Keyphrase Extraction Using PageRank and Word Features

Huong T. Le
School of Information and Communication Technology
Hanoi University of Science and Technology
Hanoi, Vietnam
huonglt@soict.hust.edu.vn

Que X. Bui
School of Information and Communication Technology
Hanoi University of Science and Technology
Hanoi, Vietnam
buixuanque8297@gmail.com

***Abstract*— Keyphrase extraction is a fundamental task in natural language processing. Its purpose is to generate a set of keyphrases representing the main idea of the input document. Keyphrase extraction can be used in several applications such as recommendation systems, plagiarism checking, text summarization, and text retrieval. In this paper, we propose an approach using PageRank and word features to compute keyphrases' scores. Experimental results on SemEval 2010 dataset show that our method provides promising results compared to existing works in this field.**

Keywords*— keyphrase extraction, unsupervised learning, PageRank, word embedding, word features

I. INTRODUCTION

Nowadays we can get a huge amount of documents from the Internet. To get appropriate documents to read, it is important to extract a set of keyphrases that reflects the main idea of the document. Also, keyphrases are useful in several tasks such as finding related documents for the document being read, summarizing, clustering, or classifying documents.

Researches on keyphrase extraction can be divided into supervised and unsupervised learning. The supervised learning considers the keyphrase extraction task as a binary classification, in which candidate keyphrases are classified as positive or negative (e.g., [1], [2], [3]). Frank et al. [1] developed a Na¨ıve Bayes classifier for this task using only two features: tf.idf of a phrase, and the distance of a phrase from the beginning of a document (i.e., its relative position). Caragea et al. [2] implemented a supervised probabilistic approach using traditional features combined with their new features basing on citation context information. Shi et al. [4] use Wikipedia as an additional feature source for their supervised keyphrase extraction method. Supervised learning approaches require training data including documents with their keyphrases. The supervised approaches with a large training dataset often provide better performance compared to the unsupervised ones. However, the system applying these approaches is difficult to change from one domain to another, as they depend on the domain of the training data.

Unsupervised approaches (e.g., [5]) do not have this problem as they do not require any training data and can be applied to any domain. These approaches are often based on different features such as tf.idf, clustering, and graph ranking to compute keyphrases' scores. Campos et al. [5] used a set of features including (1) casing; (2) word position; (3) word frequency; (4) word relatedness to context; and (5) word in different sentences. The drawback of the above approach is that it bases on text matching, without considering textual semantic meaning. Mahata et al. [6] solved this problem by computing the cosine similarity between their phrase embeddings. To get phrase embeddings, they extracted unigrams, noun phrases, named entities from a text corpus, then trained them by FastText [7] using the Skip-gram model. However, it is impossible to generate semantic embedding vectors for candidate keyphrases in the input document that does not appear in the training dataset.

Another problem in existing researches of keyphrase extraction is keyphrase overlapping. This is the case that several keyphrases generated by the system share the same words. This problem reduces the diversity of the keyphrase set, which causes the keyphrase set unable to reflect all aspects of the input document.

In this paper, we propose an algorithm to solve the above problems by processing candidate phrases at the word level using several word features and combining them to compute the weight of the phrases. Also, we propose an algorithm to solve the problem of keyphrase overlapping to get a better keyphrase set. Our experimental results on SemEval2010 corpus [8] shown that our system provides better performance compared to previous researches on this field.

The rest of this paper is organized as follows. Our method for keyphrase extraction is described in Section 2. Our experiments and results are introduced in Section 3. Finally, Section 4 concludes the paper and gives future directions for our work.

II. METHODOLOGY

A. Overview of the keyphrase extracting algorithm

Our keyphrase extracting algorithm includes nine steps:

1. Generating candidate keyphrases: Extracting unigrams, noun phrases, and named entities from the input document using the Spacy library [9]. Filtering out named entities DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, CARDINAL from the above phrases to get candidate keyphrases. The stopwords appearing at the beginning or at the end of a noun phrase or a named entity are also removed.
2. Generating the embedding vector for candidate keyphrases: We generate word embedding vectors by using FastText and the phrase embedding vector by summing up all word embedding vectors of that phrase.

This work is supported by Hanoi University of Science and Technology (HUST) under the grant project number T2020-PC-208.

978-1-6654-0435-8/21/$31.00 ©2021 IEEE

3. Identifying the topic vector of the document: It is created by extracting words with high tf.idf value in the title and in the body of the input document. We compute the topic vector by sum up all word embedding vectors of the words being extracted.
4. Computing a thematic score for each word in candidate keyphrases basing on the topic embedding vector: The system computes the cosine similarity between the embedding vectors of the document's topic and each word mentioned above.
5. Computing word scores in candidate keyphrases by a modified version of the Pagerank algorithm in [6]. We use words in the candidate keyphrases as nodes in the Pagerank graph instead of using candidate keyphrases themselves.
6. Computing word scores by theirs features described in Section II.C.
7. Compute final word scores by combining scores in steps 5 and 6.
8. Computing the candidate keyphrases' score based on the scores of words within each phrase.
9. Ranking and filtering the candidates to get the final set of keyphrases.

The methods to compute PageRank-based scores and feature-based scores of words are described in the following sections.

B. Computing word scores basing on the PageRank algorithm

To score words in candidate phrases, we use a modified version of the PageRank algorithm in [6]. We use words instead of phrases as nodes in the graph. First, stopwords are removed from the input document d. Then, a directed graph G is constructed from the remained words of that document, with each word is a node in the graph. An edge connects two nodes if they co-occur in a window size of 5. The weight of an edge connecting two words w_i and w_j is computed by their semantic similarity and their frequency of co-occurrence.

$$sr(w_i, w_j) = semantic(w_i, w_j) * cooccur(w_i, w_j)$$

In which:

- $semantic(w_i, w_j) = \frac{1}{1-cosine(w_i, w_j)}$ is the cosine similarity between two words.
- $cooccur(w_i, w_j)$ is calculated as the Pointwise Mutual Information (PMI) measure between two words:

  $cooccur(w_i, w_j) = PMI(w_i, w_j)$

  $= \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}$

  with $P(w_i)$, $P(w_j)$, and $P(w_i, w_j)$ is the probability of w_i, w_j, and the probability of co-occurrence of both w_i and w_j in the dataset. The higher value of PMI, the more similarity between w_i and w_j.

The PageRank score $R(w_i)$ of a word w_i is:

$$R(w_i) = (1-d)sw_i + d * \sum_{w_j \in E(w_i)} \left(\frac{sr(w_i, w_j)}{|out(w_j)|}\right) R(w_j)$$

in which:

- $E(w_i)$ is the set of all edges connecting to w_i
- sw_i is the thematic score of w_i calculated in Step 4
- d is the damping factor in the PageRank algorithm.
- $|out(w_j)|$ is the out-degree of the node corresponding to the word w_j

Most research chooses the damping factor in the PageRank algorithm as 0.85. In our approach, we carried out several tests with different values of the damping factor (from 0.75 to 0.9). The optimal damping factor's value in our experiments is also 0.85. Therefore, the damping factor d is assigned as 0.85 in our system.

C. Computing word scores basing on theirs features

The score of a candidate keyphrase is computed based on the score of each word in the keyphrase. The word score is computed based on three measures, including (i) word frequency, (ii) the word's surrounding context, (iii) the total number of sentences containing that word. These features are inherited and improved from measures in [5]. These features are described below.

1) Word frequency

Terms that frequently appear in a document reflect the main idea of that document. However, stopwords also appear with high frequency in documents. To reduce the effect of stopwords, the frequency of a candidate word is divided by the mean of word frequencies plus their standard deviation (σ), as in Eq. (1):

$$W_{Freq} = \frac{TF(w)}{MeanTF + \delta} \quad (1)$$

2) Word context

This value is inherited from [5], which is based on the assumption "the higher the number of different words that co-occur with the candidate word t on both sides, the less significant word t will be". The formula to compute this value (W_{Rel}) is as follow:

$$W_{Rel} = 1 + (WL + WR) * \left(\frac{TF(w)}{MaxTF}\right) \quad (2)$$

In which

- WL [WR] is the ratio between the number of different words that co-occur on the left [right] of the considered word and the number of words that it co-occurs on the left [right] of the considered word.
- TF (w) is the word frequency

- MaxTF is the highest frequency of a word in the document.

3) Number of sentences containing the considered word

A word that appears in several sentences of a document indicates the importance of that word. If it is not a stopword, it will be the keyword of the document. The value WDifSentences in [5] is used to evaluate this information:

$$W_{DifSentences} = \frac{SF(w)}{\#Sentences} \quad (3)$$

In which

- SF(w) is the number of sentences containing that word
- #Sentences is the number of sentences in the input document

D. Computing the word score

The feature-based word score is computed by the following formula:

$$S\_feature(w) = \frac{{W_{Rel}}^2}{W_{Freq} + W_{DifSentences}} \quad (4)$$

The smaller value of S_feature(w), the more important the word is. Equation (4) indicates that words which appear frequently in many sentences, and are not stopwords, are important.

The final word score is computed by the value of S_feature(w) combined with the word score computed in Section II.B, as in Eq. (5):

$$S(w) = \frac{1}{R(w)} + S\_feature(w) \quad (5)$$

The smaller the S(w) value, the more important a word is.

E. Computing the candidate keyphrase's score

The scores of a noun phrase or a named entity extracted from the input document are computed based on the scores of words within these phrases by the following formula:

$$S(ckp) = \frac{\sum_{w \in ckp} S(w)}{Tf(ckp) * ngram} \quad (6)$$

The final score of a candidate phrase (ckp) is determined by the total score of each word in that phrase divided by the frequency of that phrase in the document and the number of words in that phrase. The smaller value of the S(ckp), the more important the phrase is. The candidate keyphrases that appear more frequently in the document will have higher advantages.

F. Selecting Keyphrases

As we mentioned in Section I, a problem in existing researches on keyphrase extraction is the overlapping among keyphrases, which reduces the diversity of the keyphrase set. Our strategy to deal with this problem is that if the similarity between two keyphrases is larger than a certain threshold, the one with a higher score will be removed.

After extracting candidate keyphrases from the input document and computing their scores, we perform keyphrase extraction by ranking and filtering the candidates. The criteria to select keyphases are: (i) the value S(ckp) is low; (ii) the overlapping words among keyphrases is as small as possible. Our algorithm to get the final set of keyphrases is as follow:

==========

Input: a set of candidate keyphrases in companied with their S(ckp) value

Output: a set of keyphrases KPs

Algorithm:

1. Sort all candidate keyphrases by their S(ckp) value in increasing order. The result set is named CKPs.
2. Compute the embedding vector for each candidate keyphrase. A phrase embedding vector is computed by getting the average value of its word embedding vector.
3. Add the candidate keyphrase with the lowest S(kw) value to the keyphrase set KPs.
4. Consider the next candidate keyphrase (ckp) in CKPs:
    1. Compute the semantic similarity between this candidate with all keyphrases in KPs using the cosine measure between their embedding vectors.
    2. The ckp is removed from the CKPs in the following cases:
        - The similarity between the ckp with a keyphrase in KPs is more than 0.75
        - The ckp and a keyphrase in KPs have the same number of words and are created by words with the same lemma
        - Both the ckp and a keyphrase in KPs contain two or three words and have at least two common words
    3. Otherwise, the ckp is added to the keyphrase set KPs.
5. Repeat step 4 until reaching the end of the candidate set CKPs.

==========

We select top 5, top 10, top 15 of the keyphrase set KPs to be the keyphrases of the input document.

III. Experiments and Results

A. SemEval 2010 corpus

The SemEval 2010 dataset [8] is used to evaluate our method. Each document in the dataset is 6 to 8-page length on average, including tables and images. The dataset is created by gathering scientific papers from four different research areas, (i.e., Distributed Systems, Information Search and Retrieval, Distributed Artificial Intelligence – Multiagent Systems, and Social and Behavioral Sciences – Economics). Keyphrases of each article are created by human experts. The dataset is divided into trial, training, and test set with 44, 144, and 100 articles, respectively.

TABLE I. THE DISTRIBUTION OF ARTICLES ON DIFFERENT TOPICS OF THE SEMEVAL2010 DATASET.

Dataset	Total	#document/topic			
		C	H	I	J
Trial	40	10	10	10	10
Training	144	34	39	35	36
Test	100	25	25	25	25

B. Experimental Setting

Our experiments are conducted on a computer with Intel® Core™ i7-6600U Processor, 16GB of RAM, and the Python Integrated Development Environment PyCharm. We train FastText with data collected from Wikipedia, using the Skipgram model. The parameters using in our training step are as follow:

window size = 5; dimension = 100; number of epochs = 10.

Since Wikipedia is a scientific website, its vocabulary is close to the SemEval 2010 dataset. We only use the first one billion bytes of Wikipedia and train our system with the same parameter as the original FastText model.

C. Experimental Results

Researches on keyphrase extractions evaluate the results by comparing keyphrases generated by the system with the keyphrases in the dataset at word level and computing micro-average Precision (P), Recall (R), and F1-score (F1). We measure the performance at three levels: top 5, top 10, and top 15 keyphrases. The final results (in percentage) are shown in Table II below.

TABLE II. PERFORMANCE OF OUR SYSTEM WITH THE SEMEVAL 2010 DATASET

	P	R	F1
Micro Avg @15	47.37	28.55	35.63
Micro Avg @10	56.70	21.84	31.54
Micro Avg @5	69.20	13.33	22.35

Table III shows the comparison between our system performance and Key2Vec [6], using the same dataset SemEval2010.

TABLE III. COMPARISON WITH KEY2VEC ON SEMEVAL 2010 DATASET

	P @5	R @5	F1 @5	P @10	R @10	F1 @10	P @15	R @15	F1 @15
Our system	**69.20**	13.33	**22.35**	**56.70**	21.84	**31.54**	**47.37**	28.55	**35.63**
Key2vec	41.00	**14.37**	21.28	35.29	**24.67**	29.04	34.39	**32.48**	33.41

Table III indicates that our proposed method provides better Precision and F1-score compared to Key2Vec [6]. The recall is a bit lower than Key2Vec because of the following reason.

The evaluation tool evaluates keyphrases by the sum of word scores in the keyphrases. A word with a high score will increase the score of the keyphrase it belongs to. Because of that, the set in which high score words repeatedly appear in several keyphrases will a have higher score, resulting in a high value of recall. However, this will reduce the diversity of the keyphrase set.

Since the purpose of our approach is to increase the diversity of the keyphrases and reduce the overlapping among them, we remove the overlapping keyphrases. That is the reason we have higher Precision and lower Recall.

An example of our system output is shown in Table IV. The output keyphrases are sorted by decreasing order of their scores.

TABLE IV. AN EXAMPLE OF OUR SYSTEM OUTPUT USING THE SEMEVAL 2010 DATASET

Keyphrases generated by our system	Standard keyphrases from the SemEval 2010 dataset
'service', 'resource', 'environment', 'node', 'object', 'limit', 'support', 'management', 'grid', 'migration', 'adaptive services', 'fragmented object', 'distributed', 'project', 'edas'	'decentralized adaptive service', 'resource management', 'home environment', 'infrastructure', 'client', 'long-term service', 'eda', 'local limit', 'global limit', 'resource', 'node', 'grid computing', 'fragment object', 'adaptability'

In the above example, the keywords/keyphrases generated by our system are rarely overlapped and cover most of the standard keywords/keyphrases when comparing at the word level.

We also compared our system performance with other research on SemEval2010 dataset. The results are represented in Table V below.

TABLE V. COMPARISON WITH SOME RESEARCHES ON THE SEMEVAL 2010 DATASET

SemEval 2010 (Combined)	Our system	Mahata et al. [6]	Danesh et al. [10]	Lopez and Romary [11]	Bougouin et al. [12]
Micro Avg. F1@10	**31.54**	29.04	26.07	22.05	12.1

Table V indicates that our proposed system is better than other systems on F1-score.

IV. CONCLUSIONS AND FUTURE WORK

This paper proposed an approach to keyphrase extraction that can take advantage of word features and the semantic meaning of words and phrases. This approach can solve the problem of keyphrase overlapping and lacking information of keyphrases in the training data. Experimental results show that

our approach can provide better performance compared to existing researches in this field.

Future work includes investigating other features to compute the score of keyphrases, in order to increase the system accuracy. Also, we will apply this method for extracting keyphrases from Vietnamese text. We will integrate keyphrase extraction system to other applications such as plagiarism checking, recommendation system, etc…

REFERENCES

[1] Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. 1999. Domain-specific keyphrase extraction.

[2] Caragea, C., Bulgarov, F. A., Godea, A. and Gollapalli, S. D. (2014) Citation-enhanced keyphrase extraction from research papers: A supervised approach. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, October 25-29, 2014, A meeting of SIGDAT, a Special Interest Group of the ACL, 1435–1446.

[3] Wang, L. and Li, S. (2017) Pku_icl at Semeval-2017 task 10: Keyphrase extraction with model ensemble and external knowledge. In Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017, 934–937.

[4] Shi, T., Jiao, S., Hou, J. and Li, M. (2008) Improving keyphrase extraction using Wikipedia semantics. In Proceedings of Second International Symposium on Intelligent Information Technology Application, 2008. IITA'08., vol. 2, 42–46. IEEE

[5] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, Adam Jatowt, "YAKE! Collection-Independent Automatic Keyword Extractor," 2018.

[6] Debanjan Mahata, John Kuriakose, Rajiv Ratn Shah, Roger Zimmermann, "Key2Vec: Automatic Ranked Keyphrase Extraction from Scientific Articles using Phrase Embeddings," Association for Computational Linguistics, New Orleans, Louisiana, 2018.

[7] FastText. https://github.com/facebookresearch/fastText (last visited April 2021)

[8] Su Nam Kim, Olena Medelyan, Min-Yen Kan, Timothy Baldwin, 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In Proceedings of the 5th International Workshop on Semantic Evaluation, ACL 2010

[9] Hannibal, Matt. Spacy. Available: https://spacy.io/ (last visited April 2021)

[10] Danesh, S.; Sumner, T.; and Martin, J. H. 2015. SGRank: Combining statistical and graphical methods to improve the state of the art in unsupervised keyphrase extraction. In Proc. Joint Conference on Lexical and Computational Semantics

[11] Lopez, P., and Romary, L. 2010. HUMB: Automatic key term extraction from scientific articles in GROBID. In Proc. SemEval.

[12] Bougouin, A.; Boudin, F.; and Daille, B. 2013. Topicrank: Graphbased topic ranking for keyphrase extraction. In Proc. IJCNLP