Embedding-based Supervised Learning for Keyphrase Extraction

Tingyi LIU† and Mizuho IWAIHARA‡

Graduate School of Information, Production and Systems, Waseda University

2-7 Hibikino, Wakamatsu-ku, Kitakyushu-shi, Fukuoka, 808-0135 Japan

E-mail: †tyliu@akane.waseda.jp, ‡iwaihara@waseda.jp

Abstract Keyphrase extraction is the task of selecting a set of phrases that best represent a given document. Fine-tuning a pre-trained language model and applying the fine-tuned model to various NLP tasks has become a common effective method. Since this method achieves good performance, we think that performance of keyphrase extraction can be significantly improved by fine-tuning a pre-trained language model utilizing a suitable training strategy. Thus we propose two methods based on fine-tuning BERT and Sentence-BERT models, whether these models are trained by using binary cross-entropy loss and triplet loss, respectively. For fine-tuning BERT, we apply a cross-encoder architecture and train the model with a classifier that outputs a scalar value which represents the score of each candidate phrase, and keyphrases are selected by ranking by their scores. On the other hand, Sentence-BERT is fine-tuned with three inputs: anchor, positive examples and negative examples, so that the distance between anchor and positive example becomes smaller than the distance between anchor and negative examples. Our experimental results show an improvement over the compared baseline methods.

Keyword Keyphrase extraction, Supervised learning, Embedding-based, Fine-tuning

1. Introduction

Keyphrase extraction is a natural language processing task of automatically selecting a set of representative and characteristic phrases that can best describe a given document [17]. For large text collections, keyphrases provide faster and more accurate searches and can be used as concise summaries of documents, allowing people to quickly obtain key information about a document in a short time. In recent years, with the continuous development of the Internet, a large volume of online documents, such as research papers and news articles have accumulated. Due to its clarity and practical importance, keyphrase extraction has been a core technology for information retrieval and document classification. For this task, unsupervised methods have played an important role, because of corpus-independence and search efficiency [2] [14] [20]. However, compared with supervised methods, unsupervised methods only use the information that comes from the target document and the statistical information from the dataset. As a result, the performance improvement of unsupervised methods is hindered by the lack of context information of keyphrases [2].

In this paper, we consider supervised keyphrase extraction based on fine-tuning pre-trained language models toward keyphrase extraction and propose an embedding-based method called *KeyRank*, which combines the information of the datasets and fine-tuning of pre-trained language models. We consider two pre-trained language models, (a) BERT [6] and (b) Sentence-BERT [18]. For (a) the BERT-based method, target documents and candidate phrases are concatenated to a sequence and embedded as a vector. For (b) the Sentence-BERT-based method, target documents and candidate phrases are embedded in the same low-dimensional space. For fine-tuning, the objective function to train the model is selected as: binary cross-entropy loss for (a) and triplet loss with Euclidean distance for (b). After fine-tuning, keyphrases will get higher scores than non-keyphrases for a given document. Our methods outperform the compared baseline methods in terms of F1@K and show a better result than models without fine-tuning on three or two datasets. Moreover, to reduce the redundant keyphrases in the results, we consider the diversity between selected keyphrases by using MMR [5] and report better performance than other compared baseline methods which also consider diversity.

The rest of this paper is organized as follows. Related work on keyphrase extraction and language models is presented in Section 2. In Section 3 we show the process of our method and fine-tuning strategy. Then the details of the experiment and the final results are shown in Section 4. Section 5 summarizes the results.

2. Related work

There exists a number of noteworthy methods for keyphrase extraction, such as statistics-based methods, graph-based methods, and embedding-based methods. Since our method is based on embeddings of documents and phrases, pre-trained language models are also discussed in this section.

2.1 Keyphrase extraction method

The most common baseline on keyword extraction task is based on statistics that uses term frequency (TF) and inverse document frequency (IDF) to score and rank phrases according to the product of TF and IDF [17]. This is a simple method that is believed that the most meaningful phrases for a document should be the phrases that appear frequently in the given document but appear less frequently in other documents in the entire document collection. Although TF-IDF has been widely used due to its effectiveness, it is irrespective of the contextual information of phrases. KP-Miner [7] is a keyphrase extraction system that considers various types of statistical information beyond TF-IDF. For ranking candidate phrases, their method uses a scoring function that is a variant of TF-IDF. Another method is YAKE [4] which considers both statistics and context information and achieves a remarkable performance. Besides the position and the frequency of a term, it also uses new statistical metrics that capture context information and the spread of the terms within the document.

Another type of method is graph-based methods that exploit graph structure for this task and achieve great performance. TextRank [14] is the first graph-based keyphrase extraction method which borrows the key idea of PageRank [3]. TextRank uses POS tags to obtain candidates, creates an undirected and unweighted graph in which the candidates are added as nodes and an edge is added between nodes that co-occur within a window of N words. Then the PageRank algorithm runs until it converges. SingleRank [20] is an extension of TextRank which introduces weights to edges in the graph. In this method, the weight of the nodes is the number of co-occurrences. These graph-based methods consider the relative positions between words and capture more contextual information, but the result is affected by the tokenizer which directly affects the final results.

Since low-dimension representation of words has been proposed, embedding-based methods have played an important role in keyphrase extraction and show brilliant performance. The representative method is EmbedRank [2] which extracts candidate phrases from a given document based on POS tags. Then EmbedRank uses two different sentence embedding methods (Sent2vec [16] and Doc2vec [11]) to represent the candidate phrases and the document in the same high-dimensional vector space. Then it ranks the candidate phrases using the normalized cosine similarity between the embeddings of the candidate phrases and the document embedding, and selects top-*N* as the result. Moreover, to reduce redundancy, EmbedRank++ promotes relevancy and diversity of keyphrase distribution using Maximal Marginal Relevance [5].

2.2 Language models

Word embedding and sentence embedding play an extremely important role in the field of natural language processing. Word2vec [15] is a classic method for training word embeddings. Word2vec consists of two main models: Skip-gram model and Continuous Bag-of-Words (CBOW) model. The CBOW architecture predicts the current word based on the context, while Skip-gram predicts surrounding words given the current word.

Sent2vec [16] is similar to Word2vec and uses word n-gram features to produce sentence embeddings. Doc2vec [11] represents each document by a dense vector which is trained to predict words in the document. Doc2vec overcomes the weakness of the CBOW model and shows a great performance.

BERT [6] is a recent pre-trained model which is pre-trained with two tasks: Masked language model and next sentence prediction, surpassing the state-of-the-art results of many NLP tasks. Sentence-BERT [18] is adding a pooling operation to BERT to generated fixed-dimension vectors, allowing to use cosine distance and other various loss functions.

Recently, keyphrase extraction is modeled as a sequence labeling task and have made certain progress. Alzaidy et al. [1] follow this idea and use Bi-LSTM-CRF to capture the information in the document and predict a label for each token position. Y. Lim et al. [12] combine BERT into this sequence labeling task, adding a bi-directional layer and fully connected layer to the final BERT vector which represents the embedding of each token to fine-tune BERT for keyphrase extraction. These methods bring an improvement to keyphrase extraction task than classical methods, however the cost of training time is increasing.

3. Methodology

For keyphrase extraction task, unsupervised methods (e.g. TextRank, EmbedRank) are valued because of the simplicity and corpus independence, but at the same time, the performances of these methods have obvious limitations which is the weak point of unsupervised methods. Supervised methods which often consider sequence labeling task can overcome this disadvantage,

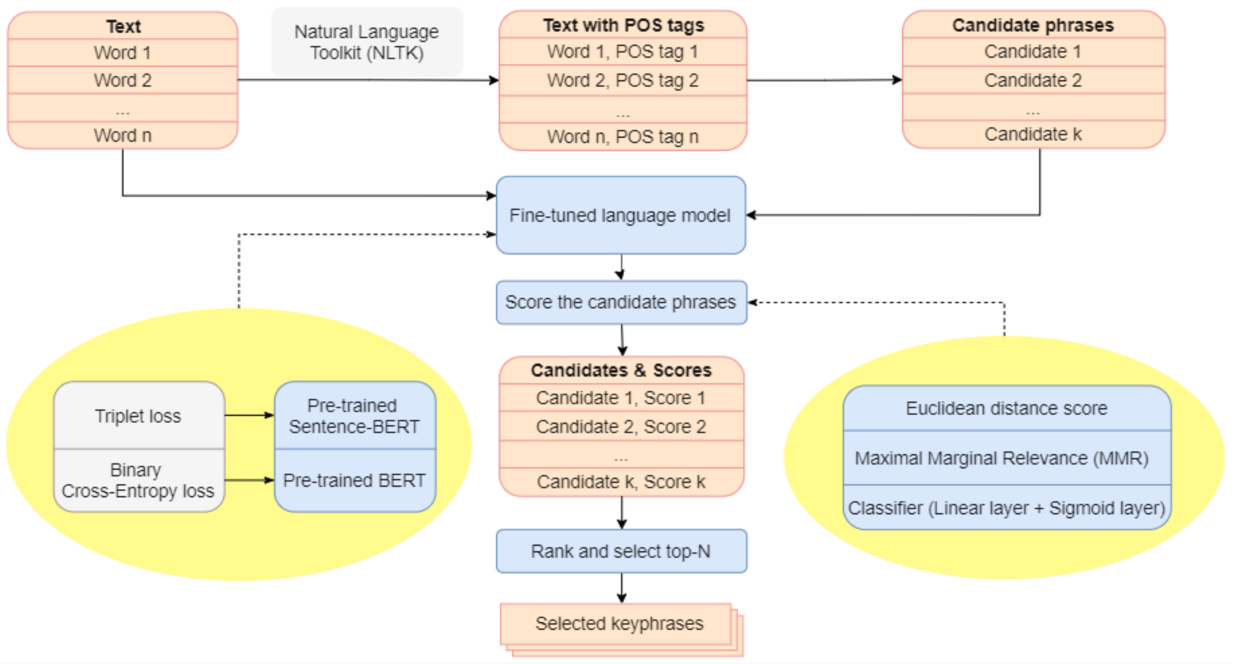

Figure 1: Flowchart of KeyRank keyphrase extraction. (a) KeyRank BERT fine-tunes BERT by binary cross-entropy loss and scores the candidate phrases by the classifier. (b) KeyRank Sentence-BERT fine-tunes Sentence-BERT by triplet loss and scores the candidate phrases by Euclidean distance and MMR.

but the model becomes complex and requires more training time. Moreover, sequence labeling task predicts the label of each token, whether it is the first token of the keyphrases, or whether it is a token after the first token in a keyphrase, and check whether the label is true, which requires such manually labeled datasets, bringing considerable costs. Thus we propose a low-cost embedding-based supervised method for keyphrase extraction, called *KeyRank*, which utilizes part-of-speech (POS) tags and fine-tuning of language models and achieves better results than state-of-the-art unsupervised methods.

We also consider Maximal Marginal Relevance (MMR) in order to combine accuracy with diversity and reduce redundant keyphrases. Figure 1 shows the flowchart of KeyRank.

There are three main steps in our method as following:

(1) We select candidate phrases which are noun phrases from a given free text document based on POS tags.

(2) (a) KeyRank BERT fine-tunes the pre-trained BERT model by binary cross-entropy loss. (b) KeyRank Sentence-BERT fine-tunes the Sentence-BERT model by triplet loss. Then these fine-tuned models generate document embeddings, which are used to score candidate phrases in the next stage.

(3) We rank the candidate phrases by a scoring function, to select the top-*N* phrases as the keyphrases.

3.1 Candidate phrases selection

For a free-style document, keyphrases are usually noun phrases that consist of zero or more adjectives followed by one or more nouns (e.g. *fuzzy system, supervised learning, word embedding*). As a result, we use POS tags to obtain qualified phrases as candidate phrases that are not allowed to end with adjectives, verbs, or adverbs, etc. The final keyphrases are selected from these candidate phrases by a scoring method. For instance, in the following sentence: "*Fuzzy/JJ systems/NNS are/VBP often/RB used/VBN to/TO solve/VB complex/JJ problems/NNS*", "*fuzzy systems*" and "*complex problems*" are candidate phrases, but "*solve complex problems*" and "*used*" are not candidate phrases.

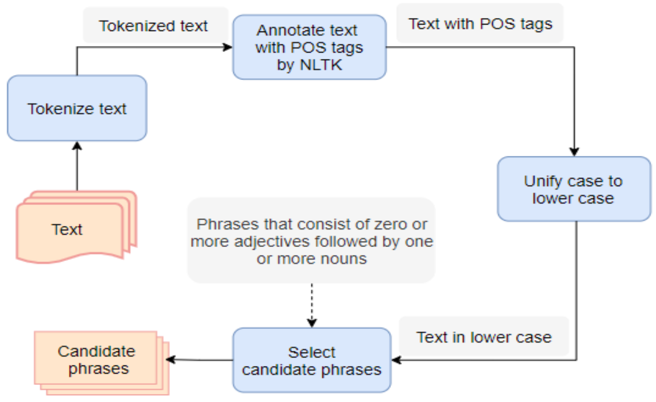

Figure 2: The process of candidate phrases selection stage

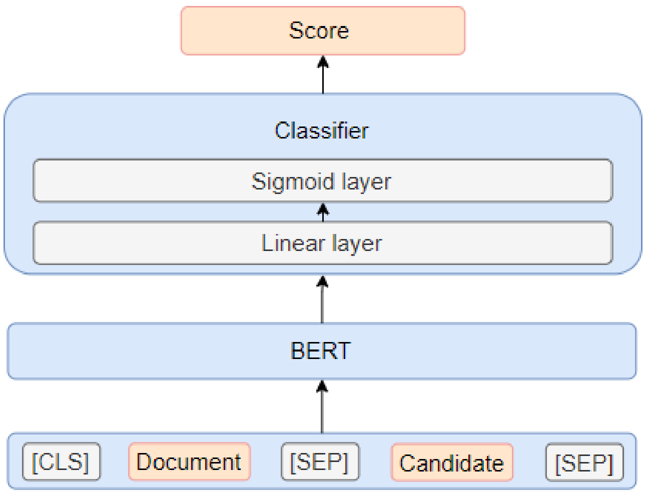

Figure 3: Architecture for fine-tuning BERT

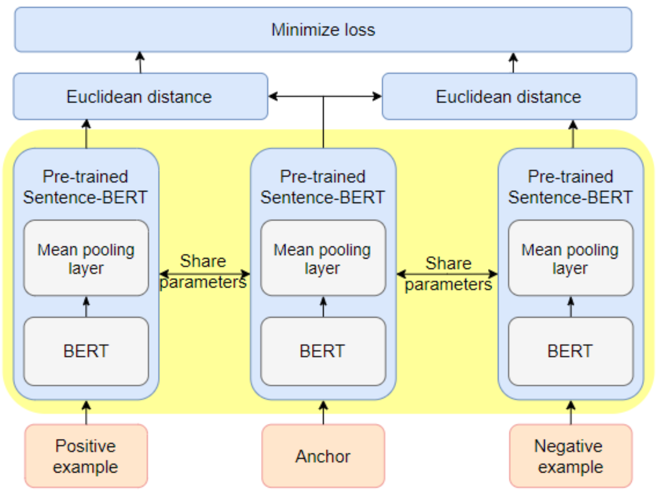

Figure 4: Architecture for fine-tuning Sentence-BERT

To obtain candidate phrases, we first tokenize the text and annotate the text with POS tags. Then we unify the case to lower case and select phrases according to the POS tags as candidate phrases, which are used in the next stage. Figure 2 shows the candidate phrase selection stage. After this stage, we can obtain a set of candidate phrases of the given document.

3.2 Language models and fine-tuning

3.2.1 BERT fine-tuning

BERT [6] is a pre-trained language model developed by Google which achieves state-of-the-art performance on a number of NLP tasks. One of the advantages of BERT is allowing fine-tuning of the model for a downstream task by a given loss function. In our case, we fine-tune BERT with two inputs: the document text and a candidate phrase. We utilize a cross-encoder [9] which performs self-attention between the document text and the candidate phrase, to capture the information between these two parts. The model reads the document text and the candidate phrase in one sequence consider the contextual information of the sequence through self-attention. Figure 3 shows our configuration for fine-tuning pre-trained BERT.

Binary cross-entropy loss: For keyphrase extraction task, the classification outcome is whether or not a candidate phrase is a golden keyphrase. So we can consider this task as a binary classification task and apply binary cross-entropy loss to fine-tune BERT with a classifier which generates a scalar between 0 and 1.

Each input of BERT is a token sequence of the concatenation of the document and a candidate phrase, where a CLS-token is added at the start and a SEP-token is added between these two parts. A classifier which consists of a linear layer and a sigmoid layer is applied on the output of BERT, where the output is the embedding of the CLS-token.

Given a document d and a candidate phrase c in this document, the output of the classifier which is the score of c is as follows:

$$SC = \sigma(W\boldsymbol{u})$$

Here W is the weight of the linear layer which can be learned in the training period, $\boldsymbol{u}$ is the output of BERT which is the embedding of the CLS-token that represents the document d and candidate phrase c, and $\sigma(\cdot)$ is the sigmoid function. Mathematically, the following loss function shall be minimized:

$$Loss = -y\,log\,SC + (1-y)\,log(1-SC)$$

$$y = \begin{cases} 1, c \in G \\ 0, c \notin G \end{cases}$$

Here, G is the set of golden keyphrases of document d. If c is a golden keyphrase of document, then $y = 1$, otherwise $y = 0$.

3.2.2 Sentence-BERT fine-tuning

Sentence-BERT is a pre-trained language model [18] which is more suitable to calculate semantic textual similarity than original BERT. Sentence-BERT can generate embeddings by three different pooling strategies: using the output of the CLS-token in BERT, using the mean of all output vectors (MEAN-strategy), and using a max-over-time of the output (MAX-strategy). In our case, we use the MEAN-strategy for pooling and fine-tune the pre-trained Sentence-BERT model with triplet network [19]. Figure 4 shows our architecture for fine-tuning pre-trained Sentence-BERT.

Triplet loss: In the scenario of fine-tuning Sentence-BERT, the loss we use that determines the training direction is of vital significance. We utilize triplet loss as the objective function for fine-tuning Sentence-BERT which can capture the difference between positive examples and negative examples, and trains the model to minimize the distance between the anchor text and positive examples, as well as maximize the distance between the anchor text and negative examples, which is suitable for this task.

Given an anchor text t_a, a positive example t_p and a negative example t_n, the following triplet loss function shall be minimized:

$$Loss = \max(\|\boldsymbol{A}-\boldsymbol{P}\| - \|\boldsymbol{A}-\boldsymbol{N}\| + \epsilon, 0)$$

Here, $\boldsymbol{A}$ is the embedding of the anchor text t_a, $\boldsymbol{P}$ and $\boldsymbol{N}$ are the embeddings of the positive example t_p and the embedding of the negative example t_n, respectively. $\|\cdot\|$ is a distance metric and ϵ is a margin that makes the distance between $\boldsymbol{A}$ and $\boldsymbol{P}$ at least ϵ closer than the distance between $\boldsymbol{A}$ and $\boldsymbol{N}$. Here, ϵ is used to train that the distance between the anchor and positive examples are smaller than the distances between the anchor and negative examples. We set the margin ϵ to be 5.

The anchor text, positive examples, and negative examples are entered for fine-tuning Sentence-BERT, where we choose the target document as the anchor, golden keyphrases as the positive examples, and the candidate phrases which are different from the golden keyphrases as the negative examples. Long documents would contain plenty of candidate phrases, which can undermine the balance of the positive and negative examples. Thus we randomly select a fixed number of negative samples as the negative examples to balance the number of positive examples and negative examples.

As for the distance metric, cosine similarity and Euclidean distance are representative in various NLP tasks. We examined these distances through experiments and found the performance of Euclidean distance achieved better results, so we adopt the Euclidean distance for the triplet loss function.

3.3 Scoring candidate phrases

In this stage, candidate phrases are scored by fine-tuned BERT and fine-tuned Sentence-BERT respectively. After scoring the candidate phrases, we rank them by the decreasing order of the scores and select the top-*N* phrases as the results.

When we adopt fine-tuned BERT as the language model, the score of candidate phrase is generated by the classifier which consists of a linear layer and a sigmoid layer. The mathematical expression of the score has been discussed in Section 3.2.1.

When we adopt fine-tuned Sentence-BERT as the language model, the scoring function for a candidate phrase c by Euclidean distance is as follows:

$$d(\boldsymbol{doc}, \boldsymbol{C}) \coloneqq \sqrt{\sum_{i=1}^{M} (\boldsymbol{doc_i} - \boldsymbol{C_i})^2}$$

$$SC = \frac{1}{1 + d(\boldsymbol{doc}, \boldsymbol{C})}$$

Here, $\boldsymbol{doc}$ is the embedding of the given document, $\boldsymbol{C}$ is the embedding of candidate phrase c, $\boldsymbol{doc_i}$ and $\boldsymbol{C_i}$ is the $i-$th item of $\boldsymbol{doc}$ and $\boldsymbol{C}$ respectively, and M is the dimension of the two embeddings.

3.4 Maximal Marginal Relevance (MMR)

If we pay attention only to the phrases which can best describe the document by selecting the top-*N* candidate phrases which are closest to the document that will lead to select redundant keyphrases, lacking diversity. MMR can alleviate this problem. Given a document d, we modify the original MMR proposed in [4] as follows:

$$MMR \coloneqq \arg\max_{c_i \in C \setminus K} \left[\lambda \cdot SC_{d,i} - (1-\lambda) \max_{c_j \in K} SC_{i,j}\right]$$

Here, C is the set of candidate keyphrases and K is the set of extracted keyphrases. $SC_{d,i}$ is the score of candidate phrase c_i discussed in Section 3.3. $SC_{i,j}$ is similar to $SC_{d,i}$ which is computed with candidate phrases c_i and c_j. Parameter λ is used to control the weights of closeness and diversity. When λ=1, the essence of MMR is the same as the score in Section 3.3. On the other hand, when λ=0, MMR only considers the diversity.

4. Experiments

In this section, we report our experimental evaluations of our fine-tuned BERT and fine-tuned Sentence-BERT models, compared with baseline methods, on commonly used datasets. F1-score is used for evaluation. We also report the results of using the MMR which considers diversity.

4.1 Datasets

We use three datasets which are commonly used for evaluating keyphrase extraction task.

Inspec: This dataset consists of 2,000 short documents from scientific journal abstracts in English. The training set, validation set and test set contain 1,000, 500, and 500 documents, respectively. Each document has two sets of keyphrases: A set of controlled terms assigned by a

Table 1: Results of different methods when N=5 and N=10 (N represents the number of extracted keyphrases)

N		Methods	Inspec			DUC 2001			SemEval 2010		
			P	R	F1	P	R	F1	P	R	F1
5	Baseline methods	TextRank	24.87	10.46	14.72	19.83	12.18	15.17	4.96	2.36	3.20
		EmbedRank Doc2vec	41.49	25.40	31.51	30.87	19.66	24.02	3.67	1.65	2.28
		EmbedRank Sent2vec	39.63	23.98	29.88	34.84	22.26	27.16	5.52	2.43	3.37
		EmbedRank++ Sent2vec ($\lambda = 0.5$)	37.44	22.28	27.94	24.75	16.20	19.58	2.77	1.24	1.71
	Pre-trained models	KeyRank BERT	43.28	26.82	33.12	11.28	7.37	8.92	2.20	0.82	1.20
		KeyRank Sentence-BERT	41.56	25.76	31.81	30.56	19.58	23.87	5.88	2.43	3.42
		KeyRank++ Sentence-BERT ($\lambda = 0.5$)	33.92	20.48	25.54	24.35	15.60	19.02	2.60	0.83	1.26
	Fine-tuned models	KeyRank BERT (fine-tuned)	**55.96**	**33.60**	**41.99**	**54.11**	**34.43**	**42.08**	**18.40**	**6.39**	**9.49**
		KeyRank Sentence-BERT (fine-tuned)	42.36	26.30	32.45	40.61	24.79	30.78	11.00	3.97	5.83
		KeyRank++ Sentence-BERT (fine-tuned) ($\lambda = 0.5$)	38.08	22.76	28.49	26.63	16.45	20.34	6.20	2.21	3.26
10	Baseline methods	TextRank	22.99	11.44	15.28	13.93	16.83	15.24	6.53	6.58	6.55
		EmbedRank Doc2vec	35.75	40.40	37.94	25.38	31.53	28.12	3.93	3.27	3.57
		EmbedRank Sent2vec	34.97	39.49	37.09	28.82	35.58	31.85	5.67	5.18	5.41
		EmbedRank++ Sent2vec ($\lambda = 0.5$)	30.31	34.29	32.18	18.27	23.34	20.50	1.89	1.71	1.79
	Pre-trained models	KeyRank BERT	37.00	42.09	39.38	11.89	15.18	13.34	2.70	1.86	2.20
		KeyRank Sentence-BERT	35.92	41.40	38.46	25.64	32.18	28.54	6.30	4.49	5.24
		KeyRank++ Sentence-BERT ($\lambda = 0.5$)	29.16	32.96	30.94	18.90	23.09	20.79	2.20	1.50	1.78
	Fine-tuned models	KeyRank BERT (fine-tuned)	**44.24**	**49.07**	**46.53**	**44.19**	**54.63**	**48.86**	**16.50**	**11.48**	**13.54**
		KeyRank Sentence-BERT (fine-tuned)	36.88	42.30	39.40	31.91	38.69	34.98	9.30	6.51	7.66
		KeyRank++ Sentence-BERT (fine-tuned) ($\lambda = 0.5$)	33.12	37.62	35.22	21.99	26.69	24.11	5.50	3.88	4.55

professional indexer associated with them and a set of uncontrolled terms that can be any suitable terms [8]. To compare with previous work, we use the uncontrolled terms as golden keyphrases.

DUC 2001: This dataset consists of 308 medium length newspaper articles from TREC-9, where the documents are organized into 30 topics. DUC 2001 dataset is often used to evaluate document summarization and has no golden keyphrases in the original dataset. The golden keyphrases we used were annotated by X. Wan and J. Xiao [20].

SemEval 2010: This dataset consists of 284 long documents which are full scientific conference papers of between 4 and 12 pages, 144 documents for training, 100 documents for testing and 40 for validation [10]. The golden keyphrases were chosen by authors and collaborators.

4.2 Baseline methods

We compare our KeyRank fine-tuned BERT and fine-tuned Sentence-BERT methods to baseline methods (i.e. TextRank, EmbedRank Sent2vec, EmbedRank Doc2vec, EmbedRank++ Sent2vec).

TextRank: TextRank [14] is the first graph-based method for keyphrase extraction proposed by Mihalcea and Tarau. The details of this method are mainly discussed in Section 2.1. It follows the idea of PageRank [3] and achieves a great performance.

EmbedRank: EmbedRank [2] is an embedding-based unsupervised method. This method uses both Sent2vec and Doc2vec to generate the embeddings of candidate phrases and the given document.

EmbedRank++: EmbedRank++ [2] considers the accuracy as well as diversity with Sent2vec and ranks the candidate phrases by MMR which is computed by normalized cosine similarity.

4.3 Experimental details

In the experiments, we use Python Natural Language Toolkit (NLTK) to generate POS tags and use AdamW [13] as the optimizer. Besides, we fine-tune both BERT and Sentence-BERT with a batch-size of 32, learning rate [5e-5, 3e-5, 2e-5, 1e-5], weight decay 0.01, warm-up over 10% of the training data, and save the model which achieves the best performance.

For pre-trained models, we use pre-trained BERT-base-uncased model and Sentence-BERT model with MEAN-pooling layer which is pre-trained with BERT-large-uncased model.

4.4 Performance comparison

Here, we compare the performances of our KeyRank method with fine-tuned BERT and fine-tuned Sentence-BERT models with diversity parameter λ=1 and λ=0.5, with the baseline methods as well as unsupervised methods with pre-trained language models. For unsupervised methods, we simply apply the pre-trained BERT model and pre-trained Sentence-BERT model to obtain embeddings of document and candidate phrases.

For evaluation, we use the common metrics of Precision (P), Recall (R) and F1-score (F1). Table 1 shows the results when N=5 and N=10, where N is the number of keyphrases we want to extract from a document. As shown in Table 1, the performance of our method KeyRank fine-tuned BERT shows the best results on both three datasets and achieves a great improvement than other compared methods, reflecting our method has significant advantages on this task. The performance of KeyRank fine-tuned Sentence-BERT is ranked second, except F1@5 on Inspec dataset which is ranked third, indicating that our supervised method is able to achieve an improvement on keyphrase extraction task. One advantage of KeyRank Sentence-BERT over KeyRank BERT is its efficiency in training and selecting top-K phrases in Euclidian space.

For longer document datasets, especially DUC 2001 dataset which is generated from newspaper articles, the performance of KeyRank fine-tuned BERT and fine-tuned Sentence-BERT are at least 14% and 3% higher than EmbedRank Sent2vec, which is the baseline that has the best performance on this dataset, respectively. Also, on the SemEval 2010 dataset, KeyRank fine-tuned BERT achieves 6.12% and 6.99% higher F1-scores than the best compared baseline method, when N=5 and N=10, respectively, while KeyRank fine-tuned Sentence-BERT achieves 2.46% and 1.11% improvements, respectively.

Notice that the F1-scores of KeyRank fine-tuned BERT is the best on both three datasets when N=5 and N=10. KeyRank pre-trained BERT shows better results than fine-tuned Sentence-BERT on the Inspec dataset when N=5, while on the other two datasets, namely DUC 2001 and SemEval 2010, the performance is the worst in all of the methods. This indicates the significance of fine-tuning BERT which applies self-attention to the document and phrases that we want to score, so that the information between these two parts can be used in the classification stage.

Table 2: Selected keyphrases of article LA051590-0065 (ID) using fine-tuned Sentence-BERT model when λ=1 (without diversity) and λ=0.5 (with diversity)

λ	Selected keyphrases
1	fire season, fire-season announcement, wildfire seasons, fire officials, firefighting forces, firefighting crews, fire-resistant species, wildfire danger, campfire, various fire prevention regulations
0.5	los angeles, fire-season announcement, fire officials, fire-resistant species, fire season, rainfall season, national forests, right, training, cleveland

We also discuss the results on the diversity of selected keyphrases in order to reduce redundancy. The F1-scores of KeyRank++ fine-tuned Sentence-BERT is higher than KeyRank++ pre-trained Sentence-BERT and EmbedRank++ Sent2vec, when λ=0.5 on the three datasets.

Table 2 shows the selected keyphrases from article LA051590-0065 (ID), generated by using fine-tuned Sentence-BERT model when λ=1 (without diversity) and λ=0.5 (with diversity). As shown in Table 2, using MMR can significantly reduce the redundant phrases in the results. Although the F1-score of using MMR to select the candidate phrases is lower than that of using Euclidean distance score, it can still produce a better performance than the compared baseline methods when λ=0.5.

5. Conclusion

In this paper, we proposed a supervised method KeyRank for keyphrase extraction from documents, utilizing fine-tuned BERT and Sentence-BERT. KeyRank fine-tuned BERT is trained with a cross-encoder by binary cross-entropy loss, while KeyRank fine-tuned Sentence-BERT is trained by triplet loss. The cross-encoder generates an embedding from two inputs: the document text and one of the candidate phrases. The self-attention layer contributes to the model so that the information between these two parts is captured and the model can make full use of contextual information. For Sentence-BERT fine-tuning, triplet loss fine-tunes the model with three different inputs: The anchor, positive examples, and negative examples and generates the embedding of positive examples closer to the anchor than negative examples which can achieve a significant improvement on this task. Our experimental results show that KeyRank has significant advantages on this task over the compared baseline methods.

References

[1] R. Alzaidy, C. Caragea, C. L. Giles. Bi-LSTM-CRF sequence labeling for keyphrase extraction from scholarly documents[C]. The World Wide Web conference. 2019: 2551-2557.

[2] K. Bennani-Smires, C. Musat, A. Hossmann, et al. Simple Unsupervised Keyphrase Extraction using Sentence Embeddings[J]. CoNLL 2018, 2018: 221.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine[J]. Computer networks and ISDN systems, 30(1-7):107–117, 1998.

[4] R. Campos, V. Mangaravite, A. Pasquali, et al. YAKE! collection-independent automatic keyword extractor[C]. European Conference on Information Retrieval. Springer, Cham, 2018: 806-810.

[5] J. Carbonell, J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries[C]. Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. 1998: 335-336.

[6] J. Devlin, MW. Chang, K. Lee, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[C]. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2019: 4171-4186.

[7] S. R. El-Beltagy, A. Rafea. KP-Miner: A keyphrase extraction system for English and Arabic documents[J]. Information systems, 2009, 34(1): 132-144.

[8] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge[C]. Proceedings of the 2003 conference on Empirical methods in natural language processing. 2003: 216-223.

[9] S. Humeau, K. Shuster, M. A. Lachaux, et al. Poly-encoders: Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring[C]. International Conference on Learning Representations. 2019.

[10] S. N. Kim, O. Medelyan, M. Y. Kan, et al. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles[C]. Proceedings of the 5th International Workshop on Semantic Evaluation. 2010: 21-26.

[11] Q. Le, T. Mikolov. Distributed representations of sentences and documents[C]. International conference on machine learning. 2014: 1188-1196.

[12] Y. Lim, D. Seo, Y. Jung. Fine-tuning BERT Models for Keyphrase Extraction in Scientific Articles[J]. Journal of advanced information technology and convergence, 2020, 10(1): 45-56.

[13] I. Loshchilov, F. Hutter. Decoupled weight decay regularization[J]. arXiv preprint arXiv:1711.05101, 2017.

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

[15] T. Mikolov, K. Chen, G. Corrado, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.

[16] M. Pagliardini, P. Gupta, M. Jaggi. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features[C]. Proceedings of NAACL-HLT. 2018: 528-540.

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

[18] N. Reimers, I. Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks[C]. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019: 3973-3983.

[19] F. Schroff, D. Kalenichenko, J. Philbin. Facenet: A unified embedding for face recognition and clustering[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 815-823.

[20] X. Wan, J. Xiao. Single Document Keyphrase Extraction Using Neighborhood Knowledge[C]. AAAI. 2008, 8: 855-860.