IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 20, 2020, accepted September 20, 2020, date of publication September 29, 2020, date of current version October 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3027567

An Empirical Study of TextRank for Keyword Extraction

MINGXI ZHANG, XUEMIN LI, SHUIBO YUE, AND LIUQIAN YANG
College of Communication and Art Design, University of Shanghai for Science and Technology, Shanghai 200093, China

Corresponding author: Mingxi Zhang (mingxizhang10@fudan.edu.cn)

This work was supported in part by the National Science Foundation of China under Grant 62002225, in part by the Natural Science Foundation of Shanghai under Grant 16ZR1422800, and in part by the Training Project of University of Shanghai for Science and Technology under Grant 16HJPY-QN04.

ABSTRACT As a typical keyword extraction technology, TextRank has been used in a wide variety of commercial applications, including text classification, information retrieval and clustering. In these applications, the parameters of TextRank, including the co-occurrence window size, iteration number and decay factor, are set roughly, which might affect the effectiveness of returned results. In this work, we conduct an empirical study on TextRank, towards finding optimal parameter settings for keyword extraction. The experiments are done in Hulth2003 and Krapivin2009 datasets, which are two real datasets. We first remove the stop word by an open published English stop word list XPO6. And then, we extract the word stems by Porter Stemmer. Porter Stemmer is a tool which can find the stems of words with multiple variants, discard redundant information, strengthen the filtering effect, and extract the effective features of the text fully. We carry out extensive experiments to evaluate the effects of the parameters to keywords extraction, and evaluate the effectiveness of corresponding results by Precision, Recall and Accuracy. Experimental results show that TextRank shows the best performance when setting co-occurrence window size $w = 3$, iteration number $t = 20$, decay factor $c = 0.9$ and rank $k = 10$ respectively, and the results are independent of the text length.

INDEX TERMS Keyword extraction, Porter Stemmer, TextRank, PageRank.

I. INTRODUCTION

Keyword extraction aims to find the keywords in a document, which can be adopted to capture the topical words from papers, web pages and other documents [1]. Keyword extraction has become an increasingly important task in practice applications, which arises in numerous applications including text classification, information retrieval and clustering. These applications usually require an effective keyword extraction function for achieving a better system performance. For example, in text classification, the returned result could be more accurate by identifying keywords effectively; and in search engines, the keywords contained in text data could find the expected web pages accurately.

Recently, some methods are devoted to generate keywords, *e.g.*, [2]–[5]. Among existing methods, TextRank [6] has been considered as one of the most essential noes due to following reasons. First, TextRank uses the relationship of vocabulary to sort the subsequent keywords based on the co-occurrence window, which can extract keywords from the text directly and reflect the semantic information between text units better. Second, TextRank can find the most informative items from the text by dividing the text into several components, such as words and sentences, and building a text graph between the components. Third, TextRank has been widely applied to real applications, including identifying web content credibility [7], extract a multi-document summary [8] and text classification [9], and no learning or training process is required previously.

Despite the merits of TextRank, there is an unavoidable task when being employed in real applications, *i.e.*, there are lots of parameters that need to be tuned, such as the co-occurrence window size, iteration number and decay factor. In previous applications of TextRank, the parameters of TextRank are set roughly. For example, [10] set the co-occurrence window size as 7 when extracting the key phrases from Chinese news articles while [11] set the window size as 2 when building text graph using the semantic unit of

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

words. Practically, different parameter settings might lead to different results. Therefore, it is a non-trivial task to examine the effects of different parameter settings in real text data.

In this paper, we conduct an empirical study on TextRank, towards finding an optimal parameter setting for keyword extraction. The experiments are done in Hulth2003 [12] and Krapivin *et al.* in 2009 [13] datasets. In the text preprocessing stage, we remove the stop word by an open published English stop word list XPO6. And then, we extract the word stems by Porter Stemmer, which is an open published tool for finding the stems of words with multiple variants, discard redundant information, strengthen the filtering effect, and extract the effective features of the text fully. We test the effectiveness by changing parameter settings, including the co-occurrence window size, iteration number, decay factor and text length. And evaluate the effectiveness by Precision, Recall and Accuracy. The main contributions of this paper are summarized as follows.

1) We conduct an experimental study on TextRank to test the effectiveness for keywords extraction, towards finding optimal parameter setting for keyword extraction. The experimental study is important to the applications of keyword extraction, which would provide references to users and help them selecting suitable parameters for achieving better system performance.
2) To extract keywords more accurately, we design a text preprocessing algorithm to remove the unnecessary information from the texts, *e.g.*, stop words and suffices, in which the stop words are removed by XPO6 and the word stems are extracted by Porter Stemmer.
3) The effectiveness of experimental results is evaluated by Precision, Recall and Accuracy, and experimental results show that TextRank shows the best performance when setting co-occurrence window size $w = 3$, iteration number $t = 20$, decay factor $c = 0.9$ and rank $k = 10$ respectively, and the results are independent of the text length.

The rest of the paper is organized as follows. Section II discusses the related work on keyword extraction. Section III gives a text preprocessing algorithm. In Section IV, we introduce TextRank model. Section V discusses the experimental setup. Section VI gives a detailed analysis on experimental results. The case studies are discussed in Section VII. Finally, the conclusions are discussed in Section VIII.

II. RELATED WORK

In recent years, lots of methods have been devoted for extracting keywords from texts. These methods can be divided into four categories: the graph-based method, statistical-based method, machine learning method and non-negative matrix factorization method.

Graph-based method finds words or phrases with important functions on the basis of the constructed text graph. TextRank is a graph-based method on keyword extraction. First, it defines the semantic links between text units as co-occurrence relationships. Second, the text graph of logical distribution characteristics is used to calculate the importance of text units. Third, it ignores the semantic relevance of words, context and auxiliary information when building semantic networks. However, the parameters of TextRank are too many, which might affect the effectiveness of keyword extraction, and existing work set the parameters are too rough. For example, [14] set the size of co-occurrence window as 2 when extracting professional terms on individual documents, while [15] sets the size of co-occurrence window as 5 and 10 to test the effectiveness of keyword extraction. Therefore, different parameter settings lead to different keywords extract results.

Recently, some derivatives of TextRank are proposed. BiKEA [2], [16] extracts keywords by building the parallel text network graph of two different languages. WS-Rank [17] uses graph-based keyword extractor to extracting keywords. Reference [18] proposes a graph-based ranking algorithm by treating Wikipedia as an external knowledge base for short text keywords extraction. Reference [19] improves TextRank based on a transfer matrix that is built by integrating TF-IDF and the average information entropy. Reference [20] builds a word collocation network for nouns and conducts a comparative study of English language network graphs. Reference [21] builds co-occurrence and grammar language network graphs for keyword extraction. Reference [22] uses centrality indicators for keyword extraction in text language network graphs. Reference [23] applies the concept of k-core on the text graph representation for single-document keyword extraction. Reference [24] uses higher-order structural features of word co-occurrence graph for keyword extraction. Reference [25] extracts keywords by using strong association rule mining, which puts three different node attributes (i.e. graph structure, node semantics and associations) into a single framework to enhance graph-based keywords extraction method.

Statistical-based methods extract keywords by using the statistical information of the words in the document. Typically, TF-IDF (term frequency-inverse document frequency) [3] adopts TF (term frequency) and IDF (inverse document frequency) to calculate weights of terms and uses weights to get the importance of terms in the document. Reference [26] uses KP-Miner to extract key phrases from both English and Arabic documents of varied length. Reference [27] uses the Markov Chains to create initial ranking for extracting keywords.

Machine learning methods treat words in document as candidate keywords that can be judged as keywords, and use classification learning algorithm or sequence labeling way. Reference [4] extracts candidate features from unstructured texts based on generating a sentence parse tree for each sentence in the input text. Reference [28] presents a two-stage spoken term detection (STD) method based on support vector machine (SVM). Reference [29] uses Maximum Entropy Partitioning (MEP) to obtain the top partition of distinctively high occurring keywords in each class.

Reference [30] proposes a document indexing method for keyword searching based on semi-Markov conditional random fields (semi-CRFs). Reference [31] presents two-stage evolutionary strategy for gene feature selection combining the genetic algorithm.

Non-negative matrix factorization (NMF) methods extract topics from document collections in form of bag of words. Reference [5] proposes a simplicial NMF-based unsupervised generic document summarization method, which generates better summaries with less repetition. Reference [32] proposes a feature extraction method via multi-view NMF with local graph regularization. Reference [33] uses sentence features and word embedding in sentence scoring to improve the quality of NMF. Reference [34] discovers topics for the short texts by utilizing a semantics-assisted non-negative matrix factorization (SeaNMF) model. Reference [35] proposes a framework to intelligently analyze Twitter data, which uses of NMF for extracting human-interpretable topics from tweets. Reference [36] improves the performance of NMF by using Non-negative Double Singular Value Decomposition (NNDSVD) method.

III. TEXT PREPROCESSING

The original text usually contains lots of unnecessary information, *e.g.*, stop words and suffices, which might affect the effectiveness of returned results when generating keywords. In the preprocessing stage, we need to remove the unnecessary information from texts before extracting keywords, which would help generating keywords more accurately.

A. REMOVE STOP WORDS

Removing stop words can improve the efficiency of the algorithm and reduce the size of the text index. In our research, stop words are removed from the text according to XPO6 [1] that is an open published English stop words list. Stop words are usually filtered from search queries because they return a lot of unnecessary information. As shown line 2 in Algorithm 1, we use list L to record the stop words. And then we check the words contained in the input text in line 7 to 5 of the algorithm. Specifically, for a word $S \in T$, we first get its lowercase s, and then break the loop if s belongs to the stop word list, which would not be outputted as a word in subsequent process.

B. EXTRACT WORDS STEM

In real text data, the words might have different variants. For example, the word ''make'' has kinds of variants, such as ''makes'', ''made'' and ''making''; and the word ''make'' also has variants, including ''used'', ''useful'', ''useless'' and ''using''. The variants of word would increase the redundant information of the text and hence reduce the accuracy of keyword extraction. Therefore, it is necessary to extract the word stems to ensure the accuracy of keyword extraction and reduce the complexity of text content.

[1]http://xpo6.com/download-stop-word-list

For the words that are not contained in the stop word list, we need to obtain their stems. The purpose of stemming is to bring variant forms of a word together, not to map a word onto its ''paradigm'' form, which would help reducing the affects different word variants for keyword extraction. As shown in line 7-10 in Algorithm 1, when s is not a stop word, we obtain its stem s', and then insert s' to list T'. After processing all words, we output the word list T' which contains the stems of the words in the texts. In our research, we use the Porter Stemmer [37] to extract word stems. Porter Stemmer is a tool which can find the stems of words with multiple variants, discard redundant information, strengthen the filtering effect, and extract the effective features of the text fully. Recently, Porter Stemmer has been widely used in real applications, *e.g.*, [38]–[42]. Porter Stemmer has kinds of versions of programming language, and we use the JAVA version of Porter Stemmer [2] in our experiment.

Algorithm 1 Text Preprocessing Procedure

Input:
Raw text T;
Output:
Processed text T';
1: Initialize list T' as null;
2: Building stop word list L by XPO6;
3: **for** $S \in T$ **do**
4: convert S to its lowercase s;
5: **if** $s \in L$ **then**
6: break;
7: **else**
8: $s' \leftarrow$ get the stem of s by Porter Stemmer;
9: insert s' to word list T';
10: **end if**
11: **end for**
12: **return** T'

IV. TextRank MODEL

TextRank uses a text graph to represent the relationship between terms for the given text. After building the text graph, the TextRank scores of words are calculated by using PageRank [43]. Initially, all the words contained in the text graph are treated as candidate keywords, and the goal of TextRank is to find the top-k most important words with highest PageRank scores and then treat them as the keywords.

A. BUILDING TEXT GRAPH

The text graph is defined as an undirected graph $G = (V, E)$ with V denoting the set of words and E denoting the set of edges between words. Specifically, the text graph is built by two steps. First, selecting the candidate keywords from the text and treat them as the nodes in V. Second, building edges between words within a given window size w. The edges of text graph are building by using a co-occurrence relation [44]

[2]https://tartarus.org/martin/PorterStemmer/java.txt

that is controlled by setting a sliding window size for the given text. Specifically, if two words v_i and v_j appear in a sentences in the text within window size w, we treat v_i and v_j as nodes and add them into V, and then we build an edge between v_i and v_j.

B. EXTRACTING KEYWORD

After obtaining text graph, the keywords are extracted as follows.

1) CALCULATE TextRank SCORE

The TextRank score of each node in the text graph is calculated by PageRank [43], which is defined as the PageRank score in the text graph. Formally, TextRank score of node v_i is defined as:

$$\mathrm{TR}(v_i) = (1-c) + c \sum_{j \in \mathrm{N}(v_i)} \frac{\mathrm{TR}(v_j)}{|N(v_j)|} \tag{1}$$

where $\mathrm{N}(v_i)$ neighbor sets of node v_i, and c is a decay factor, which is a constant between 0 and 1. Since the text graph is an undirected graph, so the in-neighbor set and out-neighbor set are not discriminated, and both of them are denoted as $N(v_i)$ uniformly.

The TextRank scores are calculated iteratively. We use $\mathrm{TR}_t(v_i)$ to represent the TextRank score of v_i at iteration t, which is calculated as follows. Initially, $\mathrm{TR}_t(v_i) = 1$; and for $t = 1, 2, \ldots$, $\mathrm{TR}_t(v_i)$ is calculated as:

$$\mathrm{TR}_t(v_i) = (1-c) + c \sum_{j \in \mathrm{N}(v_i)} \frac{\mathrm{TR}_{t-1}(v_j)}{|N(v_j)|} \tag{2}$$

Based on Equation (2), the TextRank scores for all nodes in the text graph can be calculated iteratively, and finally converges to a stable state.

2) OUTPUT KEYWORDS

After obtaining TextRank scores of the nodes in text graph, we select top-k most important words corresponding to the nodes with highest TextRank scores in text graph as the keywords, and then sort and output them.

V. EXPERIMENTAL SETTINGS

A. DATASETS

We use following two datasets to evaluate the performances of TextRank: (1) **Hulth2003**. The Hulth2003 [3] [12] dataset is a document summary dataset from the Inspec physical and engineering literature database, in which each document has two sets of keywords assigned: the manually controlled assigned keywords appear in the Inspec thesaurus instead of document, and the uncontrolled keywords, which are freely assigned by the editors; (2) **Krapivin2009**. Krapivin2009 [4] [13] dataset has 2,304 full papers from the computer science domain, which are published by ACM in the period between 2003 and 2005, and each paper has keywords originally labeled by the authors and verified by the reviewers. In the experiments, we extract the abstracts of papers from datasets, and use user-assigned keywords as the ground truth.

Hulth2003 and Krapivin2009 have been widely adopted for evaluating the performance of keyword extraction algorithms, *e.g.*, [45]–[48] use Hulth2003 as a test sample for keyword extraction, and Krapivin2009 is used by [46]–[48]. Both of them are representative datasets for keyword extraction, which contain papers with different topics and the abstracts of the papers have kinds of text lengths. And the user-assigned keywords contained in the ground truth can be used for evaluating the effectiveness of experimental results. Therefore, the results in Hulth2003 and Krapivin2009 are representative, and the obtained optimal parameter settings can be extended to other datasets of this kind.

B. EVALUATION AND COMPARISONS

For each dataset, we randomly pick 100 papers and extract keywords from the abstract of each papers by TextRank. The effectiveness is evaluated by Precision [49], Recall [50] and Accuracy [51]. Specifically, Precision is the fraction of keywords in the returned top-k word list, that is defined as:

$$\text{Precision} = \frac{|N \cap M|}{|M|} \tag{3}$$

and Recall is the fraction of the keywords appears in the top-k word list, that is defined as:

$$\text{Recall} = \frac{|N \cap M|}{|N|} \tag{4}$$

where N is the set of all the keywords of the text, which is the set of keywords in the ground truth, and M is the set of all words returned in top-k word list.

Accuracy is defined as the fraction of the words that are labelled correctly in the text, that is defined as:

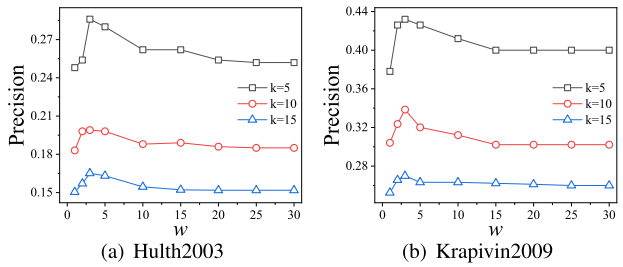
$$\text{Accuracy} = \frac{|X|}{|Y|} \tag{5}$$

where X is the set of the words that are correctly labelled, and Y is the set of candidate words that are labelled correctly and incorrectly. Given a text, the keywords contained in the top-k word list are predicted as the keywords, that are labelled as "+"; and other words are not predicted as keywords, which are labeled as "−". The datasets originally contain in the keyword list of ground truth, which can be used to validate whether the words are correctly labelled. Specifically, a word with label "+" is correctly labelled if it is not contained in ground truth, and a word with label "−" is correctly labelled if it is not contained in the ground truth.

We test the effectiveness of TextRank for different parameters, which includes the co-occurrence window size w, iteration number t, decay factor c, rank k and text length. And in each dataset, we average Precision, Recall and Accuracy scores of different texts for evaluating the effectiveness unless otherwise specified.

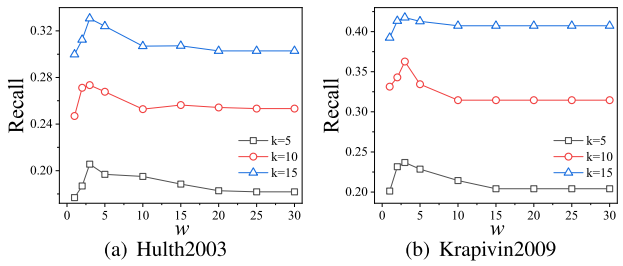
[3] https://www.researchgate.net/publication/261860598_Hulth2003tar
[4] http://dit.unitn.it/~krapivin

Our experiments are conducted on Windows 10 with Intel(R) Core(TM) i5 - 4210M CPU@2.60 GHz 2.60 GHz and 4 GB RAM. All algorithms are implemented in Java and compiled by using Eclipse 2018.



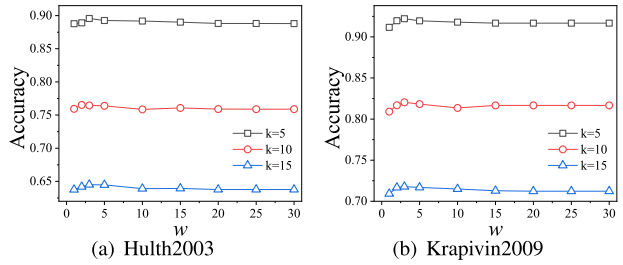
(a) Hulth2003 (b) Krapivin2009

FIGURE 1. Precision on varying w.



(a) Hulth2003 (b) Krapivin2009

FIGURE 2. Recall on varying w.



(a) Hulth2003 (b) Krapivin2009

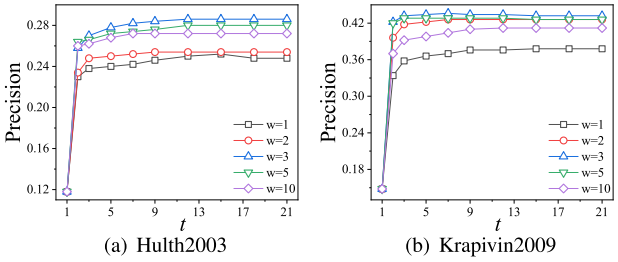
FIGURE 3. Accuracy on varying w.

VI. EXPERIMENTAL RESULTS

A. CO-OCCURRENCE WINDOW SIZE

Figure 1(a) and 1(b) show the Precision on varying co-occurrence window size w in Hulth2003 and Krapivin2009 respectively, where $k = 5, 10, 15$. The curves for different k increase as w increases from 1 to 3, since more latent connections are found as window size become bigger. As w continues to increase, the Precision scores on different k drop rapidly and then become stable. This is because the semantic relationship between words become weak as co-occurrence window size increases. As shown in Figure 2 and Figure 3, the results on Recall and Accuracy are similar. Differently, the curves in Figure 2 increase as w increases, and then decrease after $w = 3$, since a bigger co-occurrence window size would lead to bigger precision loss. In Figure 3, the change of the curves is not so obvious due to the big range of Y-axis, but we still observe that TextRank reaches the highest point when $w = 3$.

The results show that the co-occurrence window affects the performance of TextRank. According the results, we suggest that the window size is set as 3 when users extract keywords.

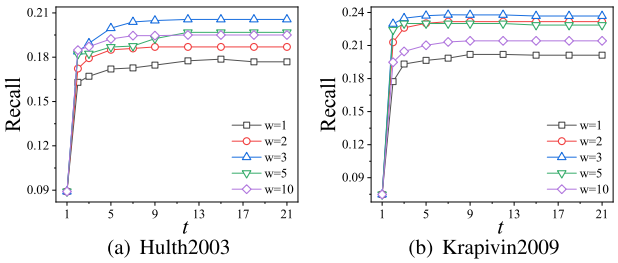


(a) Hulth2003 (b) Krapivin2009

FIGURE 4. Precision on varying t.

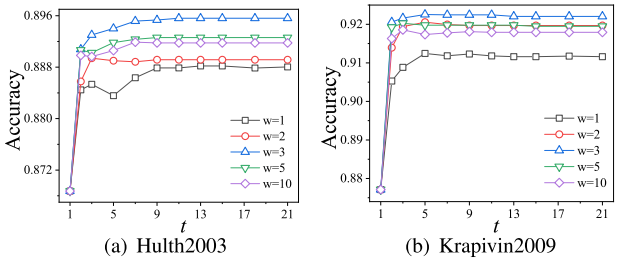
B. ITERATION NUMBER

Figure 4(a) and 4(b) show the Precision on varying iteration number t. In both datasets, the Precision scores increase as t increases and then become stable after $t = 10$. There is little change after $t = 20$ since TextRank is based on PageRank model which can gradually converge to a stable state as iteration t increases. The curves of $w = 3$ is higher than others, which is consistent to the results in Figure 4(a) and 4(b). Similar results are can be found in Figure 5 and 6, which can be explained similarly.



(a) Hulth2003 (b) Krapivin2009

FIGURE 5. Recall on varying t.



(a) Hulth2003 (b) Krapivin2009

FIGURE 6. Accuracy on varying t.

Generally, according to the results, the iteration number should be bigger 10 when users extract keywords, and a promising parameter setting is suggested to be 20 to ensure the stability of keyword extraction.

C. DECAY FACTOR

Figure 7(a) and 7(b) show the Precision on varying decay factor c. We find that the curves at different w increase as c increases from 0 to 0.9, and then show a downward trend after $c = 0.9$. In general, when $c = 0.9$, the Precision scores for

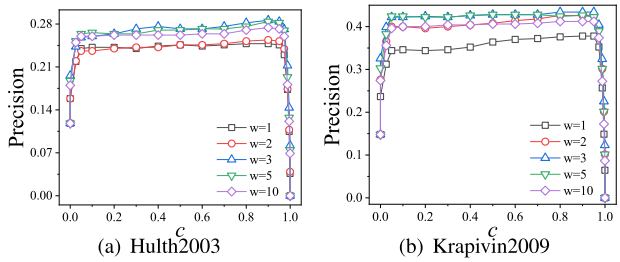


FIGURE 7. Precision on varying c.

each dataset arrives at the highest value. At the same decay factor, which the curve of $w = 3$ is higher than others. Figure 8(a) and 8(b) show the Recall of TextRank on varying decay factor c in Hulth2003 and Krapivin2009 respectively. The values of Recall decreases significantly after $c = 0.95$ until it drops to 0 at $c = 1$. The curve of $w = 3$ is higher than others at the same decay factor. Similar curve change can also be found in Figure 9.

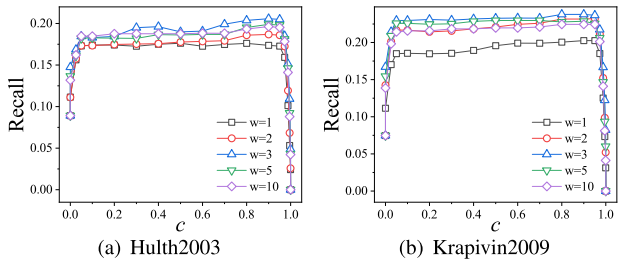


FIGURE 8. Recall on varying c.

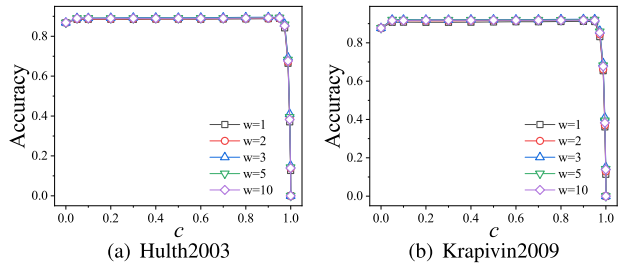


FIGURE 9. Accuracy on varying c.

Generally, the evaluation results are relatively higher than other values when the decay factor changes in the range of $c = 0.75$ to 0.95. Based on the results, we recommend that the selection range of the decay factor is $c = 0.9$, and in the following experiments, we also set c as this value to test the performance on other parameters for TextRank.

D. RANK IN RETURNED LISTS

Figure 10(a) and 10(b) show the Precision on varying rank k. In both datasets, the Precision scores decrease as k increases. And after $k = 70$, the Precision becomes stable. This is because the words with higher rank are more tend to be the keywords, and the words with lower rank should be not keywords relatively. At each rank k, the curves of different w

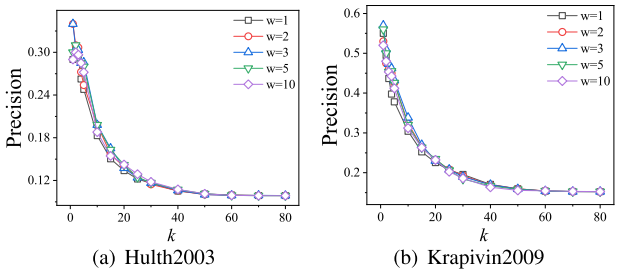


FIGURE 10. Precision on varying k.

are seems overlapped since the range Y-axis is set too bigger for observing the minor differences. Similar curve changes can also be found in Figure 12. Differently, the Recall scores show a downward trend as k increases in Figure 11, since more keywords are returned as k increases, which consequently increases the Recall scores. In practice, users usually interest to only few words when extracting keywords, and the words in lower rank usually not so relevant to the topic of the document. Therefore, we suggest that k is set as 10 to ensure the effectiveness of keyword extraction.

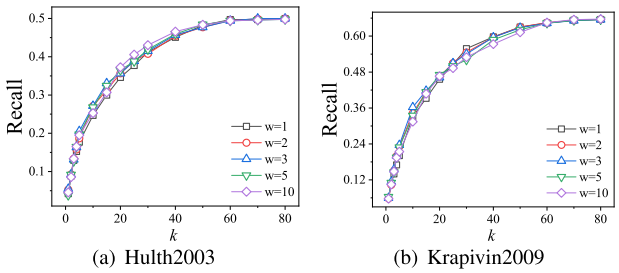


FIGURE 11. Recall on varying k.

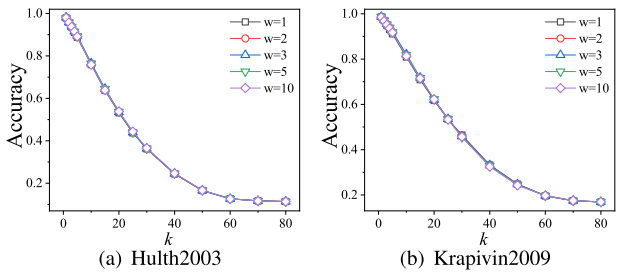


FIGURE 12. Accuracy on varying k.

E. TEXT LENGTH

Figure 13, 14 and 15 show the Precision, Recall and Accuracy scores of different documents, where each plot represents a document, and X-axes represent the text lengths and Y-axes represent the Precision, Recall and Accuracy scores respectively. In both datasets, the plots on Precision and Recall scores are randomly distributed on different text lengths, which demonstrate that the results are affected little by the text lengths. In Figure 13 and 14, we find that the plots form several horizontal lines. This is because each spot

TABLE 1. Selected texts from Hulth2003 and Krapivin2009.

Hulth2003	Fuzzy polynomial neural networks: hybrid architectures of fuzzy modeling. We introduce a concept of fuzzy polynomial neural networks (FPNNs), a hybrid modeling architecture combining polynomial neural networks (PNNs) and fuzzy neural networks (FNNs). The development of the FPNNs dwells on the technologies of computational intelligence (CI), namely fuzzy sets, neural networks, and genetic algorithms. The structure of the FPNN results from a synergistic usage of FNN and PNN. FNNs contribute to the formation of the premise part of the rule-based structure of the FPNN. The consequence part of the FPNN is designed using PNNs. The structure of the PNN is not fixed in advance as it usually takes place in the case of conventional neural networks, but becomes organized dynamically to meet the required approximation error. We exploit a group method of data handling (GMDH) to produce this dynamic topology of the network. The performance of the FPNN is quantified through experimentation that exploits standard data already used in fuzzy modeling. The obtained experimental results reveal that the proposed networks exhibit high accuracy and generalization capabilities in comparison to other similar fuzzy models (PNNs) and fuzzy neural networks (FNNs). The development of the FPNNs dwells on the technologies of computational intelligence (CI), namely fuzzy sets, neural networks, and genetic algorithms. The structure of the FPNN results from a synergistic usage of FNN and PNN. FNNs contribute to the formation of the premise part of the rule-based structure of the FPNN. The consequence part of the FPNN is designed using PNNs. The structure of the PNN is not fixed in advance as it usually takes place in the case of conventional neural networks, but becomes organized dynamically to meet the required approximation error. We exploit a group method of data handling (GMDH) to produce this dynamic topology of the network. The performance of the FPNN is quantified through experimentation that exploits standard.
Krapivin2009	This paper discusses the comprehensive performance profiling, improvement and benchmarking of a Computational Fluid Dynamics code, one of the Grand Challenge applications, on three popular multiprocessors. In the process of analyzing performance we considered language, compiler, architecture, and algorithmic changes and quantified each of them and their incremental contribution to bottom-line performance. We demonstrate that parallelization alone cannot result in significant gains if the granularity of parallel threads and the effect of parallelization on data locality are not taken into account. Unlike benchmarking studies that often focus on the performance or effectiveness of parallelizing compilers on specific loop kernels, we used the entire CFD code to measure the global effectiveness of compilers and parallel architectures. We probed the performance bottlenecks in each case and derived solutions which eliminate or neutralize the performance inhibiting factors. The major conclusion of our work is that overall performance is extremely sensitive to the synergetic effects of compiler optimizations, algorithmic and code tuning, and architectural idiosyncrasies.

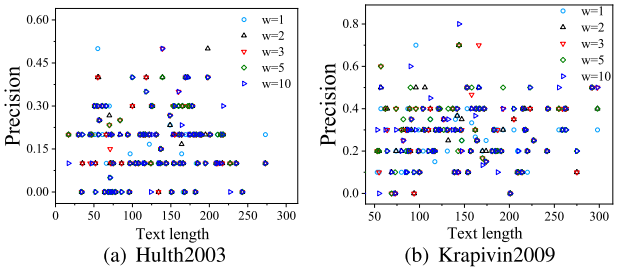


FIGURE 13. Precision on varying text length.

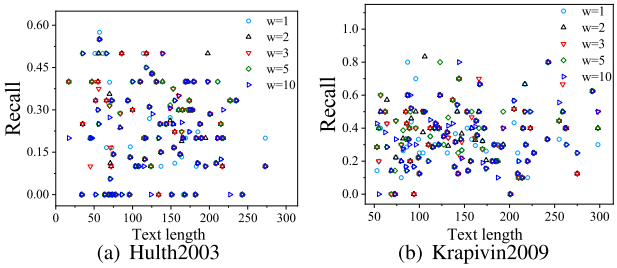


FIGURE 14. Recall on varying text length.

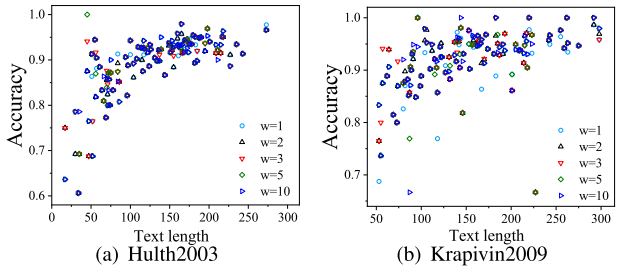


FIGURE 15. Accuracy on varying text length.

represents only a single document, and we returned only top 10 words for each document. When calculating Precision, the numerator can be only an integer from 1 to 10, and we choose 100 documents from each dataset, so lots of documents correspond to a same integer. And the denominator is 10, so the Precision score can be only one of the values in $\{0.1, 0.2, \ldots, 1.0\}$. Therefore, lots of documents correspond to a same Precision, which forms the horizontal lines in the figures. When calculating Recall, the horizontal lines are not so evident as Precision. This is because the denominator of Recall is the number of keywords contained in the ground truth. Though most of the denominators is less than 10, their values are not uniquely defined. Therefore, only a few of the documents correspond to a same Recall, so the horizontal lines are not so evident as Precision. In Figure 15, the Accuracy scores of documents with bigger text lengths are higher than the documents with smaller text lengths. This is because there are more words in the longer texts, and consequently the ratio of the words beyond the top-10 lists becomes more relatively. Such words are uniformly labelled as "-", which naturally increase the words of correctly labelled when calculating Accuracy.

Generally, the text lengths of the documents give little affect to the effectiveness, and consequently we conclude that the results of TextRank are independent of the text length.

VII. CASE STUDIES

Next we give some case studies by making use of TextRank on different parameter settings. Table 1 shows two texts selected from Hulth2003 and Krapivin2009, which are taken as examples to explore how different parameters effect on TextRank when extracting keywords.

Table 2 shows the top 10 keywords on varying co-occurrence window size w in Hulth2003, where $c = 0.9$ and $t = 20$. As shown in the returned lists, the rankings corresponding to different co-occurrence window sizes are different. For example, at rank $k = 6$, the keywords are "FNNs" when $w = 3, 5, 10$, and "results" and "FPNNs" when $w = 1, 2$; at rank $k = 7$, the keywords are "dynamically" when $w = 5, 10$, "FNNs", and "exploit" and "FPNNs" when $w = 1, 2, 3$; and at rank $k = 2, 3, 4, 5$, there are also

TABLE 2. Case studies for top-10 keyword extraction on Hulth2003.

k	$w=1$	$w=2$	$w=3$	$w=5$	$w=10$
1	models	models	models	models	models
2	data	neural	neural	neural	neural
3	PNNs	data	data	data	PNNs
4	FPNNs	experimentation	experimentation	PNNs	FPNNs
5	exploit	PNNs	polynomial	FPNNs	exploit
6	results	FPNNs	FNNs	FNNs	FNNs
7	FNNs	exploit	FPNNs	dynamically	dynamically
8	dynamically	dynamically	dynamically	structure	structure
9	fuzzy	fuzzy	fuzzy	fuzzy	fuzzy
10	network	network	network	network	network

some differences when window sizes are different. We also find that, when $w = 3$, the returned results are relatively better than others. For example, when $w = 1, 2, 5, 10$, the returned rankings contain irrelevant words, *e.g.*, "exploit", "PNNs" and "FPNNs" when $k = 5$, which are not contained in the ranking of $w = 3$. Similar cases are shown in Table 3 and can be discussed similarly.

TABLE 3. Case studies for top-10 keyword extraction on Krapivin2009.

k	$w=1$	$w=2$	$w=3$	$w=5$	$w=10$
1	conclusion	effectiveness	effectiveness	effectiveness	effectiveness
2	effectiveness	quantified	neutralize	code	code
3	code	code	code	extremely	performance
4	performance	performance	performance	performance	benchmarking
5	benchmarking	benchmarking	measure	benchmarking	parallel
6	parallel	parallel	benchmarking	parallel	challenge
7	loop	compiler	parallel	solutions	compiler
8	discusses	algorithmic	incremental	eliminate	algorithmic
9	challenge	comprehensive	compiler	sensitive	architectural
10	compiler	computational	case	compiler	applications

TABLE 4. Case studies for top-10 keyword extraction on Hulth2003.

k	$t=1$	$t=2$	$t=3$	$t=5$	$t=10$
1	standard	models	models	models	models
2	technologies	neural	neural	neural	neural
3	dwells	data	data	data	data
4	reveal	experimentation	experimentation	experimentation	experimentation
5	data	PNNs	PNNs	PNNs	polynomial
6	conventional	FPNNs	FPNNs	FPNNs	FNNs
7	contribute	exploit	FNNs	FNNs	FPNNs
8	fixed	dynamically	dynamically	dynamically	dynamically
9	structure	fuzzy	fuzzy	fuzzy	fuzzy
10	intelligence	network	network	network	network

Table 4 shows the top 10 extracted keywords on different number of iterations t in Hulth2003, where $c = 0.9$ and $w = 3$. As iteration increases, the rankings become stable. For example, there rankings are different when $t = 1, 2, 3$, and become same when $t = 5, 10$. Similar cases are shown in Table 5, which can be discussed similarly.

TABLE 5. Case studies for top-10 keyword extraction on Krapivin2009.

k	$t=1$	$t=2$	$t=3$	$t=5$	$t=10$
1	contribution	effectiveness	effectiveness	effectiveness	effectiveness
2	data	code	code	code	neutralize
3	major	neutralize	performance	neutralize	code
4	loop	performance	measure	performance	performance
5	granularity	benchmarking	benchmarking	measure	measure
6	challenge	parallel	parallel	benchmarking	benchmarking
7	eliminate	incremental	global	parallel	parallel
8	sensitive	compiler	eliminate	incremental	incremental
9	entire	algorithmic	incremental	compiler	compiler
10	specific	architectural	compiler	case	case

Table 6 shows the top 10 extracted keywords on different decay factor c in Hulth2003, where $w = 3$ and $t = 20$. Though the rankings corresponding to different decay factors are different, the results are in a stable state generally. For example, at rank $k = 5$, the keywords are "PNNs" when $c = 0.1, 0.3, 0.5, 0.7$, and "polynomial" when $c = 0.9$;

TABLE 6. Case studies for top-10 keyword extraction on Hulth2003.

k	$c=0.1$	$c=0.3$	$c=0.5$	$c=0.7$	$c=0.9$
1	models	models	models	models	models
2	neural	neural	neural	neural	neural
3	data	data	data	data	data
4	experimentation	experimentation	experimentation	experimentation	experimentation
5	PNNs	PNNs	PNNs	PNNs	polynomial
6	FPNNs	FPNNs	FPNNs	FPNNs	FNNs
7	exploit	exploit	exploit	FNNs	FPNNs
8	dynamically	dynamically	dynamically	dynamically	dynamically
9	fuzzy	fuzzy	fuzzy	fuzzy	fuzzy
10	network	network	network	network	network

TABLE 7. Case studies for top-10 keyword extraction on Krapivin2009.

k	$c=0.1$	$c=0.3$	$c=0.5$	$c=0.7$	$c=0.9$
1	effectiveness	effectiveness	effectiveness	effectiveness	effectiveness
2	neutralize	neutralize	neutralize	neutralize	neutralize
3	code	code	code	code	code
4	performance	performance	performance	performance	performance
5	benchmarking	benchmarking	benchmarking	benchmarking	measure
6	parallel	parallel	parallel	measure	benchmarking
7	incremental	incremental	incremental	parallel	parallel
8	compiler	compiler	compiler	incremental	incremental
9	architectural	algorithmic	algorithmic	compiler	compiler
10	algorithmic	architectural	case	algorithmic	case

at rank $k = 7$, the keywords are "exploit" when $c = 0.1, 0.3, 0.5$, and "FNNs" and "FPNNs" when $c = 0.7, 0.9$. Similar cases can be found in Table 7. Generally, in current parameter settings, most of the returned keywords are relevant to the topics of the texts, and the differences are not so evident.

VIII. CONCLUSION

In this paper, we conduct an empirical study on TextRank to test the effectiveness on different parameter settings, including the size of co-occurrence window, iteration number, decay factor, rank in the returned lists and text length. The experiments are done in Hulth2003 and Krapivin2009 datasets, and the effectiveness is evaluated by Precision, Recall and Accuracy. Experimental results show that TextRank shows the best performance when setting co-occurrence window size $w = 3$, iteration number $t = 20$, decay factor $c = 0.9$ and rank $k = 10$ respectively, and the results are independent of the text length.

REFERENCES

[1] P. D. Turney, "Learning to extract keyphrases from text," *CoRR*, vol. cs.LG/0212013, pp. 1–2, Oct. 2002.
[2] C.-C. Huang, M. Eskenazi, J. Carbonell, L.-W. Ku, and P.-C. Yang, "Cross-lingual information to the rescue in keyword extraction," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics: Syst. Demonstrations*, Baltimore, MD, USA, 2014, pp. 1–6.
[3] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, Jan. 1988.
[4] M. Aman, A. bin Md Said, S. J. A. Kadir, and I. Ullah, "Key concept identification: A sentence parse tree-based technique for candidate feature extraction from unstructured texts," *IEEE Access*, vol. 6, pp. 60403–60413, 2018.
[5] N. K. Anh, N. K. Toi, and N. V. Linh, "An interpretable method for text summarization based on simplicial non-negative matrix factorization," in *Proc. 5th Symp. Inf. Commun. Technol.*, Hanoi, Vietnam, Dec. 2014, pp. 57–64.
[6] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proc. Emnlp*, 2004, pp. 404–411.
[7] B. Balcerzak, W. Jaworski, and A. Wierzbicki, "Application of TextRank algorithm for credibility assessment," in *Proc. IEEE/WIC/ACM Int. Joint Conferences Web Intell. (WI) Intell. Agent Technol. (IAT)*, Warsaw, Poland, Aug. 2014, pp. 451–454.

[8] C. Xiong, X. Li, Y. Li, and G. Liu, "Multi-documents summarization based on TextRank and its application in online argumentation platform," *Int. J. Data Warehousing Mining*, vol. 14, no. 3, pp. 69–89, Jul. 2018.

[9] S. Song, Z. Wang, S. Xu, S. Ni, and J. Xiao, "A novel text classification approach based on Word2vec and TextRank keyword extraction," in *Proc. IEEE 4th Int. Conf. Data Sci. Cyberspace (DSC)*, Hangzhou, China, Jun. 2019, pp. 536–543.

[10] W. Liang, C. Huang, M. Li, and B. Lu, "Extracting keyphrases from chinese news articles using textrank and Query log knowledge," in *Proc. 23rd Pacific Asia Conf. Lang., Inf. Comput. (PACLIC)*, Hong Kong, Dec. 2009, pp. 733–740.

[11] M. Masudur Rahman and C. K. Roy, "TextRank based search term identification for software change tasks," in *Proc. IEEE 22nd Int. Conf. Softw. Anal., Evol., Reeng. (SANER)*, Montreal, QC, Canada, Mar. 2015, pp. 540–544.

[12] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Sapporo, Japan, Jul. 2003, pp. 1–5.

[13] M. Krapivin, A. Autayeu, M. Marchese, E. Blanzieri, and N. Segata, "Keyphrases extraction from scientific documents: Improving machine learning approaches with natural language processing," in *Proc. 12th Int. Conf. Asia–Pacific Digital Libraries (ICADL)*, Gold Coast, Australia, Jun. 2010, pp. 102–111.

[14] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," *Text Mining, Appl. Theory*, vol. 1, pp. 1–20, Mar. 2010.

[15] Z. Zhang, J. Petrak, and D. Maynard, "Adapted TextRank for term extraction: A generic method of improving automatic term extraction algorithms," *Procedia Comput. Sci.*, vol. 137, pp. 102–108, May 2018.

[16] C.-C. Huang, M.-H. Chen, and P.-C. Yang, "Bilingual keyword extraction and its educational application," in *Proc. 2nd Workshop Natural Lang. Process. Techn. Educ. Appl.*, Beijing, China, 2015, pp. 43–48.

[17] F. Yang, Y. Zhu, and Y. Ma, "Ws-rank: Bringing sentences into graph for keyword extraction," in *Proc. 18th Asia–Pacific Web Conf.*, Suzhou, China, Sep. 2016, pp. 474–477.

[18] W. Li and J. Zhao, "TextRank algorithm by exploiting wikipedia for short text keywords extraction," in *Proc. 3rd Int. Conf. Inf. Sci. Control Eng. (ICISCE)*, Jul. 2016, pp. 1–5.

[19] S. Pan, Z. Li, and J. Dai, "An improved TextRank keywords extraction algorithm," in *Proc. ACM Turing Celebration Conf.*, Chengdu, China, 2019, p. 131.

[20] S. Lahiri, S. R. Choudhury, and C. Caragea, "Keyword and keyphrase extraction using centrality measures on collocation networks," *CoRR*, vol. abs/1401.6571, pp. 3–5, Dec. 2014.

[21] F. Boudin, "A comparison of centrality measures for graph-based keyphrase extraction," in *6th Int. Joint Conf. Natural Lang. Process. (IJCNLP)* , Nagoya, Japan, Oct. 2013, pp. 834–838.

[22] N. Schluter, "Centrality measures for non-contextual graph-based unsupervised single document keyword extraction," in *Proc. TALN*, Marseille, France, Jul. 2014, pp. 455–460.

[23] F. Rousseau and M. Vazirgiannis, "Main core retention on graph-of-words for single-document keyword extraction," in *Proc. ECIR*, Vienna, Austria, Mar./Apr. 2015, pp. 382–393.

[24] Y. Chen, J. Wang, P. Li, and P. Guo, "Single document keyword extraction via quantifying higher-order structural features of word co-occurrence graph," *Comput. Speech Lang.*, vol. 57, pp. 98–107, Dec. 2019.

[25] H. Ma, S. Wang, M. Li, and N. Li, "Enhancing graph-based keywords extraction with node association," in *Proc. KSEM*, Athens, Greece, Aug. 2019, pp. 497–510.

[26] S. R. El-Beltagy and A. Rafea, "KP-miner: A keyphrase extraction system for english and arabic documents," *Inf. Syst.*, vol. 34, no. 1, pp. 132–144, Mar. 2009.

[27] B. Zyglarski and P. Bala, "Keywords extraction—Selecting keywords in natural language texts with Markov chains and neural networks," in *Proc. Int. Conf. Knowl. Manage. Inf. Sharing*, Valencia, Spain, Oct. 2010, pp. 315–321.

[28] K. Domoto, T. Utsuro, N. Sawada, and H. Nishizaki, "Spoken term detection using SVM-based classifier trained with pre-indexed keywords," *IEICE Trans. Inf. Syst.*, vol. E99.D, no. 10, pp. 2528–2538, 2016.

[29] S. Susan and J. Keshari, "Finding significant keywords for document databases by two-phase maximum entropy partitioning," *Pattern Recognit. Lett.*, vol. 125, pp. 195–205, Jul. 2019.

[30] H. Zhang, X.-D. Zhou, and C.-L. Liu, "Keyword spotting in handwritten chinese documents using semi-Markov conditional random fields," *Eng. Appl. Artif. Intell.*, vol. 58, pp. 49–61, Feb. 2017.

[31] R. M. Luque-Baena, D. Urda, M. Gonzalo Claros, L. Franco, and J. M. Jerez, "Robust gene signatures from microarray data using genetic algorithms enriched with biological pathway keywords," *J. Biomed. Informat.*, vol. 49, pp. 32–44, Jun. 2014.

[32] Z. Wang, X. Kong, H. Fu, M. Li, and Y. Zhang, "Feature extraction via multi-view non-negative matrix factorization with local graph regularization," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2015, pp. 3500–3504.

[33] N. T. T. Trang, L. T. Huong, and D. V. Hung, "Enhancing extractive summarization using non-negative matrix factorization with semantic aspects and sentence features," in *Proc. 8th Int. Symp. Inf. Commun. Technol.*, Nha Trang City, Viet Nam, 2017, pp. 78–83.

[34] T. Shi, K. Kang, J. Choo, and C. K. Reddy, "Short-text topic modeling via non-negative matrix factorization enriched with local word-context correlations," in *Proc. World Wide Web Conf.*, Lyon, France, 2018, pp. 1105–1114.

[35] G. Casalino, C. Castiello, N. Del Buono, and C. Mencar, "A framework for intelligent Twitter data analysis with non-negative matrix factorization," *Int. J. Web Inf. Syst.*, vol. 14, no. 3, pp. 334–356, Aug. 2018.

[36] A. Khurana and V. Bhatnagar, "Extractive document summarization using non-negative matrix factorization," in *Proc. DEXA*, Linz, Austria, Aug. 2019, pp. 76–90.

[37] M. F. Porter, "An algorithm for suffix stripping," *Program, Electron. Library Inf. Syst.*, vol. 40, no. 3, pp. 313–316, 1997.

[38] G. Jeh and J. Widom, "SimRank: A measure of structural-context similarity," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Alberta, QC, Canada, 2002, pp. 538–543.

[39] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn.*, Prague, Czech Republic, Jun. 2007, pp. 410–420.

[40] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *J. Artif. Intell. Res.*, vol. 37, pp. 141–188, Feb. 2010.

[41] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12 pp. 2493–2537, Aug. 2011.

[42] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.

[43] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the Web," in *Proc. Stanford InfoLab*, 1999, pp. 161–172.

[44] R. Mihalcea, P. Tarau, and E. Figa, "PageRank on semantic networks, with application to word sense disambiguation," in *Proc. 20th Int. Conf. Comput. Linguistics*, Geneva, Switzerland, 2004, pp. 23–27.

[45] H. Wang, J. Ye, Z. Yu, J. Wang, and C. Mao, "Unsupervised keyword extraction methods based on a word graph network," *Int. J. Ambient Comput. Intell.*, vol. 11, no. 2, pp. 68–79, Apr. 2020.

[46] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, and A. Jatowt, "YAKE! Keyword extraction from single documents using multiple local features," *Inf. Sci.*, vol. 509, pp. 257–289, Jan. 2020.

[47] S. Duari and V. Bhatnagar, "SCAKE: Semantic connectivity aware keyword extraction," *Inf. Sci.*, vol. 477, pp. 100–117, Mar. 2019.

[48] Z. Nasar, S. W. Jaffry, and M. K. Malik, "Textual keyword extraction and summarization: State-of-the-art," *Inf. Process. Manage.*, vol. 56, no. 6, Nov. 2019, Art. no. 102088.

[49] S. Herner, "Information retrieval systems: Characteristics, testing and evaluation," *J. Amer. Soc. Inf. Sci.*, vol. 33, no. 2, pp. 109–110, Mar. 1982.

[50] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommendation algorithms for e-commerce," in *Proc. 2nd ACM Conf. Electron. Commerce*, 2000, pp. 158–167.

[51] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: An overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, May 2000.

MINGXI ZHANG received the Ph.D. degree in computer software and theory from Fudan University, in 2013. He is currently an Associate Professor with the University of Shanghai for Science and Technology, Shanghai, China. His current research interests include social network analysis, and information retrieval and graph mining.

XUEMIN LI received the B.E. degree in printing engineering from Qufu Normal University, Rizhao, China, in 2018. He is currently pursuing the M.A. degree with the University of Shanghai for Science and Technology. His research interests include keyword extraction and information retrieval.

SHUIBO YUE received the B.E. degree in printing engineering from the Hunan University of Technology, Zhuzhou, China, in 2018. He is currently pursuing the M.A. degree with the University of Shanghai for Science and Technology. His research interests include social media mining and text information process.

LIUQIAN YANG received the B.E. degree in printing engineering from the University of Shanghai for Science and Technology, Shanghai, China, in 2017, where she is currently pursuing the M.A. degree. Her research interests include similarity calculation and social network analysis.