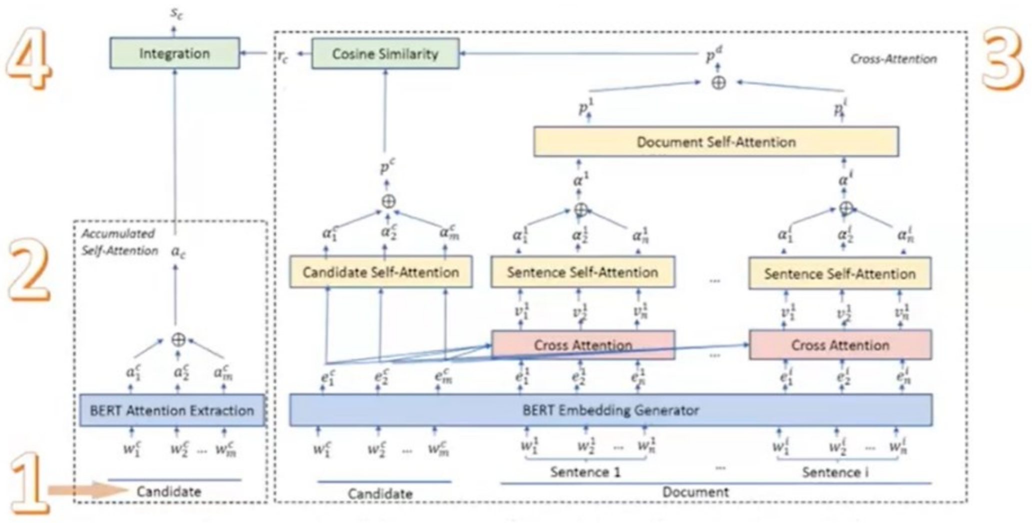

Figure 3. Structure of AttentionRank model.

(1) Candidate generation. We use the candidate extraction module implemented in EmbedRank. The module first uses parts of speech (PoS) to identify words labeled NN, NNS, NNP, NNPS, JJ, and so on. The python package NLTK is then used to generate noun phrases, which are the candidates. Most of the parameters of the program can be controlled by a simple switch in a text file edited by the experimenter or in the program code, minimizing the need for programming to create new experiments.

(2) Cumulative self-attention calculation. A collection of member words for candidate *c*: w^c={w_1^c,..., w_m^c}. The self-attention value of candidate *c* can be calculated by equations (1) and (2).

$$a = softmax_{row}(w^c \cdot w^{cT}) \cdot w^c \tag{1}$$

$$a_c = AVE(a[:,i]) \quad a_c \in R^H \tag{2}$$

(3) Cross attention computation. A pre-trained BERT model can generate a word-level embedding set of candidate word *c*: E^c={e_1^c,...e_m^c}, in which $e_i \in R^H$. Similarly, E^i is the representation of the sentence *i* containing n words: E^i={e_1^i,...e_n^i}, in which $e_i \in R^H$.

Given a sentence *i*, expressed as E^i, and a candidate word *c*, expressed as: E^c, sentence-to-candidate and candidate-to-sentence similarity can be measured by equations (4) and (5).

$$S = E^i \cdot E^{cT} \tag{3}$$

$$S_{i2c} = softmax_{row}(S) \tag{4}$$

$$S_{c2i}=softmax_{column}(S) \tag{5}$$

Sentence-to-candidate cross attention and candidate-to-sentence cross attention are calculated from equations (6) and (7).

$$A_{i2c}=S_{i2c}\cdot E^{c} \tag{6}$$

$$A_{c2i}=S_{i2c}\cdot S_{c2i}^{T}\cdot E^{i} \tag{7}$$

Based on these cross-attention, a set of neologism embeddings for sentence *i* is constructed.

$$V^{i}=AVG(E^{i},A_{i2c},E^{i}\odot A_{i2c},E^{i}\odot A_{c2i}) \tag{8}$$

V^{i} is a set of word embeddings that contain the relationship between candidate *c* and sentence *i*. Standardized sentence embeddings can be calculated using equations (9) and (10).

$$I=softmax_{row}(V^{i}\cdot V^{iT})\cdot V^{i} \tag{9}$$

$$\alpha^{i}=AVE(I[:,i]) \quad \alpha^{i}\in R^{H} \tag{10}$$

in which *AVE* indicates taking the mean value.

Given a document *d*, it contains a set of sentences:$E^{d}=\{\alpha^{1},\ldots\alpha^{i}\}$. A standardized document embedding can be calculated using equations (11) and (12).

$$P=softmax_{row}(E^{d}\cdot E^{dT})\cdot E^{d} \tag{11}$$

$$p^{d}=AVE(P[:,i]) \quad p^{d}\in R^{H} \tag{12}$$

The word embeddings set of candidate word c:$E^{c}=\{e_{1}^{c},\ldots e_{m}^{c}\}$. Candidate word embeddings can be calculated using equations (13) and (14).

$$C=softmax_{row}(E^{c}\cdot E^{cT})\cdot E^{c} \tag{13}$$

$$p^{c}=AVE(C[:,i]) \quad p^{c}\in R^{H} \tag{14}$$

The correlation between a candidate word *c* and document *d* is determined by the cosine similarity between p^{c} and p^{d}, as shown in equation (15).

$$r_{c}=\frac{p^{c}\cdot p^{d}}{\|p^{c}\|\cdot\|p^{d}\|} \tag{15}$$

(4) Final score calculation. Candidate *c*'s final score is a linear combination of cumulative self-attention values and cross-attention correlation values:

$$s_{c}=d*a_{c}+(1\text{-}d)*r_{c}, \quad \text{where } d\in[0,1] \tag{16}$$

According to the final score of the candidate words, the keyword topN results can be obtained.

4.3 AttentionRank model extracts keyword experimental results

In this section, accuracy rate (p-value), recall rate (R-value) and F1 value are still used to evaluate the BERT model algorithm compared with LSA and LDA. The comparison results are shown in Figures 4-6.

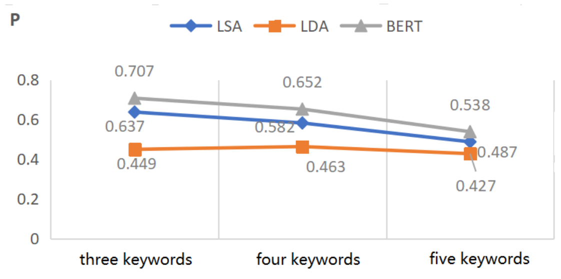

Figure 4. Comparison of accuracy rate (p-value) between BERT model and subject model.

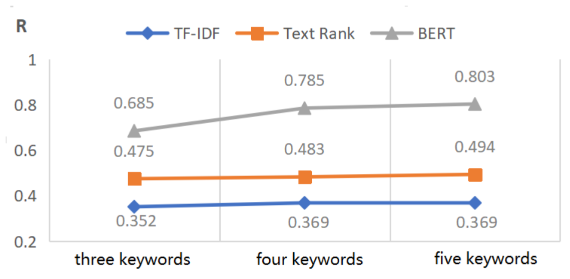

Figure 5. Comparison of recall rate (R-value) between BERT model and subject model.

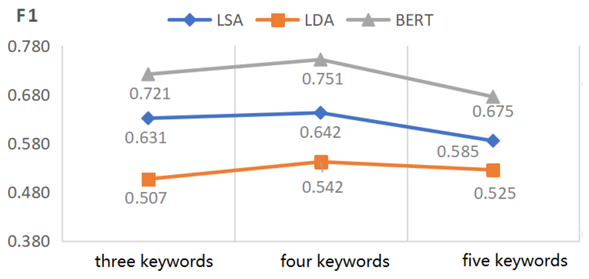

Figure 6. Comparison of F1 value between BERT model and subject model.

5. CONCLUSIONS

In this paper, AttentionRank, a keyword extraction model based on the self-attention mechanism of BERT model, is proposed by using BERT pre-trained language model. The pre-trained BERT language model can recognize keywords through self-attention and cross-attention, and enhance the ability of keyword extraction algorithm to understand context.The experimental results show that AttentionRank has obvious advantages in keyword extraction tasks. By continuously improving the accuracy and robustness of the keywords extraction algorithm, so that it can adapt to different types and fields of text data environment, the future keyword extraction algorithm is expected to achieve better results in the processing of complex text and cross-language scenes.

REFERENCES

[1] Alzaidy, R., Caragea, C. and Giles, C. L., "Bi-LSTM-CRF sequence labeling for key phrase extraction from scholarly documents," The World Wide Web Conference 2551-2557 (2019).
[2] Nallapati, R., Zhai, F. and Zhou, B., "Submariner: A recurrent neural network-based sequence model for extractive summarization of documents," Proceedings of the AAAI Conference on Artificial Intelligence (2017).
[3] Li, Z., Wei, Y., Zhang, Y., et al., "Exploiting coarse-to-fine task transfer for aspect-level sentiment classification," Proc AAAI Conf Artif. Intell., 33, 4253-4260 (2019).
[4] Wu, X., Du, Z. and Guo, Y., "A visual attention-based keyword extraction for document classification," Multimedia Tools and Applications 77, 25355-25367 (2018).
[5] Haddoud, M. and Abdeddaim, S., "Accurate. keyphrase extraction by discriminating overlapping phrases," Journal of Information Science, 40(4), 488-500 (2014).
[6] Jo, T., "Neural based approach to keyword extraction from documents," Computational Science and Its Applications—ICCSA 2003: International Conference Montreal, Canada, 456-461 (2003).
[7] Alrehamy, H. and Walker, C., "Exploiting extensible background knowledge for clustering-based automatic keyphrase extraction," Soft Computing, 22(21), 7041-7057 (2018).
[8] Yang, M., Liang, Y., Zhao, W., et al., "Task-oriented keyphrase extraction from social media," Multimedia Tools and Applications, (2018).
[9] Sterckx, L., Demeester, T., Deleu, J., et al., "Creation and evaluation of large key phrase extraction collections with multiple opinions," Language Resources and Evaluation 52, 503-532 (2018).