A Contrastive Learning Framework for Keyphrase Extraction

Jing Song[1], Xian Zu[2,3†], Fei Xie[2]

[1]Department of Electronic Information and Electrical Engineering, Hefei Normal University, Lianhua Road, Hefei 230601, China
[2]Department of Computer and Artificial Intelligence, Hefei Normal University, Lianhua Road, Hefei 230601, China
[3]Department of Safety Science and Engineering, University of Science and Technology of China, Hefei 230026, China

Keywords: Keyphrase extraction; Contrastive learning; Supervised; n-gram features; Document embedding

Citation: Song J., Zu X., Xie F.: A Contrastive Learning Framework for Keyphrase Extraction. Data Intelligence 6(4), 1032-1056 (2024). doi: https://doi.org/10.3724/2096-7004.di.2024.0018
Submitted: September 05, 2023; Revised: October 10, 2023; Accepted: January 09, 2024

ABSTRACT

Keyphrase extraction aims to extract important phrases that reflect the main topics of a document. Recently, deep learning methods are used to model semantic information and rank candidates based on the similarities between the n-grams and the document. However, existing keyphrase extraction methods mainly caused the keyphrase extraction task to be independent of the embedding. Based on the fact that phrases that are semantically closer to the document are more likely to become keyphrases, we propose a novel contrastive learning strategy for supervised keyphrase extraction by integrating local and global Information of the document. A pre-trained RoBERTa model is used to model contextual information of sub-words in the document. Then, the embedding vectors of n-grams and the document are calculated by the convolution neural layers. Finally, we propose a novel loss function for efficiently ranking candidate phrases by combining n-gram features and document embeddings during the training of the model.

1. INTRODUCTION

Keyphrases can help users quickly understand the main topics of the given document that serve as a condensed summary. Due to the important roles of keyphrases, they have been widely used in many domains, such as Web advertising, text summarization, recommendation systems, and information retrieval [1, 2]. The process of keyphrase extraction is to select multiple words or phrases that mostly capture the main topics in a document. Keyphrase extraction methods can be divided into two categories:

† Corresponding author: Xian Zu (Email: zuxian@mail.ustc.edu.cn; ORCID:0000-0003-4507-8736).

© 2024 Chinese Academy of Sciences. Published under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

supervised and unsupervised [3]. Traditional supervised approaches usually train a classifier to identify whether a candidate phrase is a keyphrase, considering Keyphrase extraction to be a classification problem. Although the supervised method needs labeled data to train keyphrase identifier, the quality of extracted keyphrases is much higher than that of the unsupervised method. Unsupervised methods utilize statistical features, such as term frequency, position, and length, or build a co-occurrence graph to rank candidate phrases. The defect of this type of method is that it ignores the semantic feature information of words and text. For example, the graph-based method is more biased towards the local co-occurrence information between words, ignoring the order of the document and the contextual semantic information of words [4].

Recently, with the development of deep learning technologies, various representation methods are proposed to model the semantic and ordering information of a text. Then, the problem of keyphrase extraction is viewed as a sequence labeling task [5]. But this kind of algorithm is more inclined to use the local contextual semantic information of the word. Recent studies show that the embedding of the whole document is beneficial to the extraction of keyphrases with the consideration that keyphrases and the document have higher similarities in the feature space. For example, Bennani-Smires et al. proposed the EmbedRank method that uses Sent2Vec to compute the embedding vectors of the document and each candidate phrase [6]. In the SIFRank method, a pre-trained language model is used to embed the document and candidate [7]. Ding and Luo proposed a hybrid attention model AttentionRank [8] to identify keyphrases that calculates self-attention and cross-attention using a pre-trained language model. The above research work can show that the semantic information of the whole document has important significance and reference value for the keyphrase extraction algorithm in the selection of semantic features.

Based on the motivation that humans often learn and distinguish different things through comparison, the idea of contrastive learning has been widely used in the fields of computer vision and natural language processing in the past two years [9, 10, 11, 12]. The core of contrastive learning is how to construct a set of positive and negative samples. The ultimate goal is to make samples belonging to the same class closer and samples of different classes farther away. To remedy the shortcomings of the above existing works, in this study we propose a new supervised model—CNNRank for keyphrase extraction based on the idea of contrastive learning. The motivation of CNNRank is that phrases that are more similar to the document semantics are more likely to become keyphrases. That is to say, during the training process of the supervised model, the vectors of candidate phrases that are closer to the document semantics will be closer to the document vector (positive samples). The score of such phrases will increase, which means that the possibility of becoming a keyphrase is greater. On the contrary, phrases with low similarity to the document are farther away from the document vector (negative samples), and the scores of such phrases will be lower. The CNNRank model uses a pre-trained language model to compute embedding vectors for documents and candidate phrases expressed in n-gram format. Then, in the training phase of our neural network model, the embedding vectors of the candidate phrases are combined with the documents. Unlike previous deep learning methods that treat keyphrase extraction as a classification problem, we treat the keyphrase extraction task as a regression problem.

CNNRank is the first to use the embedding information of both candidate phrases and the document in the deep learning network training process for supervised keyphrase extraction. It unites the two stages of text embedding and keyphrase extraction. The main contributions of this paper are summarized as follows:

- We propose a novel contrastive learning framework, which is a new supervised keyphrase extraction model called CNNRank. It utilizes a pre-trained language model and CNN layers to embed candidate phrases and the whole document. In addition, we compare the impact of different document feature representation methods on keyphrase extraction results.
- We design a new loss function for keyphrase extraction during the supervised model training process, aiming to make the representations of keyphrases and documents in the semantic vector space get closer through training, while non-keyphrases get farther apart. To the best of our knowledge, this is the first work to incorporate semantic information of the document into the training process for regression tasks.
- We showed that the CNNRank model outperforms the compared baselines. Through various comparison experiments, it is proved that the fusion of document feature representation in the algorithm is important to improve the extraction effect.

The remainder of the paper is structured as follows: Section 2 discusses the related research of keyphrase extraction and text representation. Section 3 provides the architecture of our proposed CNNRank model in detail, and Section 4 discusses the effects of this model through various types of comparison experiments. At last, Section 5 summarizes this paper and puts forward reasonable assumptions for future research.

2. RELATED WORK

The proposed research is closely related to keyphrase extraction and text representation, In view of this, this part mainly introduces the research trends of these two aspects.

2.1 Keyphrase Extraction

Keyphrase extraction emphasizes finding several phrases to represent the document. There are mainly two groups of keyphrase extraction methods: unsupervised and supervised. Unsupervised methods usually assign a score to each candidate phrase by considering various features and then ranking them to extract keyphrases. Most such existing methods are graph-based [13, 14, 15]. TextRank [16] is the first graph-based keyphrase extraction method proposed in 2004. The authors build a graph to represent the document in which each word is a node and an edge is added if two words co-occur in a window. Then, the page-rank algorithm is used to score each word. After that, many graph-based methods are proposed focusing on three improving aspects: 1) substituting candidate phrases for single words; 2) assigning weight values of the edges to represent the semantic relatedness; 3) changing the initial scores of the nodes according to their positions in the document. Considering the fact that there may be several topics in a document, topic models are applied in the graph-based methods to improve the diversity of keyphrases

extracted [17, 18]. Liu et al. [19] proposed a Topical PageRank that runs biased PageRank for each topic separately according to the relevance of words to the corresponding topic. Bougouin et al. [20] clustered candidate keyphrases into topics and used them as vertices to build the graph. Sterckx et al. [21] proposed a fast method for Topical PageRank that calculated the cosine similarity of word-topic distribution and document-topic distribution as the topic preference. Bennani-Smires et al. [6] proposed EmbedRank which scores candidate phrases by calculating the distance between the embedding of a candidate phrase and that of the full document. The experimental results showed that EmbedRank outperforms graph-based keyphrase extraction methods. Chi et al. [22] proposed ISKE algorithm which updates the score ranking of words by iterating the scores of sentences in a graph based on the semantic relationship between adjacent sentences. In addition, Aman et al. proposed the KP-Rank [23] algorithm based on the idea of topic clustering, thinking that keywords should represent the core topic of a document. Firstly, feature extraction of candidate words is carried out through the Latent Semantic Analysis technology to construct semantic space. After that, topic clustering and ranking of candidate words are performed. Zhu et al. [24] proposed an unsupervised keyphrase extraction algorithm called SAKE, which is based on the graph model and self-attention mechanism, and can solve the problem that the Transformer cannot capture semantic relationships between distant words. Hamada et al. [25] proposed an unsupervised framework called MultPAX, which uses the pre-trained language model and knowledge graph idea, regarding keyphrase Extraction as two subtasks: Present Keyphrase Extraction and Absent Keyphrase Generation, and the advanced nature of this algorithm framework is proved by comparison experiments.

Traditional supervised methods viewed the task of keyphrase extraction as a binary classification problem, where various textural features of phrases, such as TF-IDF, position in the document, part-of-speech, and candidate phrase length, are collected to train a keyphrase extraction model. There are three disadvantages of traditional supervised keyphrase extraction methods. Firstly, the features used in the classification model are mainly statistical that ignore the semantic meaning of the words. Secondly, each candidate word or phrase is classified independently of all others that do not consider the semantic relatedness among phrases. Thirdly, the document order information is also not considered. Another supervised approach treats keyphrase extraction as a sequence labeling task that overcomes the above limitations. CRF (Conditional Random Fields) is the first model to label keyphrases [26] where the authors explored the textural features from various unsupervised keyphrase extraction approaches to train the tagger. Most recent research has shown that deep-learning models including LSTM(Long short-term memory), CNN (Convolutional Neural Network), and Transformer can significantly boost performances on keyphrase extraction. Al-Zaidy et al. proposed a Bi-LSTM-CRF model that combines a word embedding layer to represent words, a Bi-LSTM(Bi-directional Long Short-Term Memory) layer to model sequential data, and a CRF layer to model the dependencies in the output labels [27]. Sahrawat et al. [28] explored different contextualized embedding techniques including ELMo, OpenAI GPT, BERT where BiLSTM-CRF is also adopted as the neural network model architecture. Xiong et al. [29] formulate keyphrase extraction as an n-gram chunk tagging task in which CNN and transformer layers are used to model n-grams. Sun et al. [30] proposed JointKPE where a multi-task training architecture is designed to capture both local phrases and global informativeness.

2.2 Text Representation

Text representation is the basis of most NLP tasks that aim to embed text units, such as words, sentences, or documents into semantic vectors. In this section, we briefly review text representation techniques including fixed word embeddings and contextual embeddings. Text representation is the basis of most NLP tasks that aim to embed text units, such as words, sentences, or documents into semantic vectors. In this section, we briefly review text representation techniques including fixed word embeddings and contextual embeddings.

Word2Vec [31] is a very notable neural network model trained on large amounts of raw text data to obtain dense semantic representations of words. There are two different training strategies: CBOW(Continuous Bag-of-Words) and Skip-gram. Given a sliding window, the object of CBOW is to predict the central word through the context words while Skip-gram predicts the context words with the input of the central word. Furthermore, Pennington et al. [32] proposed the GloVe model that combines the advantages of global matrix factorization and local context window methods. Pagliardini et al. [33] proposed Sent2vec for learning sentence embeddings that extend the CBOW model to a larger sentence context by averaging the n-gram embeddings. CNN models capture the means of the words in the sentence by aggregating the information from the neighbors. When more CNNs are stacked, the receptive field sizes are also increased. The recurrent models, such as LSTM and GRU(Gate Recurrent Unit) [34], collects the context information of words due to the sequential property.

The embedding vectors trained by the above models like Word2Vec, Glove, and Sent2Vec are fixed. However, the meaning of a word may change used in different contexts. For example, the word 'apple' refers to a kind of fruit in the sentence "I like to eat an apple" while it means the Apple corporation in the sentence "I bought a new Apple notebook". Therefore, dynamic contextual embedding techniques are proposed. There are three well-known dynamic word embedding models: ELMo (Embeddings from Language Models) [35], GPT (Generative Pre-Training) [36], and BERT (Bidirectional Encoder Representation from Transformers) [37]. ELMo [35] is a pre-trained language model that uses stacked bi-directional LSTMs combined with character-level CNNs to represent word embedding vectors. Given a sentence of N tokens, $(t_1, t_2, \ldots, t_N)$, the forward LSTM computes the probability of token t_k given the left context $(t_1, t_2, \ldots, t_{k-1})$ while the backward LSTM predicts the probability of the token given the future context. Once the language model is pre-trained, the embedding vector of a token is further fine-tuned according to the downstream tasks. The training procedure of GPT [36] has two stages. Firstly, a large corpus of unlabeled text is used to learn the initial parameters of the model through unsupervised training. Subsequently, the parameters are fine-tuned in the downstream tasks using annotated training data. In GPT, the transformer network is adopted to learn the language model that can capture longer-range linguistic information than LSTM models. BERT [37] is a bidirectional transformer model in which the objectivity is to predict the masked token according to its context. Although the transformer-based model has the potential of learning long-range dependences, the length of the text is limited due to the full attention mechanism. The self-attention operation in the transformers scales quadratically with the sequence length. Therefore, the variants of transformers are proposed to address this limitation, such as Transformer-XL [38],

Longformer [39], BigBird [40], compressive transformer [41], et al. In BigBird and Longformer, the sparse attention mechanisms are used to reduce the quadratic complexity to linear. On the other hand, in Transformer-XL [38], the authors introduce the notion of recurrence into the self-attention network that enables passing information between segments and solves the context fragmentation problem.

The above deep learning methods based on supervised learning tasks require additional training costs to improve the effectiveness of most models. The initial appearance of contrastive learning is to realize the improvement of the model effect with a low-cost scheme without increasing sample data and reasoning computing power [42, 43, 44]. In recent years, in the field of NLP, the representative methods are the unsupervised methods ConSERT [45] and SimCSE [46], and the supervised method R-Drop [47]. Among them, ConSERT uses a variety of data feature enhancement methods to build positive example samples, such as cutting part of features, scrambling word order, and randomly dropping some feature samples; SimCSE constructs positive samples by twice Dropout for model training; R-Drop loss function uses KL divergence instead of contrastive loss.

In this work, we proposed a novel supervised learning architecture for keyphrase extraction that fully utilizes the contextual embedding vectors of n-grams and the document. In past research, unsupervised methods generally directly use the semantic feature information of the entire document without training, while supervised methods rely more on the contextual semantic information of words. To the best of our knowledge, compared to other supervised methods, this is the first work that incorporates both local features of words and global features of documents into the training process for sorting tasks. We also present a thorough comparison of the performance of different text representation methods.

3. METHODOLOGY

In this section, we formulate keyphrase extraction as a candidate phrase ranking task, and the architecture of the CNNRank model and the specific process of keyphrase extraction with this model are introduced in detail. In order to use the semantic information of phrases and documents more effectively in the algorithm, we draw on the core idea of contrastive learning, that is, after training, similar samples are in the closer region of the feature space, while dissimilar samples are in the non-adjacent region. Since the phrases that are closer to the document semantics are more likely to become keyphrases, the training goal of our proposed algorithm is that the candidate phrases whose semantics are closer to the document semantics will have higher scores.

3.1 Problem Formulation

Keyphrase extraction can be formulated as a candidate phrase ranking task as follows: Given an input document, calculate the score of each candidate phrase and select the top M phrases as the final keyphrases. To accomplish this task, we propose a supervised method called CNNRank based on a deep learning network, As shown in Figure 1, which includes three important links: (1) We select candidates for a given document, model these candidates, and calculate their vector representation in the feature

space (Section 3.2). (2) We model the document and present its vector representation in the feature space (Section 3.3). In particular, here we use a variety of methods to calculate the vector representation of the document. (3) Finally, we fuse the modeling results of the candidates and the document and predict the score of each candidate to determine the final keyphrases (Section 3.4). It is particularly emphasized that in the process of deep learning model training, we propose a new loss function with higher efficiency.

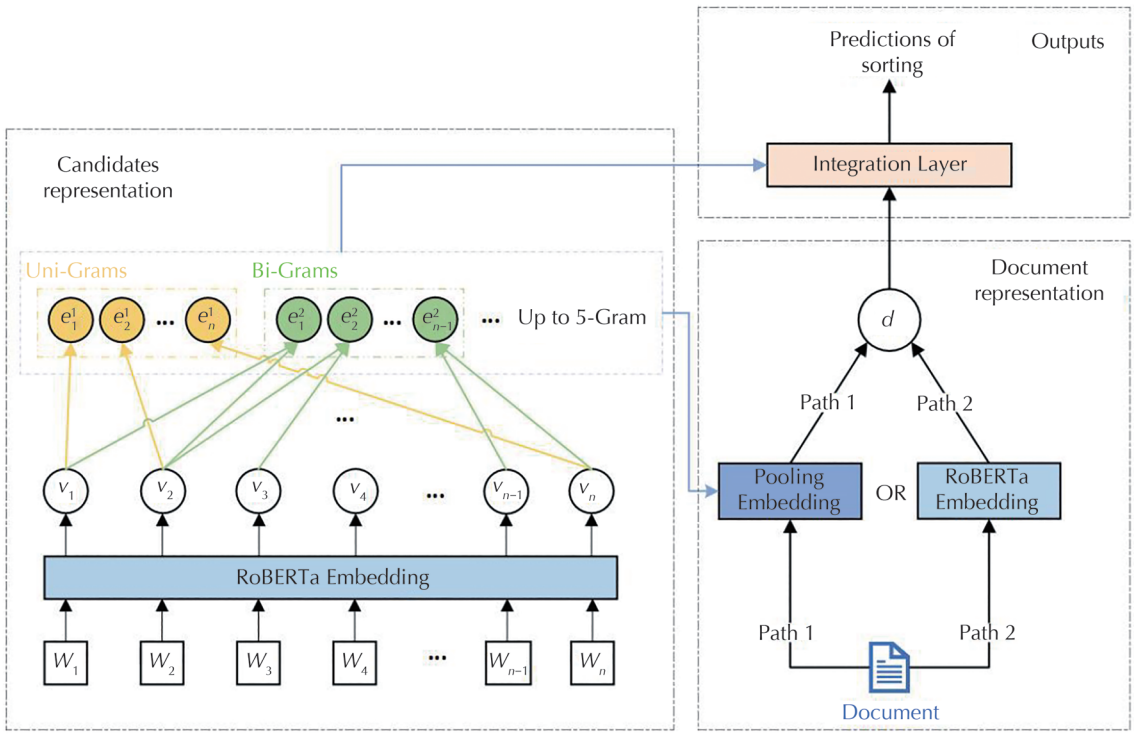


Figure 1. Proposed CNNRank architecture diagram.

3.2 Candidates Representation

This section is mainly used to complete the learning representation of the local features of candidate words. In this part, we present the details of modeling the candidate and its constituent component word embedding and CNN.

Given a document D consisting of n words, $D = \{w_1, w_2, \ldots, w_n\}$, we use a pre-trained RoBERTa model [48] to generate a k-dimensional vector representation V of all words, $V = \{v_1, v_2, \ldots, v_n\}$:

$$V = \text{RoBERTa}\{w_1, w_2, \ldots, w_n\} \tag{1}$$

We believe that the importance of a word in a document is not only related to the word itself, but also closely related to the words in several adjacent windows. Therefore, the local features of words directly affect the final keyphrase extraction results. The role of the CNN model is to effectively capture the local feature information of the words in the text. CNN uses the convolution kernel to move sequentially on the text with a certain step size, to complete the extraction of feature information in the local region. We complete the selection and vector representation of candidate words through a one-dimensional CNN model. Specifically, in the working process of one-dimensional convolution, the convolution kernel is regarded as a sliding window with a fixed size and only moves sequentially along the direction of input words. In this case, the convolution layer plays the role of feature extractor.

Drawing on the concept of the N-Gram language model in NLP, the candidates are regarded as n-gram form, that is, the sliding window operation of size N is performed on the words in sequence, to generate a sequence of byte fragments with a maximum length of N. The vector representation of the i-th candidate in each h-gram is calculated as follows:

$$e_i^h = f(conv(v_{i:i+h-1}, w) + b) \tag{2}$$

where $v_{i:i+h-1}$ represents all word vectors from the i-th to h–1 in the above word vector set V, w is the weight variable, b represents the bias item, $conv$ represents the convolution operation, the size of the convolution kernel is $h \times k$ ($1 \leqslant h \leqslant N$), and f is the activation function. The set E_h of the vector representations of all candidates in the h-gram can be calculated as follows:

$$E_h = \{e_1^h, e_2^h, \ldots, e_{n_h+1}^h\} \tag{3}$$

To sum up, one-dimensional CNN can use the vector V of the word sequence to calculate the vector E of the candidate sequence in the form of n-grams, $E = \{E_1, \ldots, E_h, \ldots, E_N\}$, where N is the maximum length of the candidate phrases.

3.3 Document Representation

In the existing supervised algorithms based on deep learning, keyphrase extraction is almost always regarded as a sequence labeling task [49, 50, 27], and it is a classification problem to judge whether each word is a keyword or not. Until 2020, in the work of Sun et al. [30], the keyphrase extraction task was regarded as a regression problem, and the experimental results proved that the extraction effect was significantly improved. However, in this study, the neural network only considered the feature information of words during the training process. In addition, in the previous unsupervised methods, there are many algorithms that directly use document information (such as topics and global semantic information) to improve the efficiency of keyword extraction. In recent years, the most typical one is EmbedRank [6]. The core idea of this work is to complete the ranking task by calculating the similarity between the candidates and the document, which directly proves the importance of document information in the extraction algorithm. Inspired by the above work, we try to integrate the global information of the document into the supervised algorithm based on the deep learning model, and directly add the document information to

the neural network training process to achieve the purpose of improving the extraction effect. Therefore, we need to model the document and map the global semantic information of the document into the same vector space as the word representation. Currently, we provide two ideas for document modeling, as shown on the right side of Figure 1 (Path1 and Path2).

3.3.1 Pooling Embedding

In the traditional graph-based method [17, 19, 21] that incorporates topic information (such as LDA), it is generally considered that a document is composed of several topics according to a certain probability, and each topic is composed of some words according to a certain probability. That is to say, the generation process of a document can be understood as: first, select a certain topic with a certain probability, and then select a certain word from the topic with a certain probability to compile a complete document. Therefore, we believe that the theme of all words affects the theme of the entire document, and the semantics of each word directly affects the semantic direction of the entire document. In this work, inspired by the functions of the pooling layer in traditional CNN, such as eliminating redundancy, feature compression, and network complexity reduction, we calculated the vector representation d of the document in the corresponding semantic space through pooling operations for the vector representation of the candidates in the k-dimension semantic space in Section 3.2. Two pooling methods, MaxPooling and AveragePooling, are adopted in the calculation process of our algorithm, and the results of these two methods are also compared in the experiment.

Method based on MaxPooling. In this algorithm, the feature vector representation obtained by the MaxPooling method is called CNNRank$_{max}$. In Section 3.2, we have obtained the feature representation E of all candidates in the document in the k-dimensional vector space, which can be understood as the mapping of phrases into k high-dimensional Spaces representing different topics or different semantics. The idea of CNNRank$_{max}$ is that for any dimension in the k-dimensional vector space, select the one with the largest eigenvalue among all candidates in this space dimension, and the maximum value is considered to best represent the semantic information of the document in this vector space. That is, the maximum value of the feature points in the neighborhood is represented as the feature vector d_{max} of the document in this space:

$$d_{max} = max_pool1d(E) \tag{4}$$

where max_pool1d is the MaxPooling function, note that the vector space dimension of d_{max} in the calculation is k-dimensional like E.

Method based on AveragePooling. In this algorithm, the feature vector representation obtained by the AveragePooling method is called CNNRank$_{avg}$. The idea of CNNRank$_{avg}$ is to use the calculation method of averaging the feature points in the neighborhood for each semantic feature space. That is, for any dimension in the k-dimensional space, the average value of all candidate vectors in this dimension space is calculated as the vector representation d_{avg} of the document in this space:

$$d_{avg} = avg_pool1d(E) \tag{5}$$

3.3.2 RoBERTa Embedding

When calculating the document vector with the above-mentioned pooling method, the embedding vectors of all n-gram candidates need to be calculated in advance. Therefore, for the purpose of reducing computational complexity, we consider computing the document vector while vectorizing words with RoBERTa in Section 3.2. That is, the RoBERTa model will generate a pooler output in the process of word vectorization, and the pooler output is the vector corresponding to the [CLS] mark in the document from the RoBERTa model. This vector incorporates the information of all words and is sufficient to represent the vector representation of the entire document. The calculation method that integrates this document representation in our algorithm is called CNNRank$_{RoBERTa}$. The experiment in Section 4.7 will prove that this simple method can reduce the negative impact of the information redundancy of irrelevant words on the extraction results.

3.4 Keyphrase Prediction

The task of the last stage is, as shown in the outputs module on the upper right of Figure 1, we fuse candidate information and document information, regard keyphrase extraction as a sorting task, then score each candidate, and Select the top M candidates as keyphrases.

In the deep learning network training process of CNNRank method, the purpose of training is to make the candidate vector with more similar semantics to the document closer to the document vector in the same vector dimension space. On the other hand, the candidate with greater semantic gap to the document will be farther away from the document vector in the training process. Therefore, in order to achieve this goal, we did not adopt the common concatenation operation in previous literature when integrating candidate word information and document information, but adopted the inner product method to calculate the initial score of each candidate. The formula for calculating the score value of each n-gram candidate is as follows:

$$s(e_i^h) = inner[e_i^h, d] \tag{6}$$

where *inner* is to do the inner product operation on the vector of the document and the candidate word, e_i^h represents the vector of the *i*-th candidate in the *h*-gram calculated in Section 3.2, *d* represents the document vector calculated in Section 3.3.

It should be considered that the same candidate may appear in different places in the document, and the different contextual semantic relationships of the candidate in different places will result in different score values. In this algorithm, for candidates that appear repeatedly, we choose the one with the highest score as the score value of this candidate. Suppose that in the *h*-gram, assuming the *i*-th candidate also appears at the (*i*+*a*)-th place, the (*i*+*b*)-th place, and the (*i*+*c*)-th place, select the one with the highest score as the score of this candidate:

$$s_{max}(e_i^h) = max\{s(e_{i+a}^h), s(e_{i+b}^h), s(e_{i+c}^h)\} \tag{7}$$

where *max* means take the maximum value operation, $e(\cdot)$ represents the vector of the same phrase in different positions, and s is the score calculated by Eq. 6.

After obtaining the score of each candidate word, The whole neural network model is trained as a sorting problem using the pairwise ranking loss:

$$L = \sum_{p^+, p^- \in D} max(0, 1 - s_{max}(p^+) + s_{max}(p^-)) \tag{8}$$

where p^+ represents the positive set (keyphrases) in document D, p^- represents the negative set (non-keyphrases) in document D, the ranking loss L will make the score of the positive label higher than the score of the negative label, that is, as the training progresses, the score of the positive example is getting higher, and the score of the negative label is getting lower.

Finally, when the training is over, the top M candidates are selected as the final keyphrases. The details of Training Process of CNNRank are shown in Algorithm 1.

Algorithm 1 Training Process of our proposed model CNNRank

Input: Training set S, mini-batch size m, iteration t, number of epoch e.
Pre-Processing: Preprocess the corpus, remove redundant fields, and generate location information for marked keyphrases.
for epoch in range (0 , e)
  for mini-batch in Training set S
    for each document D
      $\{w_1, w_2, ..., w_n\} \leftarrow$ D Word Segmentation
      $V \leftarrow \{w_1, w_2, ..., w_n\}$ All words vectorization using RoBERTa
      $E \leftarrow V$ Calculate the vector representation of each candidate by Eq.(3)
      $d \leftarrow$ D Calculate the document vector using either Eq. (4), Eq. (5) or Eq. (6)
      $s \leftarrow inner(e, d)$ Calculate the score for each candidate using Eq.(7)
      $s_{max} \leftarrow s$ Choose the highest score of the same candidate
      $L \leftarrow s_{max}$ Calculate the ranking loss L using the pairwise ranking loss
    end for
  Optimize the model, With the Adam update rule, minimize the pairwise ranking loss L
  end for
end for
return top M keyphrases

The innovation of the loss function in this section will be analyzed in the experiments in Section 4.8 and accompanied by a description of the model optimization speed.

4. EXPERIMENTAL DETAILS

To fully evaluate the effectiveness of the three algorithms (CNNRank$_{RoBERTa}$, CNNRank$_{avg}$, CNNRank$_{max}$) proposed in this paper, this part describes the experimental details, including benchmark datasets and baselines, evaluation metrics, experimental settings, and analyzes the experimental results in detail.

4.1 Datasets and Evaluation Metrics

We evaluate our algorithms on three public benchmark datasets, including Inspec [51], KP20k [52], and OpenKP [29]. Inspec is a scientific keyphrase extraction dataset with 2000 English short documents of scientific journal abstracts, which is divided into training sets (1000 abstracts), validation sets (500), and testing sets (500). KP20k is also a scientific keyphrase extraction dataset containing over 560,000 scientific articles provided by numerous online digital libraries. We follow the split principle of the original worker to divide it into training sets (527,830 documents), validation sets (20K), and testing sets (20K). The third dataset, OpenKP, is nearly 150K web documents searched from the Bing search engine. 134K, 6.6K, and 6.6K documents are a partition of its training sets, validation sets, and testing sets respectively. The detailed statistics of each testing set are provided in Table 1.

Table 1. Statistics of three benchmark datasets. Columns are: number of documents; average length of documents; average number of keyphrases manually tagged per document; average number of words in the keyphrase.

Datasets	Num of doc	Avg Len of doc	Avg Num of kp	Avg Len of kp
Inspec	500	134	9.8	2.5
KP20k	20K	147	5.3	2
OpenKP	6.6K	900	1.8	2

In the experiment, common evaluation criteria Precision (P), Recall (R) and F_1 (comprehensive evaluation index) of the keyphrase extraction method were adopted for evaluating the performance of our algorithm. In addition, since the number of keyphrases correctly annotated manually in the documents of the three corpora is different, the setting of the number of keyphrases predicted by our algorithm for each corpus is also different in the experiment. Specifically, the top 1, 3, and 5 keyphrases are selected for OpenKP, and the top 1, 3, 5, and 10 keyphrases are selected for Inspec and KP20k.

4.2 Baselines

From the three categories of keyphrase extraction methods mentioned in Section 2.1, namely the traditional supervised method, unsupervised method, and deep learning-based method, we selected eleven state-of-the-art algorithms as the baseline method and compared them against with CNNRank.

- **Kea** [53], a traditional supervised method, selects keyphrases by calculating the tf-idf features of words and a naive Bayesian classifier.
- **CopyRNN** [52], a supervised method based on deep learning, uses an RNN-based encoder-decoder network and copying mechanism to extract keyphrases.
- **Bi-LSTM** [27], a supervised method based on deep learning, uses Bi-LSTM neural network and sequence labeling method to extract keyphrases.
- **Bi-LSTM-CRF** [27], a supervised method based on deep learning, adopts the ideas of Bi-LSTM and CRF models.
- **JointKPE** [30], a supervised method based on deep learning, adopts a multi-task training architecture during model training. In the comparison experiment, we selected RankKPE and jointKPE in the original text for comparison.
- **TF-IDF** [54], a traditional unsupervised method based on statistics, needs to calculate the word frequency features of words in documents.
- **TextRank** [16], a traditional graph-based unsupervised method, establishes a graph model by calculating the adjacent relationship between words, and uses the PageRank algorithm to calculate the score of candidate words.
- **TopicRank** [20], a traditional unsupervised method based on the graph and topics, uses topic clusters to build the graph model.
- **SingleRank** [55], a traditional graph-based unsupervised method, only selects nouns or adjectives as graph vertices, and the weight calculation of edges depends on the distance between words in the window.
- **PositionRank** [13], a traditional unsupervised method based on a graph, adds the position information of the word in the document when calculating the graph node score.
- **EmbedRank** [6], an unsupervised method based on deep learning, uses embedding technology to add semantic information to the extraction algorithm.
- **ISKE** [22], an unsupervised graph-based approach, incorporates semantic relationships between adjacent sentences into word scoring strategies.
- **KP-Rank** [23], an unsupervised method, uses LSA and clustering techniques for topic-level semantic modeling of candidate words.
- **SAKE** [24], an unsupervised method, incorporates the self-attention mechanism into the traditional graph model.
- **MultPAX** [25], an unsupervised keyphrase extraction framework, uses BERT-Encoder and knowledge graphs (DBpedia and BabelNet) to calculate the similarity of keyphrases to documents.
- **SOTA** and **FBPrompt** [56], a supervised method based on LLM methods, adopts the techniques of SOTA.

4.3 Implementation Details

We implemented the deep learning-based approaches in PyTorch, and the non-deep learning approaches using the pke toolkit [57] and StanfordCoreNLP [58]. The training process used the A100 GPU with 128G RAM and took about 1 hour, 60 hours, and 140 hours on Inspec, OpenKP, and KP20k,

respectively. The word initial vectorization is accomplished using the pre-trained model "roberta-base". The optimizer used for all our three methods for CNNRank is Adam with a 5e-5 learning rate and 0.1 dropout rate. Like previous work [29, 30], the maximum length of n-grams is set to 5. We use Python for all code writing.

4.4 Experiments on the OpenKP dataset

In this section, we mainly compare CNNRank with its RankKPE method, which is closer in thought. Since the experimental link set up in the RankKPE original paper [30] was carried out on the OpenKP dataset, we also conducted comparative experiments on this dataset. Since the average number of pre-marked correct keyphrases for each document in OpenKP statistics in Table 1 is 1.8, we set the maximum number of different keyphrases extracted by the algorithm to 5, and the experimental results are shown in Table 2.

Table 2. Observe the comparative experimental results of CNNRank and RankKPE in the OpenKP dataset.

Model	F_1%@1	F_1%@3	F_1%@5
RankKPE	38.90	39.29	**33.57**
CNNRank_{max}	36.63	38.28	32.97
CNNRank_{avg}	38.29	39.30	33.48
$\text{CNNRank}_{RoBERTa}$	**38.94**	**39.45**	33.49

In Table 2, when the number of keyphrases extracted is set to 1 and 3, the F_1 value of CNNRank exceeds RankKPE. Since the biggest difference between the above two algorithms is that CNNRank incorporates document feature information during the training process, the above experimental results fully prove the positive impact of document information on the algorithm. However, when the number of keyphrases extracted is 5, the F_1 value of CNNRank decreases, which may be because the number of correctly labeled phrases in the OpenKP document is less than 3. Therefore, when extracting more than this number of keyphrases, extracting keyphrases that are closer to the document information will produce a redundant effect, resulting in a decrease in the F_1 value.

4.5 Experiments on the KP20k dataset

On the KP20k dataset, we compared CNNRank with 12 baseline methods. Since the number of keyphrases correctly labeled for each document in the KP20k corpus ranges from 5 to 10, in order to achieve the fairness of the experimental comparison results, all comparison methods in this dataset are compared when the number of keyphrases extracted is 10. Table 3 records the experimental results in detail. From the results in Table 3, it can be found that the keyphrase extraction effect of the CNNRank_{avg} and $\text{CNNRank}_{RoBERTa}$ algorithms proposed in this paper is better than other baseline methods. Specifically, compared with the four methods based on graphs, the CNNRank's F_1 value is increased by 24.33% at

the highest and 22.85% at the lowest. Compared with the six methods based on deep learning, the F_1 value has increased by 22.29% at the highest and 0.08% at the lowest. Compared with the traditional unsupervised method TF-IDF, the F_1 value of the three algorithms of CNNRank increased by 23.93% at the highest and 23.56% at the lowest. Compared with the traditional supervised method Kea, the F_1 value of the three algorithms of CNNRank increased by 20.29% at the highest and 19.92% at the lowest.

Table 3. The results of comparison experiment between CNNRank and 10 baseline methods on the KP20k dataset.

Methods	KP20k(F_1%@10)
Kea	14.0
TF-IDF	10.36
TextRank	9.96
TopicRank	10.03
SingleRank	10.1
PositionRank	11.07
EmbedRank	12.0
CopyRNN	25.4
Bi-LSTM	19.09
Bi-LSTM-CRF	35.63
jointKPE	34.15
RankKPE	34.21
CNNRank_{max}	33.92
CNNRank_{avg}	34.29
$\text{CNNRank}_{RoBERTa}$	34.29

From the above experimental data, we have the following two points of inspiration: First, the effect of CNNRank_{avg} and $\text{CNNRank}_{RoBERTa}$ algorithms is significantly improved compared with the two algorithms of JointKPE and other baseline methods, which is enough to prove that our algorithm has a useful positive impact on the results by fusing the feature information of documents. Second, the extraction effect of CNNRank_{max} did not meet expectations, which shows that when we add document information during the training process, the idea of only using the maximum value of each vector space as the vector representation of the document is flawed. CNNRank_{max} lacks consideration of the extensiveness and comprehensiveness of document information, which leads to a decrease in the extraction effect. The best performance among all baseline methods is the supervised method RankKPE based on deep learning. In

order to further prove the advantages of the CNNRank$_{avg}$ and CNNRank$_{RoBERTa}$ algorithms, we compared the F_1 value of the above three methods in the case of extracting different numbers of keywords, as shown in Table 4. From the experimental data, it can be clearly seen that the effect of CNNRank is better than RankKPE in any case, which directly proves the effectiveness of the algorithm idea proposed in this paper, that is, the idea of fusing document information has a positive impact on the extraction results.

Table 4. The detailed experimental results(Precision, Recall, and F_1 value) of RankKPE, CNNRank$_{avg}$, and CNNRank$_{RoBERTa}$ on the KP20k corpus, and the bold font indicates the maximum value.

Model	F_1% @1	F_1% @3	F_1% @5	F_1% @10	P% @1	P% @3	P% @5	P% @10	R% @1	R% @3	R% @5	R% @10
RankKPE	29.71	41.83	41.46	34.21	56.81	43.43	35.28	24.09	**21.91**	46.83	60.98	78.60
CNNRank$_{avg}$	29.72	41.87	**41.50**	34.29	**57.12**	43.47	35.28	24.13	21.85	46.91	**61.12**	78.77
CNNRank$_{RoBERTa}$	**29.74**	**41.99**	41.45	**34.29**	56.96	**43.62**	35.26	**24.14**	21.89	**47.02**	61.06	**78.82**

In addition, in order to further observe the algorithm effect of CNNRank, we calculated the experimental results when the number of extracted keyphrases is 1, 3, 5, and 10, as shown in Figure 2. After observation, it is found that when the number of keyphrases extracted is 5, the F_1 value reaches the peak, and when 10 keyphrases are extracted, the F_1 value decreases instead. According to previous work, each document is composed of words with different topics, and those words closest to the topic of the document are more likely to be keywords. However, each document does not contain too many topics, so the scope of the document feature information on the algorithm extraction effect is also limited, which leads to the occurrence of F_1 value rising first and then falling.

4.6 Experiments on the Inspec dataset

In the Inspec dataset, we conducted a comparison experiment between CNNRank and 10 baseline methods. According to statistics, the average number of correctly annotated keyphrases in each document in the Inspec dataset is 9.8, so we compare the experimental data with the F_1 value under the condition that the number of keyphrases extracted by each method is 10, and the experimental results are recorded in Table 5 in detail. It can be seen from the results in Table 5 that the keyphrase extraction effect of this algorithm is superior to other baseline methods. Specifically, compared with the graph-based method (TextRank, TopicRank, SingleRank, PositionRank, and ISKE), the maximum F_1 value is increased by 36.65%; compared with the deep learning-based method (EmbedRank, CopyRNN, Bi-LSTM, Bi-LSTM-CRF, SAKE, and MultPAX), the maximum F_1 value is increased by 33%; compared with the method based on SOTA technology (SOTA, and FBPrompt), the maximum F_1 value is increased by 7.39%; compared with the traditional unsupervised method TF-IDF, the F_1 value is increased by 42.78%; compared with the traditional supervised method Kea, the F_1 value is increased by 20.49%; compared with the topic clustering-based method KP-Rank, the F_1 value is increased by 15.34%.

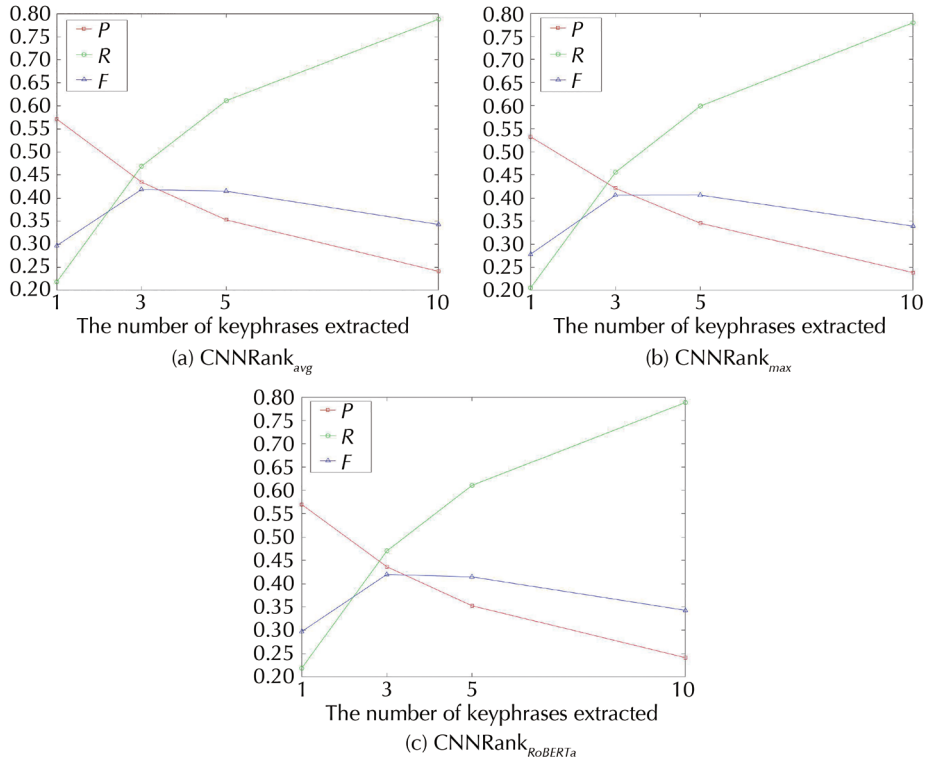


Figure 2. The Precision (*P*), Recall (*R*), and F_1 value of CNNRank in the KP20k dataset when the number of keyphrases extracted is 1, 3, 5, and 10 respectively.

The experimental data of these F_1 value growth is enough to prove the effectiveness of the algorithm CNNRank proposed in this paper. In addition, we draw the following conclusions from the experimental data of all baseline methods on the two datasets in Section 4.5 and Section 4.6: First, although the supervised method has a high algorithm complexity during the training process and needs to prepare training samples in advance, its extraction effect is generally higher than that of the unsupervised method. This is due to the fact that the supervised method obtains an optimal model through repeated training, and the model has better prediction or classification ability for unknown data. Therefore, how to train the optimal model by better modeling the text and extracting more useful feature information is exactly the research direction for us to improve the effectiveness of the algorithm. Second, the effect of the method based on the deep learning model is higher than that of the traditional baseline method. Therefore, in order to improve the effectiveness of the algorithm, most of the research work on keyphrase extraction now focuses on the supervised algorithm based on the deep learning model.

Table 5. The results of comparison experiment between CNNRank and 10 baseline methods on the Inspec dataset.

Methods	Inspec(F_1%@10)
Kea	34.5
TF-IDF	12.21
KP-Rank	39.65
TextRank	18.34
TopicRank	24.71
SingleRank	27.58
PositionRank	25.58
ISKE	37.8
EmbedRank	35.39
CopyRNN	28.2
Bi-LSTM	21.99
Bi-LSTM-CRF	44.9
SAKE	30.79
MultPAX	37.1
SOTA	47.6
FBPrompt	49.9
CNNRank_{max}	49.97
CNNRank_{avg}	53.75
$\text{CNNRank}_{RoBERTa}$	**54.99**

4.7 CNNRank_{max} VS. CNNRank_{avg} VS. $\text{CNNRank}_{RoBERTa}$

In order to compare the effects of the three text representation algorithms of CNNRank proposed in Section 3.3, we conducted comparative experiments on the three datasets. From the experimental results in Table 2 to Table 5, it can be found that on the three corpora, the order of F_1 value from high to low is $\text{CNNRank}_{RoBERTa}$, CNNRank_{avg} and CNNRank_{max}. In other words, in any experimental environment, the highest F_1 value is $\text{CNNRank}_{RoBERTa}$, and the lowest is CNNRank_{max}. This is enough to show that among the three text representation methods, $\text{CNNRank}_{RoBERTa}$ is the most accurate representation of text characteristics, while CNNRank_{max} is the worst. This may be because there are usually many topics in a

document. If CNNRank_{max} is used to calculate the vector representation of a document, the maximum value of all candidate phrase vectors in each dimension is taken. Therefore, there is a problem of one-sided representation of document information, which ultimately leads to a poor extraction effect. However, CNNRank_{avg} integrates different feature vector information of multiple candidate phrases when averaging all candidate phrase vectors, so the F_1 value is higher than that of CNNRank_{max}. The candidate phrases of the above two methods are all n-gram phrases. Although the comprehensiveness of information representation is improved in the calculation process, phrases that differ greatly from the semantic information of the document will be recalculated in multiple n-gram candidates, and there is bound to be irrelevant information redundancy. for example, in the Inspec dataset file named "1987. abstr", there is a sentence "we all appreciate the need for, ...". When using the above CNNRank_{avg} and CNNRank_{max} to calculate the maximum or average value of all n-gram word vectors, the vector representations of "we", "we all", "we all appreciate", "we all appreciate the", and "we all appreciate the need" will be used multiple times. As a result, redundant phrase vectors such as "we" and "the", which are not closely related to keyphrases, will be counted repeatedly, thus negatively affecting the results. Compared with the previous two methods, the $\text{CNNRank}_{RoBERTa}$ method directly integrates the semantic information of each word, and each word is calculated only once. There is no problem with repeated calculation of redundant feature information of irrelevant words, and the extraction effect is improved by reducing the participation of words that are not closely related to the semantic information of the document.

4.8 Observation of the Training Process

An innovative point of the algorithm in this paper is that a new loss function calculation method is proposed when training the keyphrase extraction model. When the RankKPE algorithm in literature [30] calculates the score of each candidate phrase, it directly maps each n-gram through a linear layer to obtain a score. The criterion for calculating the score of each candidate in this algorithm is that the closer the candidate is to the semantics of the document, the higher the score will be. Therefore, in each Epoch, the loss function does not use the simple mapping in RankKPE to calculate the score of each candidate, but uses the result of the inner product operation between each candidate vector and the document vector as the score value. The size of the vector inner product can measure the similarity between two vectors, and the closer to the document semantics in training, the higher the score of the word will be. In the experiment, we found that CNNRank adopted this new loss function calculation method directly using the similarity between phrases and documents, which not only improved the extraction effect, but also significantly accelerated the convergence rate of the model in training than RankKPE. Taking the KP20k dataset as an example, we counted the convergence speed of the two algorithms CNNRank and RankKPE under 21 Epoch training, as shown in Figure 3. One epoch is to use all the samples in the training set to train once. From Figure 3, we find that on the KP20k dataset, RankKPE and CNNRank have the highest F_1 value and the model tends to be stable at about the 10th and 8th Epoch respectively. This experimental result also proves that our algorithm has a faster convergence speed and better effect during model training.

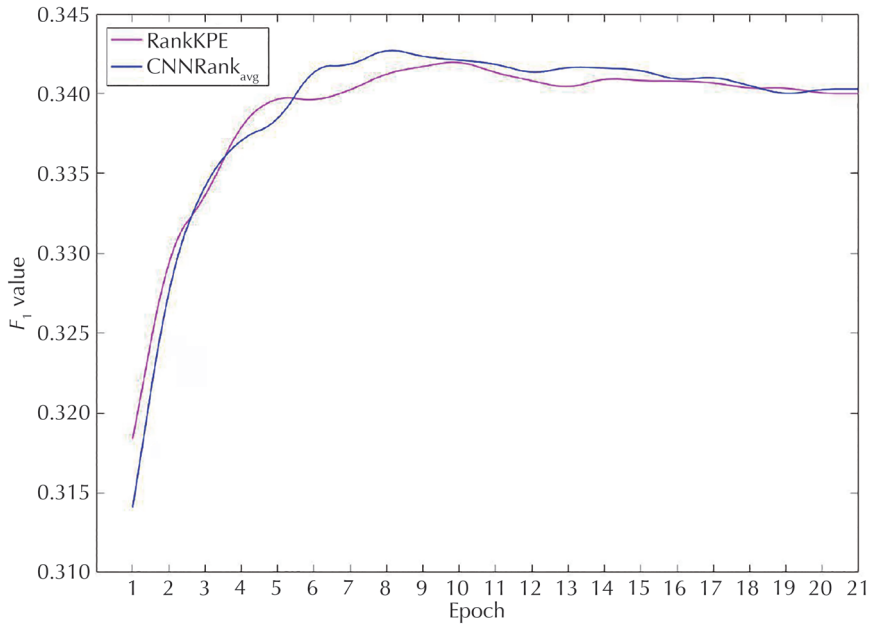


Figure 3. Comparison of model convergence rates of RankKPE and CNNRank algorithms in the training process on KP20k corpus.

5. CONCLUSION AND FUTURE WORK

In this paper, we propose a new contrastive learning framework for the keyphrase extraction task, which is a supervised method called CNNRank that fuses n-gram representations of phrases and document representations. This is the first time that the idea of contrastive learning is used in a keyphrase extraction algorithm. This method treats keyphrase extraction as a regression problem, and innovatively designs a new loss function calculation rule in the process of supervised model training, thereby improving the performance of keyphrase extraction. To be specific, after obtaining the context feature information of the phrase and the global feature information of the document, we calculated the score of each candidate phrase after the multi-feature fusion training, and the M phrases with the top score rank were selected as keyphrases. In particular, in the above process, when we calculate the semantic feature representation of the document, three different calculation methods are used. Experiments on three public corpora prove that CNNRank outperformed the existing popular methods.

In future studies, on the one hand, we will continue to study how to learn more feature information helpful to the extraction results under the current popular deep learning framework, and reduce the complexity of the algorithm as much as possible. On the other hand, we also expect to make some breakthroughs in the extraction algorithm for long texts.

ACKNOWLEDGEMENTS

This research was funded by the National Natural Science Foundation of China (No.61503116), the Special Project of Provincial Scientific Research Platform of Hefei Normal University (No.2020PT15), the Natural Science Foundation of the Anhui Higher Education Institutions of China (No.KJ2021A0902, No.2022AH052140).

AUTHOR CONTRIBUTIONS

Jing Song: writing original draft; visualization; validation.

Xian Zu: methodology; data curation; writing original draft; supervision.

Fei Xie: data curation; conceptualization; writing review and editing.

DATA AVAILABILITY

All data are available from the authors upon reasonable request.

COMPETING INTERESTS

The authors declare that they have no competing interest.

ETHICAL AND INFORMED CONSENT FOR DATA USED

We comply with all of the journal's ethical and informed consent requirements for the data used.

REFERENCES

[1] Merrouni, Z.A., Frikh B. & Ouhbi, B.: Automatic keyphrase extraction: a survey and trends: Journal of Intelligent Information Systems 54(2), 391–424 (2020)

[2] Rabby, G., Azad, S., Mahmud, M., Zamli, K.Z. & Rahman, M.M.: Teket: a tree-based unsupervised keyphrase extraction technique: Cognitive Computation 12(4), 811–833 (2020)

[3] Papagiannopoulou, E. & Tsoumakas, G.: A review of keyphrase extraction: Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 10(2), e1339 (2020)

[4] Zu, X. & Xie, F.: Keyphrase extraction using deep and wide learning features: Intelligent Data Analysis 26(6), 1643–1660 (2022)

[5] Alzaidy, R., Caragea, C. & Giles, C.L.: Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents. In: The world wide web conference, pp. 2551–2557 (2019)

[6] Kamil, B.S., Musat, C., et al.: Embedrank: Unsupervised keyphrase extraction using sentence embeddings. arXiv preprint arXiv:1801.04470 (2018)

[7] Sun, Y., Qiu, H.P., et al.: Sifrank: a new baseline for unsupervised keyphrase extraction based on pre-trained language model: IEEE Access 8: 10896–10906 (2020)
[8] Ding, H. R. & Luo, X.: Attentionrank: Unsupervised keyphrase extraction using self and cross attentions. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 1919–1928 (2021)
[9] He, K.M., Fan, H.Q., et al.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 9729–9738 (2020)
[10] Chen, T., Kornblith, S., Norouzi, M. & Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning, pp. 1597–1607 (2020)
[11] Wu, H. L., Ma, T. F., et al.: Unsupervised reference-free summary quality evaluation via contrastive learning. In: Proceedings of the 2020 Conference on Empirical Methods in Nat-ural Language Processing, pp. 3612– 3621 (2020)
[12] Luo F. L., Yang, P.C., et al.: Rethinking denoised auto-encoding in language pre-training. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 2922–2932 (2021)
[13] Florescu, C. & Caragea, C.: Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1105–1115 (2017)
[14] Biswas, S. K., Bordoloi, M., & Shreya, J.: A graph based keyword extraction model using collective node weight: Expert Systems with Applications 97, 51–59 (2018)
[15] Zu, X., Xie F., & Liu, X.: Graph-based keyphrase extraction using word and document embeddings. In: 2020 IEEE International Conference on Knowledge Graph (ICKG), pp. 70–76 (2020)
[16] Mihalcea, R. & Tarau, P.: Textrank: Bringing order into text. In: Proceedings of the 2004 conference on empirical methods in natural language processing, pp. 404–411 (2004)
[17] Boudin, F.: Unsupervised keyphrase extraction with multipartite graphs. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pp. 667–672 (2018)
[18] Li, L., Liu, J., et al.: Unsupervised keyword extraction from microblog posts via hashtags: Journal of Web Engineering 17, 93–120 (2018)
[19] Liu, Z., Huang, W., Zheng, Y. & Sun, M.: Automatic keyphrase extraction via topic decomposi-tion. In: Proceedings of the 2010 conference on empirical methods in natural language processing, pp. 366–376 (2010)
[20] Bougouin, A., Boudin, F. & Daille, B.: TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction. In: International Joint Conference on Natural Language Processing (IJCNLP), pp. 543–551 (2013)
[21] Sterckx, L., Demeester, T., Deleu, J. & Develder, C.: Topical word importance for fast keyphrase extraction. In: Proceedings of the 24th International Conference on World Wide Web, pp. 121–122 (2015)
[22] Chi, L. & Hu, L.: Iske: An unsupervised automatic keyphrase extraction approach using the iterated sentences based on graph method: Knowledge-Based Systems 223:107014 (2021)
[23] Aman, M., Abdulkadir, S. J., et al.: Kp-rank: a semantic-based unsupervised approach for keyphrase extraction from text data: MULTIMEDIA TOOLS AND APPLICATIONS 80(8), 12469–12506 (2021)
[24] Zhu, P., Gong, C. & Wei, Z.: Sake: A graph-based keyphrase extraction method using self-attention: In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligence Systems, pp. 339–350 (2022)
[25] Hamada, M., et al.: Multpax: Keyphrase extraction using language models and knowledge graphs. In: International Semantic Web Conference, pp. 303–318 (2022)

[26] Gollapalli, S. D. & Li, X.: Keyphrase extraction using sequential labeling. arXiv preprint arXiv:1608.00329 (2016)
[27] Alzaidy, R., Caragea, C. & Giles, C.L.: Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents. In: The World Wide Web Conference, pp. 2551–2557 (2019)
[28] Sahrawat, D., Mahata, D., et al.: Keyphrase extraction as sequence labeling using contextualized embeddings. In: Advances in Information Retrieval: 42nd European Conference on IR Research, pp. 328–335 (2020)
[29] Xiong, L., Hu, C., et al.: Open domain web keyphrase ex-traction beyond language modeling. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Lan-guage Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 5175–5184 (2019)
[30] Sun, S., Liu, Z., et al. Capturing global informativeness in open domain keyphrase extraction. In: Natural language Processing and Chinese Computing: 10th CCF International Conference, pp. 275–287 (2021)
[31] Mikolov, T., Chen, K., et al.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
[32] Pennington, J., Socher, R. & Christopher D. M.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing, pp. 1532–1543 (2014)
[33] Pagliardini, M., Gupta, P. & Jaggi, M.: Unsupervised learning of sentence embeddings using com-positional ngram features. arXiv preprint arXiv:1703.02507 (2017)
[34] Chung, J., Gulcehre, C., et al.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
[35] Matthew E. P., Mark N., Mohit I., et al.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 2227–2237 (2018)
[36] Radford, A., Narasimhan, K., et al. Improving language understanding by generative pre-training. 2018.
[37] Devlin, J., Chang, M., Lee, K. & Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of naacl-HLT, pp. 2 (2019)
[38] Dai, Z., Yang, Z., et al.: Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860 (2019)
[39] Beltagy, I., Matthew E. P. & Cohan, A.: Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150 (2020)
[40] Zaheer, M., Guruganesh, G., et al.: Big bird: Transformers for longer sequences: Advances in Neural Information Processing Systems 33, 17283–17297 (2020)
[41] Rae, J. W., Potapenko, A., et al.: Compressive transformers for long-range sequence modelling. arXiv preprint arXiv: 1911.05507(2019)
[42] Wu, Z., Xiong, Y., et al.: Unsupervised feature learning via non-parametric instance discrimination. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3733–3742 (2018)
[43] Ye, M., Zhang, X., Yuen, P. C., & Chang, S. F.: Unsupervised embedding learning via invariant and spreading instance feature. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6203–6212 (2019)
[44] He, K., Fan, H., et al.: Momentum contrast for unsupervised visual representation learning. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9726–9735 (2020)
[45] Yan, Y., Li, R., et al.: Consert: A contrastive framework for self-supervised sentence representation transfer. arXiv preprint arXiv: 2105.11741 (2021)

[46] Gao, T., Yao, X., & Chen, D.: Simcse: simple contrastive learning of sentence embeddings. arXiv preprint arXiv: 2104.08821 (2021)
[47] Wu, L. J., Li, J., et al.: R-drop: Regularized dropout for neural networks: Advances in Neural Information Processing Systems 34, 10890–10905 (2021)
[48] Liu, Y. H., Ott, M., et al.: Roberta: A robustly optimized bert pretraining approach. ArXiv preprint arXiv:1907.11692 (2019)
[49] Patel, K. & Caragea, C.: Exploring word embeddings in crf-based keyphrase extraction from research papers. In: Proceedings of the 10th International Conference on Knowledge Capture, pp. 37–44 (2019)
[50] Sahrawat, D., Mahata, D., Kulkarni, M., et al.: Keyphrase extraction from scholarly articles as sequence labeling using contextualized embeddings. arXiv preprint arXiv:1910.08840 (2019)
[51] Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge. In: Proceedings of the 2003 conference on Empirical methods in natural language processing, pp. 216–223 (2003)
[52] Meng, R., Zhao, S. Q., et al.: Deep keyphrase generation. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 582–592 (2017)
[53] Witten, I. H., Paynter, G. W., et al.: Kea: Practical automatic keyphrase extraction. In: Proceedings of the fourth ACM conference on Digital libraries, pp. 254–255 (1999)
[54] Salton, G. & Buckley, C.: Term-weighting approaches in automatic text retrieval: Information processing & management 24(5), 513–523 (1988)
[55] Wan, X. J. & Xiao, J. G.: Single document keyphrase extraction using neighborhood knowledge. In: AAAI, pp. 855–860 (2008)
[56] Huang, Z. , Zhou, J., Xiao, G. & Cheng, G.: Enhancing in-context learning with answer feedback for multispan question answering. In: CCF International Conference on Natural Language Processing and Chinese Computing, pp. 744–756 (2023)
[57] Boudin, F.: pke: an open source python-based keyphrase extraction toolkit. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, pp. 69–73 (2016)
[58] Toutanova, K., Klein, D., et al.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, pp. 252–259 (2003)

AUTHOR BIOGRAPHY



Jing Song received her master degree from Hefei University of Technology in 2004. She is a senior engineer with Hefei Normal University. Her research interests include natural language processing.



Xian Zu received the master degree in twenty thirteen from University of Science and Technology of China. Now she is a lecturer at Hefei Normal University. Her research interests include data mining and natural language processing.



Fei Xie received his Ph.D. degree from Hefei University of Technology in 2011. He is a professor with Hefei Normal University. He is a master supervisor. His research interests include data mining and natural language processing.