IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 9, 2020, accepted June 16, 2020, date of publication June 24, 2020, date of current version July 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3004628

Automatic Keywords Extraction Based on Co-Occurrence and Semantic Relationships Between Words

XIANGKE MAO, SHAOBIN HUANG, RONGSHENG LI, AND LINSHAN SHEN
College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China
Corresponding author: Linshan Shen (shenlinshan@hrbeu.edu.cn)

ABSTRACT Automatic keywords extraction is a method that extracts words or phrases from a document which can express the main idea of the document. In this paper, we propose an unsupervised keywords extraction framework for individual documents, which improves the keywords extraction from two aspects. In the step of candidate keywords selection, we use the methods of removing the stopwords, regular matching, and length filtering to reduce the number of candidate keywords, but improve the quality. In the step of scoring words, we use word co-occurrence, semantic relationships (WordNet, Word Embedding, Normalized Google Distance), and three ways to combine word co-occurrence and semantic relationships to measure the weight of edges in the graph model. In experiments, we use Precision, Recall, and F1-measure values as evaluation criteria to compare all keywords extraction methods we proposed with other strong baseline methods in two datasets. According to the results of experiments, methods under our proposed framework achieve good results. We verify that the methods of using both word co-occurrence and semantic relationships have a better effect on keywords extraction than using co-occurrence or semantic relationships only. At the same time, we also find that for the keywords extraction of individual documents, the method of using co-occurrence between words has a better effect than semantic relationships.

INDEX TERMS Automatic keywords extraction, graph model, semantic similarity, TextRank, word co-occurrence.

I. INTRODUCTION

Automatic keywords extraction (AKE) is a kind of method that automatically catches the theme of one document using a small set of words occurred in the document. Especially in the age of '' information explosion'', AKE is one method for people to learn information quickly from the document ocean. AKE is widely used in many natural language processing (NLP) tasks, such as Text Classification (TC) [1], Document Summarization (DS) [2], [3], Information Retrieval (IR) [4], [25] *et al.* For an IR system, keywords can be applied to index documents and improve the accuracy rate of retrieval results. For a document, keywords can be seen as a condensed summary. Although we know that keywords extraction is very useful, it is a time and money consuming task to assign keywords manually. Therefore, the automatic extraction of keywords from documents has attracted great interest from researchers in recent years.

Most methods for extracting keywords can be divided into three steps: The first step is to get candidate keywords from the document; the second step is scoring the candidate keywords, and the last is selecting keywords based on the results of the first two steps.

For the first step, we aim to reduce the number of candidate keywords but improve the quality. In this paper, we remove the stopwords based on the stopwords list first. Next, we use CoreNLP[1] tool to tag every remaining word in the document, and use regular expression to extract the noun chunks whose pattern is zero or more adjectives followed by at least one noun. Finally, we use the length rule to filter the content extracted by regular expression.

According to different ways of scoring words, AKE methods can be divided into two types: supervised and

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

[1] https://stanfordnlp.github.io/CoreNLP/

unsupervised. In supervised methods, keyword extraction is formulated as a classification problem, where candidate keywords are classified into two categories, keyword and non-keyword. There are many manually defined features used for classifier training, such as word position, TF-IDF, word occurred in the title, part-of-speech (PoS) *et al.* Many classification algorithms are also used, such as logistic regression, support vector machine, naive bayes, decision tree, neural network, *et al.* Although these features and algorithms have shown good results in keywords extraction, all need training datasets. In unsupervised methods, we get rid of the constraint of needing training datasets, which usually adopts various scoring indicators (such as tf-idf, word position, *et al.*) to rank the candidate keywords. In recent years, graph-based keyword extraction models have been widely used, which represent the document as a graph. The nodes in the graph represent the words in the document, and the edges represent the relationships between nodes. Different relationships between nodes can get different topological structures, so when using rank algorithms on graph, we can get different scores of the same word. In this paper, we use the word co-occurrence and semantic relationships to score words. The reason for combining these two relationships together is that only using word co-occurrence to construct word graph is limited by the size of the window. However, the use of semantic relationships can break through this limitation and link any two semantically related words in the document. In this paper, we use WordNet [26], normalized google distance (NGD) [27] and word embedding [24], [32], [34] to calculate the semantic similarity between words.

For the final step, we select the keywords from candidate keywords. According to the results of word scoring, the score of candidate keywords is calculated and sorted from high to low. After word stemming, we remove the duplicate candidate keywords and choose the top-k as keywords.

The main contributions of this paper are summarized as follows: 1

1) We propose a method to select candidate keywords, which reduces the number of candidate keywords, but improves the quality.
2) We combine word co-occurrence and semantic relationship to extract keywords. In particular, we propose a word graph modeling method that integrates word co-occurrence and semantic relationship into the same graph.
3) We verify the effectiveness of the proposed keywords extraction approach on two datasets, DUC2001 and DUC2002. We also find that the co-occurrence relationship between words is better than the semantic relationships in the keyword extraction task of a single document.

In this paper, we mainly focus on the second step of scoring words in the document. The following sections of this article are organized as follows: In the second section, we describe the keyword extraction methods used in recent years. In the third section, we introduce our proposed method in detail. In the fourth section, we apply our methods on public datasets to extract keywords and compare with others based on the metrics of precision, recall, and F1-measure, and we also analyze the results of experiments. In the fifth section, we conclude this paper and look forward to future work.

II. RELATED WORKS

There are a number of articles that have been well summarized in various aspects of keywords extraction, see [28]–[30]. In this paper, we mainly review the related work of selecting the candidate keywords and word scoring approaches.

Selecting candidate keywords from documents is the first step of most keywords extraction approaches. The quality of the selection will directly affect the results of keywords extraction. In [5], they used three term selection approaches to get candidate keywords. The first is n-gram, the second is noun phrases chunks, and the last is PoS tag patterns, extracting five most frequently occurring patterns from keywords presented in training data. In [6], they use words with special PoS such as non or adjective as candidate keywords. In [7], which get the candidate keywords through removing specified word delimiters and stopwords in the document, they generate a stopwords list from training dataset. [8] uses the regular expression to extract noun groups with specific patterns. In our methods, selecting candidate keywords from documents mainly inspired by [7] and [8].

In the step of word scoring, the methods of scoring can be roughly divided into two types: supervised and unsupervised learning. In supervised learning methods, the first step is to train the classifier through the training datasets labeled with keywords, and then predict whether the words in the new document are keywords according to the trained classifier. Reference [9] proposed the first supervised keywords extraction system named KEA, which only uses two features: TF-IDF and the word first occurrence position. The classification model they use is Naive Bayes. In [5], Hulth incorporates linguistic knowledge into keywords extraction system based on supervised learning. Which use inner document frequency, collection frequency, the first occurrence position, and PoS tag(s) as features. In [18], they proposed a supervised model to extract keywords from academic papers, which design a novel feature from citation network and combine it with some traditional features to perform classification. All these supervised keyword extraction methods are limited by the training datasets, but unsupervised learning methods break through this limitation, so more and more attention has been paid to them.

Among all unsupervised keyword extraction methods, graph-based methods have attracted the most attention. Because it can not only be independent of the corpus and training data, bu also can be combined with other methods. Inspired by the PageRank algorithm [31], TextRank [6] was first proposed to extract keywords and sentences from document, which uses the word in the document to denote the nodes, and word co-occurrence in the same window to establish edge relation between nodes. Various TextRank based

methods have been proposed. SingleRank [11] assigns the number of co-occurrence of words in the sliding window to the weight of edges. Reference [11] also proposed a method named ExpandRank to overcome the limitation of building the graph from a single target document do not have enough information, which finds a document set closed to target document to expand the information of word graph. Reference [12] compared various centrality measures for graph-based keywords extraction, which show that the results of simple degree centrality achieved as well as the classical TextRank method, especially for short documents, closeness centrality perform best. Some keywords extraction methods related to graph model also include [39], [40].

In order to better represent the relationship between nodes in a graph, more and more work tends to use the semantic relationships between words to calculate the weight of edges in the graph. Reference [13] proposed a method called SemanticRank, which uses semantic relationships to extract keywords and sentences from the document. In experiments, their proposed method is outperformed than weighted and unweighted PageRank methods. In [41], they use the information supplied both by word embeddings and local statistical information to compute the weight of edges. Reference [14] proposed a graph-based ranking algorithm with the aids of latent vector representation of terms and term relations embedded in patents. In order to enhance the quality of semantic similarity between candidate words, [15] first introduced a weighting scheme that computes informativeness and phrase scores of words using the information supplied by both word embedding vectors and local statistics. Reference [16] introduced a SemGraph approach to extract keywords from a collection of texts through building semantic relationship graphs based on WordNet, which can select the words with statistical significance. Reference [29] proposed a parameterless method for constructing graph of text that captures the contextual relationship between words, and designed a novel word scoring method that aims to capture contextual hierarchy, semantic connectivity and position weight of words.

Although there are many graph-based keywords extraction efforts focusing on the semantic relationships between words, few efforts have been made to combine word co-occurrence with semantic relationships. Therefore, in this paper, we study the effects of word co-occurrence and semantic relationships on keyword extraction. We propose three methods to extract keywords by combining word co-occurrence with semantic relationships.

Some work adds prior knowledge to the nodes in the graph to emphasize the importance of words, such as position of words, the TF-IDF value, and the similarity between words and title. In [18], they proposed an unsupervised model for keywords extraction from scholarly documents named PositionRank, which incorporates all positions of one word occurring in the document into a biased PageRank. Reference [19] proposed a novel unsupervised graph-based keywords extraction method called Keyword Extraction using Collective Node Weight (KECNW), which considers various factors can influence the importance of words, including position, frequency, centrality, and strength of neighbors. Under the same datasets and measurements, the effect of the KECNW method is outperformed. In [20], Bellaachia and Al-Dhelaan proposed a novel unsupervised graph-based keywords ranking method, called NE-Rank, which considers word weights in addition to edge weights when calculating the ranking.

There are some other papers that add the topic model to the process of extracting keywords. Reference [21] proposed a method to decompose traditional random walk into multiple random walks specific to various topics, they build a Topical PageRank (TPR) on word graph to measure word importance with respect to different topics. In [22], they proposed a graph-based method called TopicRank, which represents a document as a complete graph where vertices are not words but topics. Topics are defined as single and multiword with significant similarity. With the extensive use of deep learning in various fields [35], many new models have also been proposed in the field of keywords extraction [17], [36]–[38], and achieved the best results in many datasets. However, many keywords extraction methods based on deep learning are generally supervised, and the model training will take a long time, so it will not be compared with the methods proposed in this article.

III. METHODS

In this section, we mainly introduce the methods we designed for extracting keywords from documents in detail. Our methods all include three stages, the first stage is selecting candidate keywords from documents, the second stage is scoring the candidate keywords and ranking the candidate keywords, and the last stage is to select keywords from ranked candidate keywords. The framework of our proposed methods is shown in Figure 1.

A. SELECT CANDIDATE KEYWORDS

We first pre-process the document, including document segmentation, tokenization, and PoS tagging. After that, we started to select the candidate keywords.

1) Stopwords Removal
   We already have some very common stopwords in English, such as " a ", " the ", " of ", but they can't satisfy us very well in keywords extraction. In keywords extraction task, except the words we selected as keywords, the other words all can be seen as stopwords. Here we directly use the stopwords provided in [7], and they have verified in experiments that the stopwords list can improve the keywords extraction effect. Next, we give a simple example. " Compatibility of systems of linear constraints over the set of natural numbers ". After we use the symbol " # " to replace the stopwords. We will get " Compatibility # systems # linear constraints # # set # natural numbers ".
2) Regular Matching
   According to results of PoS tagging and stopwords removal, we can get the sequence [(' Compatibility ',

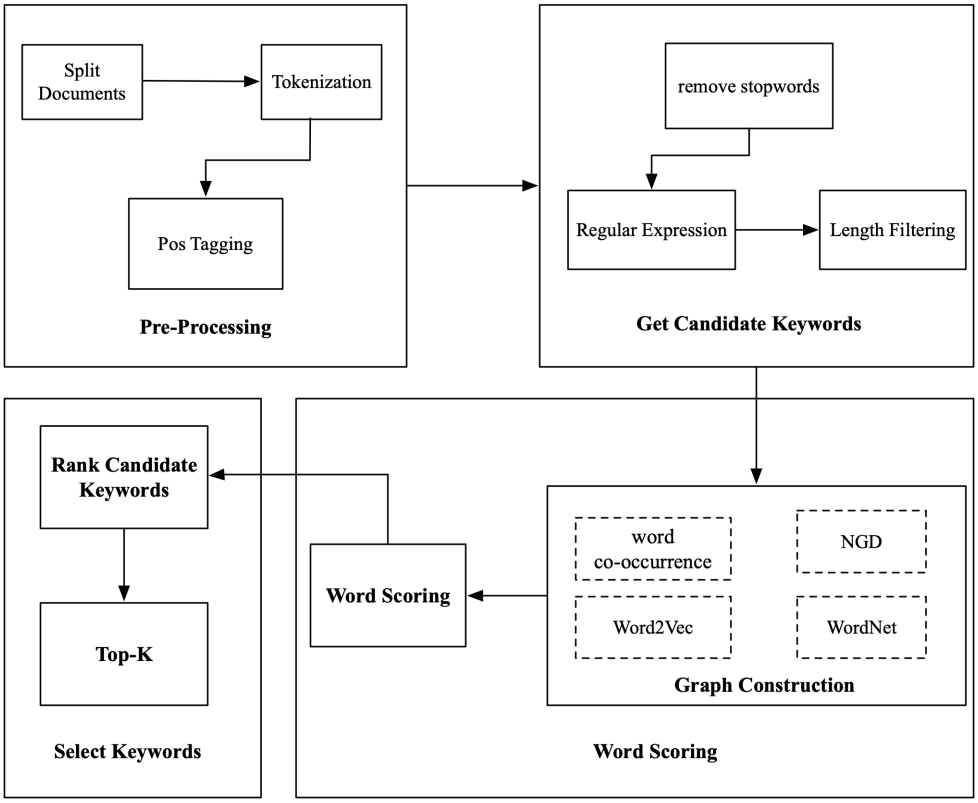

FIGURE 1. Our proposed framework for keywords extraction.

‘ NN ’), ‘ # ’, (‘ systems ’, ‘ NNS ’), ‘ # ’, (‘ linear ’, ‘ JJ ’), (‘ constraints ’, ‘ NNS ’), ‘ # ’ ‘ # ’, (‘ set ’, ‘ NN ’), ‘ # ’ (‘ natural ’, ‘ JJ ’), (‘ numbers ’, ‘ NNS ’)]. Next, we will use a regular expression to extract noun chunks. Based on the PoS of the keywords manually provided by experts, we found that most of the keywords are composed of nouns and adjectives. So, in this paper, we mainly extract nouns and adjectives from documents to form candidate keywords. The regular expression we use is

$$\{< JJ > * < NN[S|P] * > +\}$$

where JJ means adjective and NN means noun. Here, we will get some candidate keywords from example, “ compatibility ”, “ systems ”, “ linear constraints ”, “ natural numbers ”.

3) Length Filtering
According to the length statistic of keywords extracted by experts on Inspec training dataset, most of them are between 1 and 5. So, in this paper, we filter out the candidate keywords longer than five words in the regular matching results.

In the experimental part, we compare the quantity and quality of the selected candidate keywords before and after using our candidate keywords selection method.

B. WORD SCORING

Word scoring is the most important step in keywords extraction. Here, we will introduce the scoring methods based on word co-occurrence, semantic relationships, and the combination of word co-occurrence and semantic relationships, separately.

1) SCORING BASED ON WORD CO-OCCURRENCE

The TextRank algorithm is a variant of PageRank, which represents the text as a graph, and measures the importance of words according to all words in the text. Formally, let $G = (V, E)$ be a graph, where V is denoted as words in document, and E is a set of edges based on the co-occurrence of words, the weight of edge is calculated by the number of co-occurrence between two words in the same slide window. TextRank uses (1) to calculate the score of nodes in the graph.

$$R(V_i) = (1 - d) + d \cdot \sum_{j:V_j \to V_i} \frac{w_{ji}}{\sum_{k:V_j} w_{jk}} R(V_j) \qquad (1)$$

where w_{ji} is the weight of edge from node V_j to the current node V_i and $\sum_{k:V_j} w_{jk}$ is the summation of all edge weight in the previous nodes V_j. Here, d is the damping factor which denotes the probability of randomly selecting one node in the graph, and the value is usually set to 0.85.

2) SCORING BASED ON SEMANTIC RELATIONSHIPS

From the formula (1), we can easily find that the weight of the edges can affect the final score of words. Different methods for edge weight calculation can get different word graphs, so the score of words will be different. In TextRank, the edges in the graph are determined by whether the two words appear in the same window. However, when using word co-occurrence to calculate the weight of edges, it is limited by the size of the sliding window. The use of semantic relationships can break through this limitation and establish weighted edges between any semantically related word. Next we will simply introduce three different ways to calculate the semantic relationships between words.

1) Word Embedding (WE)
Many deep learning models in NLP are based on word embedding as input features. In this paper, we use the pretrained word vectors from SpaCy[2] to calculate the semantic similarity between two words. In SpaCy, the word vector is trained by the GloVe [32] model, where the word is represented as a 300 dimensional vector, so we can use formula (2) to calculate the semantic similarity between words.

$$sim(\mathbf{w}, \mathbf{v}) = \frac{\mathbf{w} \cdot \mathbf{v}}{\|\mathbf{w}\|\|\mathbf{v}\|} \tag{2}$$

Here, $\mathbf{w}$, $\mathbf{v}$ represent word vectors of the words w and v respectively.

2) WordNet
A lot of semantic similarity measures based on WordNet[3] have been proposed, but in this paper, we only use the shortest path between words in WordNet to calculate the distance between two words. We use the *path_similarity* function provided by NLTK[4] tool to get the semantic similarity of word pairs.

3) Normalized Google Distance (NGD)
The Normalized Google Distance (NGD) algorithm [27] is used to measure the semantic distance between two words or phrases. The algorithm assumes that the two words occurred in the same document have semantic relationships, so the more often two words occur in the same document, the semantic relationship between the two words is more relevant. NGD between two words x and y can be calculated as (3).

$$NGD(x, y) = \frac{max\{log(f(x)), log(f(y))\} - log(f(x, y))}{logM - min\{log(f(x)), log(f(y))\}} \tag{3}$$

Here, M is the total number of pages in the database (generally a constant), $f(x)$ is a function, return the number of pages in the database contain the word x, and $f(x, y)$ returns the number of x and y occurred on one page. The formula (3) gives a normalized score which ranges from 0 to ∞. A distance of zero indicates the two words are identical and a distance of ∞ indicates two words never occurred in one page and definitely has no relation.
In this paper, we use Wikipedia as the large corpus. We first download the Wikipedia pages from the website.[5] Next we remove the html labels and extract the context of the page, and then we pre-process the text, including cut text into words, remove stopwords and word stemming. Finally, we build the inverted index for all unique words in the Wikipedia corpus. Based on the inverted index, we can easily obtain the NGD for any two words.

3) SCORING BASED ON WORD CO-OCCURRENCE AND SEMANTIC RELATIONSHIPS

When only using the co-occurrence relationship between words to determine the edge, it is limited by the window size, and there is no edge connection between words that are semantically related but do not appear in the same window. The semantic relationships between words exist objectively, if we only use the semantic relationships between words, we discarded the co-occurrence relationship of words in the document. Therefore, in this paper, we combine these two relationships more comprehensively to measure the importance of words in a document. The three combinations we designed are as follows:

a) The First Combination
First, we use the word co-occurrence to build the word graph, and then use the semantic distance between words to weight the edges in the graph. The edge's weight can be calculated using (4).

$$weight(w, v) = freq(w, v) \cdot semantic(w, v) \tag{4}$$

where the $freq(w, v)$ is the frequency of the words w and v occurred in the same window, and $semantic(w, v)$ is the semantic similarity between words w and v.

b) The Second Combination
The second combination is that we use the word co-occurrence graph to score the words in graph first, and then we use the first combination methods to measure the importance of words in a document. These two different word graphs calculate the words score independently. Finally, we combine these two scores linearly. The formula is seen as (5).

$$score_c = \lambda \cdot score_{co} + (1 - \lambda) \cdot score_{sr} \tag{5}$$

where the λ is a number, the range is [0,1]. When the λ is set to 0, the equation (5) is reduced to equation (4) to calculate the score of the words. While λ is 1, we only use the scores based on word co-occurrence graph.

c) The Third Combination
The third combination is that we regard the words that occurred in the same window as having the strong relation, so we first build the word graph based on word

[2]https://github.com/explosion/SpaCy
[3]https://wordnet.princeton.edu
[4]https://www.nltk.org/
[5]https://dumps.wikimedia.org/backup-index.html

co-occurrence, and then for the words not to appear in the same window, we use the semantic relation to weight the edge. For the weight of the edges, we can use the (6) to calculate.

$$weight(w, v) = \begin{cases} freq(w, v) & (w, v) \in co \\ semantic(w, v) & (w, v) \notin co \end{cases} \tag{6}$$

where the co is a set of word pairs co-occurred in the same sliding window. After constructing the graph, we use equation (1) to score the words.

In order to prevent the appearance of complete graphs, for all graphs we build, we set a threshold for edge's weight. If the edge's weight is under the threshold, we set the edge's weight as zero. The threshold we used in this paper is 0.1.

C. EXTRACT KEYWORDS

After getting candidate keywords from the document, and scoring for the words, next we will rank the candidate keywords and select the keywords. For scoring the candidate keywords, we add the score of each word contained in candidate keywords up, seen as (7).

$$score_c = \sum_{w \in c} score(w) \tag{7}$$

where c is represented as a candidate keyword, and w is the word in c.

According to the scores of candidate keywords, we rank them from high to low. And after words stemming, we removed the duplicate candidate keywords. Finally, we select the first K as keywords.

D. ALGORITHM DETAILS

In this part, we will describe the proposed framework for keywords extraction in the form of an algorithm and analyze its time complexity. In the algorithm 1, 1-3 lines describe the pre-processing process of the document. 5-7 lines describe the selection of candidate keywords. 8-9 lines represent the scoring of words in document, where function *graph()* realizes the construction of word graph according to the co-occurrence and semantic relations between words, and function *score()* implements the scoring process of equation (1). And 10-11 lines realize the selection of keywords, where the function *get_candidate_score()* is used to score candidate keywords according to the results returned by *score()*.

Next, we analyze the time complexity of algorithm 1. In addition to the pre-processing of the document, the most time-consuming two steps are to build the graph in line 8 and score the vertices in the graph in line 9. In order to facilitate the analysis, it is assumed that the candidate keywords contain m words, the word graph has n nodes, and the size of the sliding window is w.

During the construction of the graph, when the sliding window is used to construct the graph, it needs to be moved $m-w$ times from left to right. And, in each window, the upper limit of the edges that can be generated is $w(w-1)/2$. Therefore, in the first combination method, the time complexity of building graph based co-occurrence is $T_{11} = mw(w-1)/2$. In the second combination method, we build the graph twice, so the time complexity is $T_{21} = 2T_{11}$. When building the graph in the third combination method, not only includes edges based on co-occurrence, but also edges based on semantics. Hence, the time complexity is $T_{31} = T_{11} + n(n-1)/2$.

After building the word graph, we use the pagerank algorithm to calculate the weight of each word in graph. The first and third combined methods build only one graph for a single document, so the time complexity is $T_{12} = T_{32} = tn^2$. However, the second combination builds two, so $T_{22} = 2tn^2$. Where t represents the maximum number of iterations, which is 100 in this paper.

The total complexity of the first combination method is $T_1 = T_{11} + T_{12} = mw^2/2 + tn^2$. The second is $T_2 = 2(mw^2/2+tn^2)$. The third is $T_3 = mw^2/2+n^2/2+tn^2$. Where $n \gg w$. Therefore, the time complexity of the algorithms 1 is $T = O(2tn^2)$, which has the same time complexity with basic TextRank. Therefore, when we use the three combined methods proposed, it will not increase the time complexity.

Algorithm 1 Keywords Extraction

Input:
Document collection D for keywords extraction.
Stopwords list sw.
Pre-calculated word semantic similarity sm (WordNet, Word Embedding, NGD)

Output:
Keywords extracted from collection D

1: Cut the doc in D into words.
2: Generate stopwords list sw from T.
3: PoS tagging D and get PoS_D.
4: **for** each $d \in Pos\_D$ **do**
5: no_stop_doc = remove_stopwords(sw, doc)
6: candidate_keywords = get_candidate(no_stop_doc, regular expression)
7: length filtering.
8: G(V, E) = graph(candidate_keywords, sm) // build graph G.
9: words_score = score(G) // Get the score of words in graph G.
10: candidate_score = get_candidate_score(words_score) // Calculate the score of candidate keywords based on the score of words.
11: keywords = top_K(candidate_score).
12: **end for**

IV. EXPERIMENT ANALYSIS

In this section, we first introduce the datasets we used in experiments, then we describe the evaluation metrics for evaluating the effect of proposed methods. Finally, we show the experimental results and analyze the results.

TABLE 1. Overview of experimental datasets. |D|: Number of documents in corpus, SN: sentence number in corpus, WN: word number in document collections, AS: average number of sentences per document, AW: average number of words per document, RK: reference keywords number of corpus, RKI: number of reference keywords occurred in corpus, AK: average number of keywords per document occurred in corpus, AL: average length of per keyword.

datasets	|D|	SN	WN	AS	AW	RK	RKI	AK	AL
Inspec	500	3029	67300	6.18	134.6	4913	3829	7.68	2.27
DUC01	308	11768	237486	38.21	771.05	2488	2488	8.08	2.09

A. TWO DATASETS

The Hulth 2003 dataset was first used in the paper [5], which contains 2000 abstracts from the Inspec database and every abstract has two kinds of manually assigned keywords. One is 'controlled' and the other is 'uncontrolled'. In 'controlled', it assigned through a given dictionary, while 'uncontrolled' is freely assigned by experts. In this paper, we only use the uncontrolled keywords to evaluate the effect of our methods. Hulth also divides the dataset into three parts, 1000 for training, 500 for validation and 500 for testing. In our experiments, we only use 500 testing documents to extract keywords. The details of the dataset show in Table 1.

The DUC2001 datasets was provided by the Document Understanding Conference(DUC[6]). The manually assigned keywords were created by Wan *et al.* in [11]. The dataset contains 309 news articles, but there are two articles with the same content, so the actual number of articles is 308. They finally labeled 2488 keywords from the dataset. The details of the dataset shown in Table 1.

B. METHODS AND EVALUATION METRICS

We use the following methods to extract keywords from two datasets:

- ***TF-IDF***. We calculate the *tf* of each candidate keyword in the target document, and *idf* is estimated from both datasets.
- ***TextRank***. The method used in [6]. Where they build graph based on the words co-occurrence, and edges have the same weight.
- ***SingleRank***. The method proposed in [11]. Where the weight of edge is the co-occurrence number of words in the slide window.
- ***ExpandRank***. The method proposed in [11], where they use the neighborhood documents to adjust the weight of edges.
- ***RAKE***. The method used in [7].
- ***CO***. In our framework, we use the same way as SingleRank to build graph and score words.
- ***Semantic Relationships based Methods***. We only use the semantic relationships to determine the edges of graph. (WE, WordNet, NGD)
- ***Combination Methods***. The first combination method include FCWN (first combination with WordNet), FCWE (first combination with word embedding), FCNGD (first combination with NGD). The second combination method include SCWN (second combination with WordNet), SCWE (second combination with word embedding), SCNGD (second combination with NGD). The third combination method include TCWN (third combination with WordNet), TCWE (third combination with word embedding), TCNGD (third combination with NGD).

To evaluate the keywords extraction methods, we use the results obtained from the above methods to compare with the keywords manually assigned by experts. There are three metrics we use below.

Precision:

$$P = \frac{|EK \cap MANU|}{|EK|}$$

Recall:

$$R = \frac{|EK \cap MANU|}{|MANU|}$$

F1-measure:

$$F1 = \frac{2PR}{P + R}$$

where the EK is represented as the automatically extracted keywords and $MANU$ is represented as the manually assigned keywords. P is precision, which measures the percentage of right extracted keywords in automatically extracted keywords. R is recall, which measures the percentage of right extracted keywords in manually assigned keywords. F value is the harmonic mean of precision and recall.

C. RESULTS AND ANALYSIS

We mainly verify and analyze the effect of the proposed keywords extraction framework through three experiments. First, we verify the effect of our proposed candidate keywords selection method. Second, we compare and analyze the keywords extraction effect of the proposed methods with strong baselines. Lastly, we measure the impact of parameters on keywords extraction.

1) CANDIDATE KEYWORDS SELECTION METHOD VALIDATE

In the first experiment, we follow the keywords extraction framework we proposed to extract candidate keywords. Then, we compare the candidate keywords obtained before and after using our method. The effect is shown in Table 2.

From the results in Table 2, we can find that after using the candidate keywords selection method proposed by us, on both datasets, the number of candidate keywords decreased, but the number of candidate keywords appearing in reference keywords increased. The results demonstrate

[6]https://www-nlpir.nist.gov/projects/duc/data.html

TABLE 2. Before and after using our candidate keywords selection method. CKN: Number of candidate keywords selected from corpus. CKINR: The number of candidate keywords in reference keywords.

datasets	Inspec		DUC01	
	CKN	CKINR	CKN	CKINR
before	16151	3044	46285	2134
after	13154	3177	37611	2248

TABLE 3. Comparison of keyword extraction results among combination methods.

methods	Inspec			DUC01		
	Precision	Recall	F1	Precision	Recall	F1
FCWN	0.359	0.650	0.463	0.240	0.297	0.265
FCWE	0.364	0.658	0.469	0.250	0.309	0.276
FCNGD	0.364	0.658	0.468	0.253	0.314	0.280
SCWN	0.377	0.681	0.485	0.271	0.335	0.300
SCWE	0.382	0.691	0.492	0.280	0.346	0.310
SCNGD	0.383	0.692	0.493	0.283	0.350	0.313
TCWN	0.384	0.695	0.495	0.285	0.353	0.316
TCWE	0.389	0.703	0.501	0.295	0.365	0.326
TCNGD	**0.390**	**0.705**	**0.502**	**0.297**	**0.368**	**0.329**

TABLE 4. Comparison with strong baselines.

methods	Inspec			DUC01		
	Precision	Recall	F1	Precision	Recall	F1
TextRank	0.312	0.431	0.362	0.236	0.289	0.260
SingleRank	0.328	0.593	0.422	0.247	0.303	0.272
ExpandRank	0.383	0.692	0.493	0.288	0.354	0.317
RAKE	0.337	0.415	0.372	0.253	0.314	0.280
CO	0.374	0.677	0.482	0.268	0.332	0.296
WordNet	0.368	0.664	0.473	0.241	0.299	0.267
WE	0.373	0.674	0.481	0.248	0.307	0.275
NGD	0.375	0.677	0.482	0.253	0.313	0.279
TCNGD	**0.390**	**0.705**	**0.502**	**0.297**	**0.368**	**0.329**

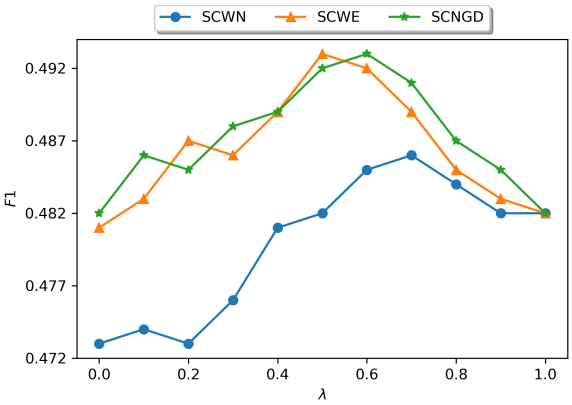

FIGURE 2. F1 values change with λ (Inspec).

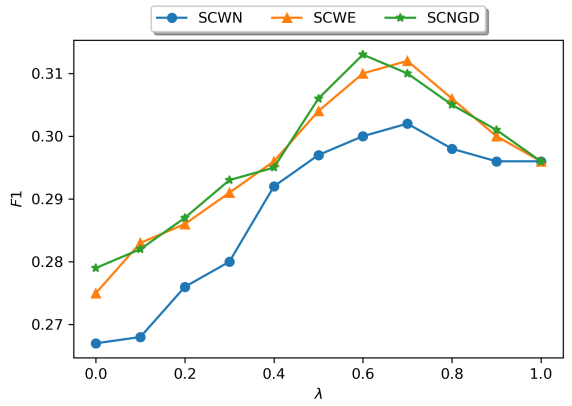

FIGURE 3. F1 values change with λ (DUC2001).

the good effectiveness of the proposed candidate keywords selection method used in keywords extraction.

2) KEYWORDS EXTRACTION EFFECT EVALUATION

In this experiment, we first compare the results we get from the proposed combination methods. The results are shown in Table 3. And then, we choose the best method from proposed methods to compare with strong baselines, the results are shown in Table 4.

In Table 3 and Table 4, all methods get the same number of keywords from each dataset. In the Inspec, we use the ranked top 15 candidate keywords as the selected keywords, Here we all get 6921 words or phrases from documents. In DUC01, we use the top 10 as selected keywords from documents, and we all get 3080 words or phrases. The damping factor used in all methods is equal to 0.85, and the window size is 6. So, we can guarantee the fairness of comparisons among different methods.

According to the results obtained in Table 3, by comparing the keyword extraction effects among the three combination methods, we can easily find that the third combination method has better results than the other two. The F1 values of TCNGD in the Inspec and DUC01 datasets are 0.502 and 0.329, respectively. We analyze the reason for this is that the third combination method can not only express the relative position relationship between words in the window, but also make up for the relationship between words not occurred in the same window through the semantic relationships. We can more accurately and comprehensively identify the important words in the document. While the other two combination methods are limited by the size of the sliding window, and only use a combination of co-occurrence and semantic relationships in the window.

Among the three combined methods, we can find that the method using WordNet to calculate semantics obtained the lowest results, while the method using NGD achieved the best results. At the same time, it can also be found that the experimental results using the methods of NGD and word embedding to measure the semantic relationship between words are very close. We can infer that word embedding and NGD are better able to obtain the semantic relationship between words. This is because word embedding and NGD are calculated based on large corpus, so they contain richer semantic information about words. At the same time, when using WordNet, there is no word sense disambiguation, which will affect the calculation of semantics between words.

In Table 4, comparing the results of CO and SingleRank methods with the same word graph, CO achieves good results on both datasets. We infer that the reason is that our proposed

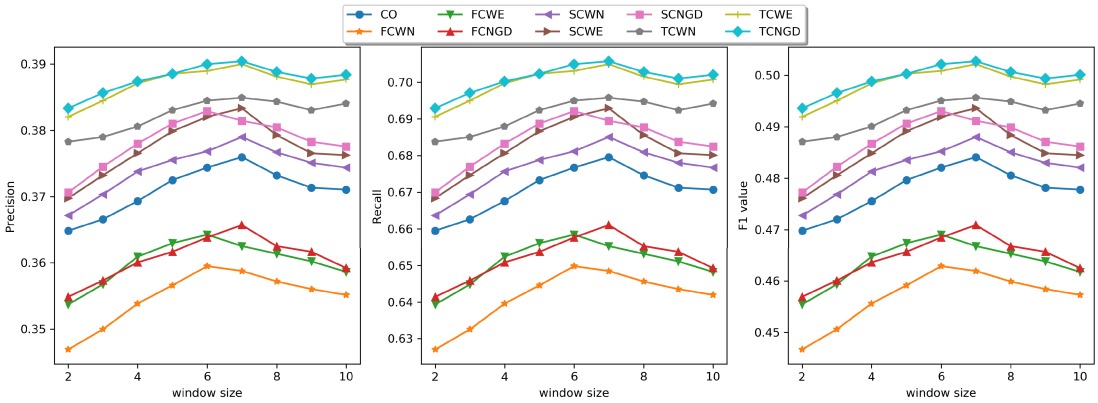

FIGURE 4. On the Inspec dataset, Precision, Recall, and F1 values change with the window size.

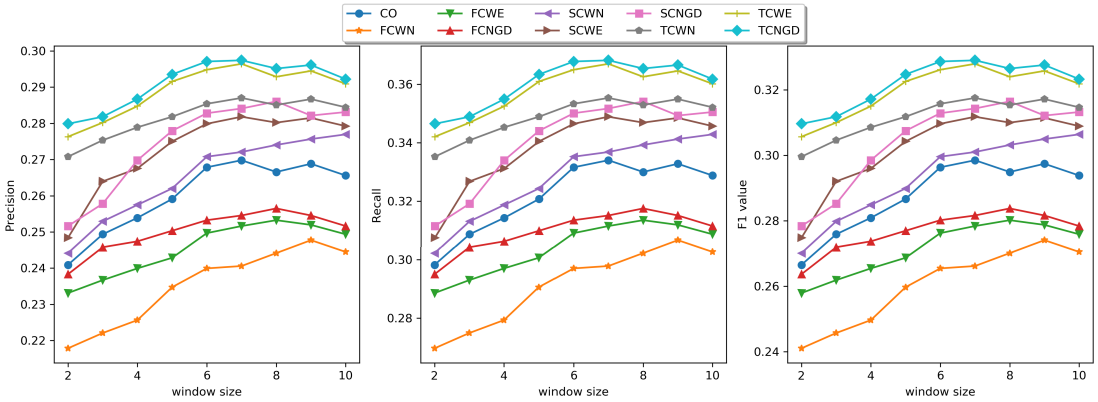

FIGURE 5. On the DUC01 dataset, Precision, Recall, and F1 values change with the window size.

candidate keywords selection method plays a key role in improving the effect of keywords extraction.

In Table 4, we compare the best performing TCNGD in the proposed combination methods with several strong baselines. As can be seen from the table, TCNGD outperforms all baselines on both datasets. For example, on DUC01 dataset, TCNGD achieves an F1-value of 0.329 as compared to 0.317 achieved by ExpandRank. From the table, we can also find that among all the baselines, the performance of ExpandRank is the best. ExpandRank adds neighborhood knowledge to adjust the weight of edges occurred in the slide window. While in TCNGD, we use the semantic relationship between words based on NGD to add new edges and change the structure of the word graph. Thus, we can calculate the weight of the words in the graph more comprehensively.

By comparing the results from methods Co, FCWN, FCWE and FCNGD. The P, R and F1 values of CO are bigger than other three methods. These four methods create the same word graph structure, the difference between them is the weight of edges. We assign weight of edges in CO method according to the number of co-occurrence of words in the sliding window, and the other three methods use the formula (3) to calculate the weight for edges. According to the results, we verify that the co-occurrence relationship between words in a document is stronger than the semantic relationships. As long as two words occur in the same sliding window, there must be a strong relationship between the two words, but sometimes the semantic distance can not express this strong relationship very well.

According to all the results in Table 3 and Table 4 that we get from experiments, we can see that if we only use word co-occurrence relations or semantic relations alone, we can not achieve the best results. But when the two relationships are combined, the effect of keywords extraction is improved.

D. PARAMETER ANALYSIS

The parameters we used in the proposed methods are d in equation (1), λ in equation (5) and window size w. For all methods, the value of d is set to 0.85. Next, we will explore the influence of the parameters w and λ on the keyword extraction results.

1) RELATIVE CONTRIBUTIONS OF CO-OCCURRENCE AND SEMANTIC RELATIONSHIPS ON KEYWORDS EXTRACTION

According to equation (5), it can be found that the final score of words is the fusion of two scoring results, in which λ controls the proportion of the two scores in the final score. In order to investigate the effect of λ on keywords extraction, we first change the final score of words by setting different value of λ, then extract keywords according to words score,

and finally compare the effect of keywords extraction under different values of λ according to F1 value.

Figure 2 and Figure 3 show the F1 value change with λ on Inspec and DUC2001, respectively. Here, the window size w set to 6. As can be seen from the figures, when the two scoring results are fused, the effect of keyword extraction is improved. At the same time, it is found that when the value of λ is between 0.5 and 0.7, the maximum values of F1 are obtained on both datasets.

2) EXPLORE THE EFFECT OF WINDOW SIZE ON KEYWORDS EXTRACTION

Next we will explore the impact of window size on CO and three combination methods of keywords extraction. Here we set the window size from 2 to 10. The results of the two datasets are shown in Figure 4 and Figure 5, respectively.

According to Figure 4 and Figure 5, we can easily find that those FCWN, FCWE and FCNGD methods that use the first combination get the lowest results in precision, recall and F1 values. Again demonstrate that in a word graph, the neighboring relation is more import than semantic relation between words. We also can find that the *CO*,*F**,*S** methods are greatly affected by the change of window size, while the P, R and F1 values in *T** methods are less affected by the change of window size. This is because when using *CO, F** and *S** methods, the structure of a graph depends entirely on the co-occurrence of words, so changing the size of the window will change the topological structure of the graph. However, in the *T** methods, the edges of the graph not only depend on the co-occurrence relationship of words in window, but also on the semantic relationships between words that do not appear in window, so changing the size of the window has little influence on the structure of graph, so it has little impact on keywords extraction.

V. CONCLUSION

In this paper, we compare and discuss the effects of keywords extraction methods based on the Inspec and DUC01 datasets. First, our methods are verified that the combination of co-occurrence and semantic relationships between words can improve the effectiveness of keywords extraction. Using the semantic relationships between words, which breaks the restriction we link two nodes in the graph with an edge must appear in the same window, so that any two related words in the word graph can be connected by an edge, which makes the important words in the graph more prominent. Second, through the experiments on two datasets, we also demonstrate that the co-occurrence relationship between words in a document is stronger than semantic relationships between words.

In the future, we will further enhance the effectiveness of keywords extraction from two aspects: the generation of candidate keywords and word scoring. For the generation of candidate keywords, we will study the discriminant methods of stopwords to generate a specific stopwords list for keywords extraction. For word scoring, we will try different word embedding methods to measure word distance such as FastText [33], BERT [34]. And give more prior knowledge for words to highlight the importance of words in documents, such as the location of words, topical information, and so on. At the same time, we will apply our methods to different document types, such as scholarly papers and auditing documents.

REFERENCES

[1] S. Menaka and N. Radha, "Text classification using keyword extraction technique," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 12, pp. 734–740, 2013.

[2] R. Al-Hashemi, "Text summarization extraction system (TSES) using extracted keywords," *Int. Arab J. Technol.*, vol. 1, no. 4, pp. 164–168, 2010.

[3] X. Jiang, P. Hu, L. Hou, and X. Wang, "Improving pointer-generator network with keywords information for Chinese abstractive summarization," in *Proc. Int. Conf. Natural Lang. Process. Chin. Comput.* Cham, Switzerland: Springer, 2018, pp. 464–474.

[4] G. Xylomenos, E. Zafeiratos, and M. Prokopakis, "Keyword-based information retrieval for the WoT," in *Proc. 4th ACM/IEEE Symp. Edge Comput.*, Nov. 2019, pp. 407–412.

[5] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Morristown, NJ, USA, 2003, pp. 216–223, doi: 10.3115/1119355.1119383.

[6] Mihalcea, Rada, and Paul Tarau, "Textrank: Bringing order into text," in *Proc. Conf. on Empirical Methods Natural Lang. Process.*, 2004, pp. 404–411.

[7] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," *Text Mining*, vol. 14. Chichester, U.K.: Wiley-Blackwell, 2010, pp. 1–20, doi: 10.1002/9780470689646.ch1.

[8] Z. Liu, P. Li, Y. Zheng, and M. Sun, "Clustering to find exemplar terms for keyphrase extraction," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2009, pp. 257–266.

[9] Y.-F.-B. Wu, Q. Li, R. S. Bot, and X. Chen, "Domain-specific keyphrase extraction," in *Proc. 14th ACM Int. Conf. Inf. Knowl. Manage.*, Bremen, Germany, 2005, pp. 283–284.

[10] C. Caragea, F. A. Bulgarov, A. Godea, and S. Das Gollapalli, "Citation-enhanced keyphrase extraction from research papers: A supervised approach," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1435–1446.

[11] X. Wan and J. Xiao, "Single document keyphrase extraction using neighborhood knowledge," in *Proc. AAAI*, vol. 8, Jul. 2008, pp. 855–860.

[12] F. Boudin, "A comparison of centrality measures for graph-based keyphrase extraction," in *Proc. Int. Joint Conf. Natural Lang. Process. (IJCNLP)*, Nagoya, Japan, Oct. 2013, pp. 834–838.

[13] G. Tsatsaronis and I. K. Varlamis, "SemanticRank: Ranking keywords and sentences using semantic graphs," in *Proc. 23rd Int. Conf. Comput. Linguistics*, Aug. 2010, pp. 1074–1082.

[14] M. T. Khan, Y. Ma, and J.-J. Kim, "Term ranker: A graph-based re-ranking approach," in *Proc. FLAIRS Conf.*, 2016.

[15] R. Wang, W. Liu, and C. McDonald, "Corpus-independent generic keyphrase extraction using word embedding vectors," in *Proc. Softw. Eng. Res. Conf.*, vol. 39, 2014, pp. 1–8.

[16] J. Martinez-Romo, L. Araujo, and A. Duque Fernandez, "SemGraph: Extracting keyphrases following a novel semantic graph-based approach," *J. Assoc. Inf. Sci. Technol.*, vol. 67, no. 1, pp. 71–82, Jan. 2016.

[17] E. Papagiannopoulou and G. Tsoumakas, "Local word vectors guiding keyphrase extraction," *Inf. Process. Manage.*, vol. 54, no. 6, pp. 888–902, Nov. 2018, doi: 10.1016/j.ipm.2018.06.004.

[18] C. Florescu and C. Caragea, "PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 1105–1115.

[19] S. K. Biswas, M. Bordoloi, and J. Shreya, "A graph based keyword extraction model using collective node weight," *Expert Syst. Appl.*, vol. 97, pp. 51–59, May 2018, doi: 10.1016/j.eswa.2017.12.025.

[20] A. Bellaachia and M. Al-Dhelaan, "NE-rank: A novel graph-based keyphrase extraction in Twitter," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.*, Dec. 2012, pp. 372–379.

[21] Z. Liu, W. Huang, Y. Zheng, and M. Sun, "Automatic keyphrase extraction via topic decomposition," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2010, pp. 366–376.
[22] A. Bougouin, F. Boudin, and B. Daille, "TopicRank: Graph-based topic ranking for keyphrase extraction," in *Proc. IJCNLP*, 2013.
[23] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford coreNLP natural language processing toolkit," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics, Syst. Demonstrations*, 2014, pp. 55–60.
[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: http://arxiv.org/abs/1301.3781
[25] V. Mala and D. K. Lobiyal, "Semantic and keyword based Web techniques in information retrieval," in *Proc. Int. Conf. Comput., Commun. Autom. (ICCCA)*, Apr. 2016, pp. 23–26.
[26] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[27] R. L. Cilibrasi and P. M. B. Vitanyi, "The Google similarity distance," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 370–383, Mar. 2007.
[28] S. Beliga, A. Meštrović, and S. Martinčić-Ipšić, "An overview of graph-based keyword extraction methods and approaches," *J. Inf. Organizational Sci.*, vol. 39, no. 1, pp. 1–20, 2015.
[29] S. Duari and V. Bhatnagar, "SCAKE: Semantic connectivity aware keyword extraction," *Inf. Sci.*, vol. 477, pp. 100–117, Mar. 2019.
[30] E. Papagiannopoulou and G. Tsoumakas, *A Review of Keyphrase Extraction*. Hoboken, NJ, USA: Wiley, 2019
[31] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the Web," in *Proc. WWW*, 1999.
[32] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.
[33] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, Dec. 2017.
[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*. [Online]. Available: http://arxiv.org/abs/1810.04805
[35] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[36] Y. Zhang, M. Tuo, Q. Yin, L. Qi, X. Wang, and T. Liu, "Keywords extraction with deep neural network model," *Neurocomputing*, vol. 383, pp. 113–121, Mar. 2020.
[37] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, and Y. Chi, "Deep keyphrase generation," 2017, *arXiv:1704.06879*. [Online]. Available: http://arxiv.org/abs/1704.06879
[38] D. Suleiman, A. A. Awajan, and W. Al Etaiwi, "Arabic text keywords extraction using Word2vec," in *Proc. 2nd Int. Conf. New Trends Comput. Sci. (ICTCS)*, Oct. 2019, pp. 1–7.
[39] D. A. Vega-Oliveros, P. S. Gomes, E. E. Milios, and L. Berton, "A multi-centrality index for graph-based keyword extraction," *Inf. Process. Manag.*, vol. 56, Nov. 2019, Art. no. 102063.
[40] W. D. Abilhoa and L. N. Castro, "TKG: A graph-based approach to extract keywords from tweets," *Proc. DCAI*, 2014, pp. 425–434.
[41] R. Wang, W. Liu, and C. McDonald "Using word embeddings to enhance keyword identification for scientific publications," *Databases Theory and Applications* (Lecture Notes in Computer Science), vol. 9093, M. Sharaf, M. Cheema, and J. Qi, Eds. Cham, Switzerland: Springer. 2015.

XIANGKE MAO was born in 1992. He is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Harbin Engineering University, China. His main research interests include text summarization and keywords extraction.

SHAOBIN HUANG was born in 1965. He is currently a Professor with the College of Computer Science and Technology, Harbin Engineering University, China. His research interests include data mining, natural language processing, and machine learning.

RONGSHENG LI was born in 1991. He is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Harbin Engineering University, China. His research interests include natural language processing and machine learning.

LINSHAN SHEN was born in 1978. He is currently a Lecturer with the College of Computer Science and Technology, Harbin Engineering University, China. His research interests include data mining and recommendation systems.