IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 25, 2019, accepted October 6, 2019, date of publication October 22, 2019, date of current version December 2, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2948891

Document Specific Supervised Keyphrase Extraction With Strong Semantic Relations

HUITING LIU[1,2], LILI WANG[1,2], PENG ZHAO[1,2], AND XINDONG WU[3,4], (Fellow, IEEE)

[1]Key Laboratory of Intelligent Computing and Signal Processing of the Ministry of Education, Anhui University, Hefei 230601, China
[2]School of Computer Science and Technology, Anhui University, Hefei 230601, China
[3]Research Institute of Big Knowledge, Hefei University of Technology, Hefei 230601, China
[4]Mininglamp Academy of Sciences, Mininglamp Technology, Beijing 100084, China

Corresponding author: Huiting Liu (htliu@ahu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB1000901, in part by the National Natural Science Foundation of China under Grant 61202227, Grant 61602004, and Grant 91746209, and in part by the Provincial Natural Science Foundation of Anhui Higher Education Institution of China under Grant KJ2018A0013.

ABSTRACT Keyphrase extraction is the task of automatically extracting descriptive phrases or concepts that represent the main topics in a document. Finding good keyphrases in a document can quickly summarize knowledge for information retrieval and decision making. Existing keyphrase extraction methods cannot be customized to each specific document, and cannot capture flexible semantic relations. In this paper, a keyphrase extraction algorithm using maximum sequential pattern mining with one-off and general gaps condition, called Ke-MSMING, is presented. Ke_MSMING first searches all keyphrase candidates from a document using sequential patterns mining and the topic model, and then adopts supervised machine learning to classify each keyphrase candidate as a keyphrase or not. Finally, Ke_MSMING selects top-N keyphrases as the final keyphrases. Ke_MSMING not only uses baseline features and pattern features but also uses centrality features obtained from the cooccurrence semantic network, and the cooccurrence networks can yield powerful semantic relations for keyphrase extraction. Experimental results on two datasets demonstrate that Ke_MSMING has better performance than other state-of-the-art keyphrase extraction approaches.

INDEX TERMS Keyphrase extraction, sequential pattern mining, general gap constraints, classification.

I. INTRODUCTION

Automatic keyphrase extraction is an important research direction in text mining, natural language processing and information retrieval. A keyphrase [1] is an ordered list of words that captures the thematic or important points discussed in a document. It is the smallest unit of understanding text, but it is important because a keyphrase gives us the flexibility and ease to accurately characterize a document. Many studies have been conducted to extract high-quality keyphrases from documents.

Keyphrase extraction usually proceeds in three steps: keyphrase candidate extraction, ranking/classification, and postprocessing. In the candidate extraction step, potentially important phrases are identified and extracted from the documents. In the ranking/classification step, these candidate terms are either ranked according to some ranking function derived from the document structure or classified as to whether they represent key terms. In the postprocessing step, top-k terms from the ranked list (or terms that are classified as keyphrases) are semantically normalized to yield a set of phrases so that each denotes a single concept.

The existing methods of ranking/classification mainly fall into two categories: (1) unsupervised learning-based methods and (2) supervised learning-based approaches. For unsupervised learning-based approaches, keyphrase selection ranks keyphrase candidates in terms of a specific measure. For supervised learning-based approaches, keyphrase selection is considered a binary classification problem, where each phrase candidate in a document is determined as a keyphrase or not. Studies [2], [3] have shown that semantic relations in context can help improve the performance of keyphrase extraction, so semantic relations are crucial in extracting a complete keyphrase candidate set from documents.

The original work on keyphrase extraction simply treats single words with high frequency as keyphrase candidates. However, frequency alone cannot capture semantic relationships in the document. Some studies, such as Kea [4], regard

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

contiguous frequently occurring n-gram words as keyphrase candidates. Although a keyphrase candidate has a few contiguous frequently occurring words, it still ignores some semantic relations in context [5]–[7]. KP-Miner [8] used sequences of words that do not contain punctuation marks or stop words as keyphrase candidates. However, to some extent, KP-Miner cannot capture semantic relations in text, which often results in unsatisfactory candidates for keyphrase extraction.

Some studies treat the keyphrase candidate search as a sequential pattern mining task, where single words of documents are viewed as characters of sequences and keyphrase candidates are viewed as patterns. For example, Xie *et al.* [9] proposed a method, sequential pattern mining with wildcards to search keyphrase candidates. Wildcards capable of conveying the semantic meaning of the text are inherently diverse and flexible. Compared to other approaches, the method of sequential pattern mining can discover a richer keyphrase candidate set, which helps to improve the quality of keyphrase extraction. However, sequential pattern mining usually needs to manually set the appropriate gap constraints. Additionally, they need to scan a document multiple times to search for patterns. Wang *et al.* [10] proposed a novel algorithm called KCSP for document-specific keyphrase candidates search using sequential pattern mining with gap constraints. KCSP scans a document once to search patterns with automatically specified gap constraints. KCSP first searches all keyphrase candidates from the document then ranks them with entropy (called PF-H), and finally selects top-N keyphrase candidates as final keyphrases. Although such a pattern-based method can capture semantic relationships in the document, it only uses pattern features of the document. Multiple discriminative features should be used to add inherent robustness and improve the performance of keyphrase extraction models.

Unsupervised approaches are domain-independent and do not require labeled training data, while supervised keyphrase extraction allows for more expressive feature design and has been reported to outperform unsupervised methods on many occasions [11]. We found two main reasons for this. First, using multiple discriminative features to rank keyphrase candidates adds inherent robustness to the models. Second, the supervision signal helps models to disregard noise. However, most supervised models cannot capture semantic relationships between words, and some studies [12]–[14] show that semantic features improve the performance of keyphrase extraction. Our general goal in this paper is to explore how the fundamental keyphrase extraction method can extract document-specific keyphrases with strong semantic relations. For this purpose, we develop a supervised learning method that uses baseline features, pattern features and word centrality features, and it combines sequential patterns discovered from each document with the topic model to extract keyphrase candidates. Our method can extract high-quality keyphrases from documents.

The key contributions of this paper are described as follows:

- In this paper, we present a novel method that first combines sequential patterns discovered from each document with the topic model and then adopts supervised machine learning to extract keyphrases. Discovered sequential patterns can capture strong semantic relationships between words, rather than the isolated meaning of individual words. Additionally, topic models can effectively obtain semantic information of documents and select important topic words or phrases as keyphrase candidates.
- Word cooccurrence networks and k-core analysis are novel word centrality features in our study. Word centrality measures on documents yield a powerful set of features for keyphrase extraction and to the best of our knowledge, they have never been used in document-specific keyphrase extraction.
- We conducted extensive experiments on SemEval-2010 and INSPEC datasets to evaluate the effectiveness and efficiency of Ke-MSMING on the proposed problem.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 introduces sequential pattern mining with wildcards. Section 4 proposes the techniques for document-specific supervised keyphrase extraction with strong semantic relations. Section 5 reports our experimental results, and we conclude the paper in Section 6.

II. RELATED WORK

The proposed research is closely related to document summarization, keyphrase extraction, and sequential pattern mining. In the following, we review related work from these three aspects.

A. DOCUMENT SUMMARIZATION

Document summarization is an important problem that has many applications in information retrieval and natural language processing. There are two types of summarization: abstractive summarization and extractive summarization. Extractive methods generate a summary for a document by directly selecting salient sentences from the original document. In contrast, abstractive methods synthesize information from the input document to generate a summary using natural language processing techniques.

Most extractive summarization approaches can be broadly classified into feature-based approaches [15], [16], graph-based approaches [17], [19] and semantic-based approaches [20], [21]. For feature-based methods, Ren *et al.* [16] presented a deep learning method to model relationships among sentences for extractive summarization. They also exploited a set of surface features such as sentence positions to enrich the knowledge of the method. For graph-based approaches, they first produced a similarity graph, in which each node represents a sentence. When the cosine similarity value between a pair of sentences exceeds a threshold, these two sentences are connected by

an edge. Other improved graph-based algorithms have also been proposed [18], [19], [22]. Semantic-based methods consider semantics behind the document content to generate accurate and readable summaries. Latent semantic analysis, an approach that maps a document to a latent semantic space, has also been shown to work well for document summarization [21].

There are some approaches for generating abstractive summaries, e.g., sentence compression [23] and sentence merging to rewrite the sentences [24]. After these studies, the author [25] presented a method Opizer-A to generate abstractive summaries of opinions. Opizer-A has two phases: clustering of textual segments and text generation based on templates.

B. KEYPHRASE EXTRACTION

Keyphrases provide semantic metadata that summarize and characterize documents. Keyphrase extraction generates keywords for academic journals, video, audio, and software, and it is particularly important in the publishing industry for carrying out some important tasks, such as the recommendation of new articles or books to customers, highlighting missing citations to authors, and identifying potential reviewers for submissions and the analysis of content trends [26]. Existing work on keyphrase extraction can be divided into supervised and unsupervised approaches.

Unsupervised learning-based approaches treat the keyphrase selection as a ranking task, and they usually assign a saliency score to each candidate phrase by considering various features and then extract keyphrases. KeyRank [10] uses sequential pattern mining to extract proper keyphrases from a document in English. The difference between KeyRank and our work is two-fold. First, KeyRank uses unsupervised approaches to extract keyphrases. Second, all the keyphrase candidates in KeyRank are frequent words in the document. Smires *et al.* [27] focused on the task of extracting keyphrases from the current document. They used sentence embeddings to extract keyphrases. You *et al.* [28] proposed an efficient core word expansion algorithm to generate candidate phrases and utilized three group features, position features, statistical features and granularity-related features, to score phrases.

In supervised learning-based approaches, keyphrase selection is formulated as a classification task. Many learning algorithms, such as support vector machines, naïve Bayes and decision tree, are used to construct a classification model. GenEx [29] and Kea [4] are two typical supervised approaches for keyphrase selection. In GenEx, a set of feature parameters are tuned by a genetic algorithm to maximize the performance on a training dataset. Kea utilizes naïve Bayes algorithm for domain-based keyphrase extraction. In this method, Kea uses two features, i.e., TF-IDF(term frequency-inverse document frequency) values and the first occurrence position of a keyphrase candidate. KeyEx [30] also uses these two features. The different between KeyEx and our work is three-fold. First, KeyEx uses baseline features and pattern features to extract keyphrases. Second, all the keyphrase candidates in KeyEx are frequent words of the document. Third, KeyEx uses sequential pattern mining with nonnegative gap constraints. Onan *et al.* [31] combined classification algorithms with ensemble learning methods to extract keyphrases. Yang *et al.* [32] proposed a weakly supervised learning approach to extract keyphrases. They considered users' specific needs and used a few prespecific seed keyphrases to guide the model.

C. SEQUENTIAL PATTERN MINING

Sequential pattern mining is an important research area in data mining. Sequential patterns have broad applications[33], such as decision making [34], air quality monitoring [35], and web recommender systems [36]. The sequential pattern mining problem was first introduced by Agrawal and Srikant [37] based on their study of customer purchase sequences. The studies of sequential pattern mining have also been extended to cover different types of sequential patterns. Flores-Garrido *et al.* [38] proposed an algorithm for mining frequent patterns in a single graph using inexact matching. Pang *et al.* [39] presented an algorithm to mine distinguishing sequential patterns from data where sequence elements can be similar to each other, and the associated patterns are called similarity-aware distinguishing sequential patterns (simDSP). Yang *et al.* [40] proposed the kDSP-miner algorithm to mine top-k distinguishing sequential patterns. Zhang *et al.* [41] proposed the CCSpan algorithm to mine closed sequential patterns.

In recent years, the problem of sequential pattern mining with wildcards where a wildcard can match any item in the alphabet has been studied. In [42], a heuristic algorithm, named dynamically changing node property (DCNP), was designed based on Nettree. The algorithm mines frequent sequential patterns with wildcards and a one-off condition from a sequence database. After this study, Wu *et al.* [43] proposed sequential pattern mining with periodic gap constraints using an incomplete Nettree structure. Liu *et al.* [44] introduced the one-off condition that requires the same position not to be shared by any two occurrences of a pattern.

In this paper, we also study the problem of mining sequential patterns with wildcards, but our focus is to apply such patterns for keyphrase extraction. Intuitively, single words in a document are the minimum meaningful and independent units, and a document is an ordered list of words. Therefore, some studies treat the keyphrase candidate search as a task of sequential pattern mining with gap constraints, where single words of documents are viewed as characters of sequences, and keyphrase candidates are viewed as patterns. Wang *et al.* [45] and Xie *et al.* [30] applied sequential pattern mining with wildcards to search keyphrase candidates (called KCSP and SPMW, respectively). KCSP utilizes sequential pattern mining with gap constraints to extract frequent sequential patterns. SPMW uses a depth-first pattern growth strategy based on a data structure, the level instance graph, to represent all instances of a pattern with a gap constraint.

TABLE 1. Comparison of sequential pattern mining algorithms.

Work(year)	Pattern Types	One-off Condition	Gap Types	Wildcard
[38] (2015)	frequent pattern	No	-	no
[40] (2017)	distinguishing pattern	No	nonnegative gaps	yes
[41] (2015)	closed patterns	No	-	no
[42] (2015)	frequent pattern	Yes	general gaps	yes
[43] (2017)	frequent pattern	No	general gaps	yes
[45] (2017)	frequent pattern	Yes	-	yes
[30] (2017)	frequent pattern	Yes	nonnegative gaps	yes
this paper	frequent pattern	Yes	general gaps	yes

We use a sequential pattern mining with gap constraints algorithm, named MSAING, to search keyphrase candidates. First, MSAING processes the pattern and sequence using a reverse strategy to obtain the maximum number of matching results. Second, it significantly reduces the time and space overhead with a linear table structure in the matching process and improves the matching rate using the backtracking method. Finally, to further improve the efficiency of the algorithm, it determines whether internal repetition exists in the pattern according to the internal checking mechanism. Please refer to our paper [44] for the specific algorithm process.

Table 1 presents the sequential pattern mining algorithm analyzed in this section, indicating for each one the year of publication, pattern types, whether it uses a one-off condition, the gap types, and whether it uses wildcards.

III. PROBLEM STATEMENT

In this paper, we formally define the problem of sequential pattern mining with general gap constraints and one-off conditions.

A sequence S is an ordered list of items, denoted by $S = \{(s_0 s_1 \cdots s_i \cdots s_{n-1} | s_i \in \Sigma, 0 \le i \le n-1)\}$, where Σ is the set of all possible items in the sequences and n is the length of a sequence.

Since a document contains a set of paragraphs, and each paragraph can be regarded as a sequence, a document can be viewed as a sequence database, defined as $SeqDB = \{S_1, S_2, \cdots S_N\}$. Additionally, single words of documents are treated as characters of sequences, and keyphrase candidates are viewed as patterns in this paper.

A wildcard is a symbol that matches any item in Σ. A gap is a list of wildcards, whose size refers to the number of wildcards. A pattern can be represented as $P = \{(p_0[\min_0, \max_0]p_1 \cdots [\min_{j-1}, \max_{j-1}]p_j \cdots [\min_{m-2}, \max_{m-2}]p_{m-1}) | p_j \in \sum\}$, where m is the length of the pattern, and $[\min_j, \max_j]$ $(\min_j < \max_j)$ specifies the minimal and maximal gap sizes between p_j and p_{j+1}.

TABLE 2. An example of sequence database SeqDB.

	Sequence
S_1	Bcabaac
S_2	Acccaacc
S_3	cccaaacc
S_4	abccabcc

Definition 1 (Pattern Occurrence and Instance): Given a pattern $P = p_0 p_1 \cdots p_{m-1}$, a sequence $S = s_0 s_1 \cdots s_i \cdots s_{n-1}$, and gap constraints $g[\min_i, \max_i]$, if there exists a position sequence $occ =< o_0, o_1, \cdots, o_j, \cdots, o_{m-1} >$ where $0 \le j \le m-1, 0 \le o_j \le n-1$ such that

(1) the o_j^{th} character of the sequence matches with the jth character of the pattern, that is $S_{o_j} = p_j$

(2) any element of the position sequence *occ* is different from each other $o_{j-1} \ne o_j$

(3) the gap between the adjacent two elements should satisfy the gap constraint; that is

$$\begin{cases} \min_{j-1} \le o_j - o_{j-1} \le \max_{j-1}, & \text{if } o_{j-1} > o_j \\ \min_{j-1} \le o_j - o_{j-1} - 1 \le \max_{j-1}, & \text{if } o_{j-1} < o_j \end{cases}$$

then *occ* is an occurrence of the pattern P in the sequence S. If $occ =< o_0, o_1, \cdots, o_j, \cdots, o_{m-1} >$ is an occurrence of P in $S_j \in SeqDB$, then $(j, < o_0, o_1, \cdots, o_j, \cdots, o_{m-1} >)$ is said to be an instance of P in SeqDB.

Definition 2 (One-Off Condition): Assume $occ = (j, < o_0, o_1, \cdots, o_k, \cdots, o_{m-1} >), occ' = (j', < o'_0, o'_1, \cdots, o'_q, \cdots, o'_{m-1} >)$ is two different instances of a pattern P in SeqDB. If the positions of any two elements in these two instances are not the same, we say that the two instances satisfy the one-off condition.

Definition 3 (Support): The support of a pattern P in a sequence database SeqDB is defined as the maximal size of all possible instance sets in which any two instances satisfy the one-off condition. The support of P is denoted by $Sup(P)$. If the support of P is not less than a given support threshold, we say that P is a frequent pattern.

Definition 4 (General Gap Constraints): If pattern P exists a gap constraint $\{\exists \min_i < 0 | 0 \le i \le m-1\}$, then the gap constraint is called a general gap constraint; otherwise, the gap constraint is a nonnegative gap constraint.

Example 1: Table 2 shows a sequence database $SeqDB = \{S_1, S_2, \cdots S_N\}$. Given a pattern P = ac and a general gap constraint g[−2,2], (1, <2,1>), (1, <4,6>) are two instances of P in S_1, while (1, <4,1>) and (1, <5,1>) are not instances of P because they do not satisfy the gap constraint. (1, <5,6>) does not satisfy the one-off condition, because position 6 has been used in (1, <4,6>). The support of P in S_1 is 2. In total, there are ten instances of P in SeqDB satisfying the general gap constraint and the one-off condition: (1, <2,1>), (1, <4,6>), (2, <0,1>), (2, <4, 3>), (2, <5,6>), (3, <3,1>), (3, <4,2>), (3, <5,6>), (4, <0,2>), (4, <4,3>). The support of P in SeqDB is 10.

In this paper, we search all keyphrase candidates from the document using the sequential pattern mining algorithm where all documents are split into paragraphs. Thus, a document can be viewed as a sequence database consisting of paragraphs. The alphabet is the set of distinct words in the document. Since the frequency of some important words may be relatively low, it is unreasonable to consider only the frequency of the words. Thus, before using sequential pattern mining, we use a topic model to filter important words and how to use the topic model is discussed in Section 4.

IV. DOCUMENT-SPECIFIC KEYPHRASE EXTRACTION

The keyphrase extraction process can be divided into three major steps: preprocessing, sequential pattern mining, and classification for keyphrase extraction. In addition, these three steps of the keyphrase extraction algorithm Sp_MSMING are explained in the following subsections, A, B and C. In subsection D, we expand the basic keyphrase extraction algorithm Sp_MSMING into an improved algorithm Ke_MSMING based on the LDA topic model. Sequential pattern mining aims to search all keyphrase candidates from the document.

A. PREPROCESSING

The main purpose of the preprocessing step is to transform each document into a sequence database for pattern mining. The preprocessing task includes removing special characters in the document, paragraph segmentation, stemming of text, removal of noisy symbols and deleting stop words. For example, we first remove variables, equations and the information about authors in the preprocessing step. Second, we segment a document into several paragraphs and regard each paragraph as a sequence according to the segmentation of the raw document. That is, we segment a document into two paragraphs if the original document has two paragraphs. Then, we delete the suffix of some words, such as the character 's' in the word 'tables', and the string 'ing' in the word 'interesting'. Finally, we remove stop words according to the list of stop words, which is often used in the preprocessing step.

We use the Porter stemmer [46] for stemming, and all characters are changed to lowercase. Stemming gives better results for keyphrase extraction method.

B. SEQUENTIAL PATTERN MINING

In this section, we devise an efficient algorithm MSMING (**M**aximum **S**equential pattern **M**ining w**I**th o**N**e-off and **G**eneral gaps condition) [44] to search all keyphrase candidates from the document. Sequential pattern mining algorithm MSMING extracts a complete keyphrase candidate set. Since wildcards provide gap constraints with great flexibility for mining patterns, the keyphrase candidates can capture semantic relations in the document. However, existing sequential pattern mining-based approaches use nonnegative gap constraints. In this paper, MSMING uses general gap constraints. Compared with nonnegative gap constraints, general gap constraints are more flexible. To enhance the diversity of the expression of a document, keyphrases tend to appear in the document with different forms, and the order of the words in keyphrases often changes. For example, the keyphrase 'data mining' may appear several times in a document with different forms, such as 'mining of the data' and 'data mining'. Sequential pattern mining with general gap constraints regards 'mining of the data' as an appearance of the pattern 'data mining', while sequential pattern mining with nonnegative gap constraints does not. Therefore, sequential pattern mining with general gap constraints can increase the quality of keyword extraction.

Furthermore, we use a new data structure linear table to embed the pattern support calculation into the pattern growth procedure. Different from the level instance graph used in [30], the linear table represents all instances of a pattern with the gap constraint, and it can not only ensure the completeness of mined patterns but also reduce time and space complexity.

Given a sequence database $SeqDB = \{S_1, S_2, \cdots, S_N\}$, pattern $P = p_0 p_1 \cdots p_{m-1}$, and the gap constraint $g[\min_i, \max_i](\min_i \leq \max_i)$, the linear table of P is defined as a 2-tuple (P, S), where S is the sequence. A length-m pattern P is divided into m rows, with the ith $(1 \leq i \leq m)$ row corresponding to the element p_{i-1}. A length-n pattern S is divided into n columns, with the jth$(1 \leq j \leq n)$ column corresponding to the element s_{j-1}. Assume W_1 and W_2 are two respective words of p_{i-1} and p_i; if W_1 appears at the position s_{j-1} of the sequence S, then (p_{i-1}, s_{j-1}) is 1. If W_2 is in the same sequence at the position s_j, then (p_i, s_j) is 1; otherwise, (p_i, s_j) is 0. If both (p_{i-1}, s_{j-1}) and (p_i, s_j) are 1, and the gap between s_{j-1} and s_j satisfies the gap constraint, then there is an edge from (p_i, s_j) to (p_{i-1}, s_{j-1}).

Table 3 shows an example of MSMING. Given a sequence $S = bcabaac$, a pattern $P = ac$ and the gap constraint $g[-2, 2]$. Table 3 has two rows, and three elements in the first row are 1; they are (p_0, s_2), (p_0, s_4) and (p_0, s_5), which means that the letter 'a' appears three times in S, and the locations are 2, 4, and 5, respectively. Two elements in the second row are 1; they are (p_1, s_1) and (p_1, s_6), which means that the letter 'c' appears at locations 1 and 6. These two elements in the second row satisfy the gap constraint $g[-2, 2]$ with two elements in the first row. Therefore, there are two edges $(p_1, s_1) \rightarrow (p_0, s_2)$ and $(p_1, s_6) \rightarrow (p_0, s_4)$, as shown in Table 3. Two occurrences (2, 1) and (4, 6) of the pattern P are found. When searching from element (p_1, s_6), the occurrence $(p_1, s_6) \rightarrow (p_0, s_5)$ is not considered because position (p_1, s_6) was used in occurrence $(p_1, s_6) \rightarrow (p_0, s_4)$. The detailed MSMING is described in Algorithms 1 and 2.

Algorithm 1 describes the process for obtaining the set of frequent sequential patterns FP, that is, the keyphrase candidates set. From step 2 to 6, each frequent pattern P with length 1 is selected from SeqDB. Thereafter, from step 7 to 13, the set of frequent sequential patterns with prefix P are discovered. The support of a sequential pattern is calculated by calling algorithm 2, as shown in steps 4 and 9.

TABLE 3. An example of MSMING.

P \ S	$s_0=b$	$s_1=c$	$s_2=a$	$s_3=b$	$s_4=a$	$s_5=a$	$s_6=c$
$P_0=a$	0	0	1	0	1	1	0
$P_1=c$	0	1	0	0	0	0	1

Algorithm 1 MSMING

Input: $SeqDB$, $g = [\min, \max]$, $\min\_sup$
Output: the set of frequent sequential patterns FP

1. $\Sigma = \{$*the* set of patterns whose length is 1 in SeqDB$\}$
2. **for** each $e \in \Sigma$ **do**
3. $P = e$;
4. sup = Cal_sup(S, P, g); //calculate the support of pattern P
5. **if** (sup >= min_sup) //whether P is a frequent pattern
6. $FP = FP \cup P$;
7. **for** each $e \in \Sigma$
8. $P' = P \circ e$;
9. sup' = Cal_sup (S, P', g);
10. **if** (sup' >= min_sup)
11. $FP = FP \cup P'$;
12. **end if**
13. **end for**
14. **end if**
15. **end for**

The main process for calculating support is illustrated in Algorithm 2. Steps 2 to 4 create a linear table. Steps 5 to 7 use general gap constraints and one-off conditions to determine whether a pattern has an occurrence in the sequence. During the course of searching the occurrence of a pattern, we use a left-first pattern growth strategy. Steps 8 to 13 count the number of occurrences of patterns, and set the value of the used position in the linear table to true according to the one-off condition.

1) TIME COMPLEXITY

The main process for sequential pattern mining (MSMING) is illustrated in Algorithm 2. Steps from 2 to 4 create a linear table, and the time complexity is $O(n + lmg)$, where n is the length of the sequences, m is the average length of the pattern, $g = \max\{\max_i - \min_i\}(0 \le i \le m-2)$, and l is the max length of the linear table. Steps from 5 to 7 count the number of patterns, whose time complexity is $O(m^2 g)$. The total time complexity of MSMING is $O(n + kmg(l + m))$, k denotes the frequency at which P_{m-1} appears in the sequences.

2) SPACE COMPLEXITY

We need $O(lm)$ to create a linear table. Therefore, the space complexity is $O(lm)$, where l is the max length of the linear table and m is the average length of the pattern.

Algorithm 2 Cal_sup(S, P, g)

Input: $P = p_0 p_1 \cdots p_{m-1}$, $g = [\min, \max]$, $S = s_0 s_1 \cdots s_{n-1}$
Output: occurrence $|P|$

1. **for** $i = 0$ to $i < n$
2. **if** ($s_i = p_{m-1}$) **then**//whether create linear table
3. create linear table; // create linear table
4. flag_table = true;
5. **if** (flag_table)
6. occ = left-first pattern growth strategy of the table; // use general gap constraints and one-off conditions
7. **end if**
8. **if** (occ)
9. $occurrence = occurrence \cup occ$; //count the number of patterns
10. **for** $j = 0$ to $m-1$
11. $used[occ_j] = true$; //use one-off condition to set value of the used position true
12. **end for**
13. **end if**
14. **end if**
15. **end for**
16. return occurrence $|P|$ //output the support of pattern P

C. CLASSIFICATION FOR KEYPHRASE EXTRACTION

1) FEATURES EXTRACTION

In this subsection, we use the discovered sequential patterns to formulate a supervised classification task for keyphrase extraction. The keyphrase extraction problem is considered a binary classification problem, where each candidate keyphrase in a document is determined as a keyphrase or not.

To train document-specific keyphrase extraction models, Xie *et al.* [30] proposed integrating two types of features into the learning process: (1) baseline features and (2) pattern features. The former features are commonly used in existing methods, and the latter are collected from the mined patterns. However, baseline features and pattern features only show the statistical information of the words or the patterns, but they cannot describe the centrality features of the words. Word centrality features are features defined on word cooccurrence semantic networks [47], and they can capture the semantic connections between words.

For each document, we design a cooccurrence semantic network. The network is constructed by adding all the unique words as nodes and drawing edges between the adjacent words in position. Centrality measures on such cooccurrence

Abstract: A challenging problem faced by researchers and developers of distributed real-time and embedded (DRE) systems is devising and implementing effective adaptive resource management strategies that can meet end-to-end quality of service(QoS) requirements in varying operational conditions. This paper presents two contributions to research in adaptive resource management for DRE systems. First, we describe the structure and functionality of the Hybrid Adaptive Resource management Middleware (HyARM), which provides adaptive resource management using hybrid control techniques for adapting to workload fluctuations and resource availability. Second, we evaluate the adaptive behavior of HyARM via experiments on a DRE multimedia system that distributes video in real-time. Our results indicate that HyARM yields predictable, stable, and high system performance, even in the face of fluctuating workload and resource availability.

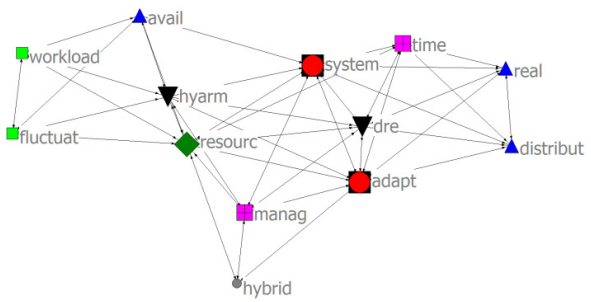
(a) The original document

abstract challeng face research develop distribut real time embed dre system devis implement effect adapt resourc manag strategi meet qualiti servic qo requir vari oper condit contribut adapt resourc manag dre system first structur function hybrid adapt resource manag middlewar hyarm adapt resourc manag hybrid control techniqu adapt workload fluctuat resourc avail second evalu adapt behavior hyarm experi dre multimedia system distribut video real time our hyarm yield predict stabl system perform fluctuat workload resourc avail

(b) The preprocessed document

	adapt	distribut	dre	hyarm	hybrid	manag	real	resourc	system	time	avail	fluctuat	workload
adapt		2	3	2	2	3	2	4	3	2			
distribut	2		2				2		2	2			
dre	3	2		2		2	2	3	3	2			
hyarm	2		2			2		3	3		2	2	2
hybrid	2					2		2					
manag	3		2	2	2			3	2				
real	2	2	2						2	2			
resourc	4		3	3	2	3			4	2	2	2	2
system	3	2	3	3		2	2	4		2	2		
time	2	2	2				2	2	2				
avail				2				2	2			2	2
fluctuat				2				2			2		2
workload				2				2			2	2	

(c) A coword matrix



(d) The cooccurrence semantic network

FIGURE 1. Process for designing the cooccurrence semantic network.

semantic networks can yield a powerful set of features for keyword extraction. Fig. 1 shows the process of designing the cooccurrence semantic network.

The original document is given in subgraph (a), and the document in subgraph (b) has been preprocessed by removing stop words and stemming. From the document in subgraph (b), by counting the cooccurrence of words, we can construct a coword matrix, as shown in subgraph (c). Based on the matrix in (c), we obtain the cooccurrence semantic network shown in (d).

Word cooccurrence networks and k-core analysis are novel word centrality features in our study. For each document, a word cooccurrence network is constructed by adding all the word types (i.e., unique words) as nodes, and drawing an edge between the words that occur in the same sequence satisfying given gap constraints. Word centrality measures on documents can yield a powerful set of features for keyword extraction, and to the best of our knowledge, they have never been used in document-specific keyword extraction. In this paper, the word centrality features we focus on include degree and coreness. We integrate three types of features into the learning process: (1) baseline features, (2) pattern features, and (3) centrality features, as shown in Table 4.

TABLE 4. Features used to train the keyphrase extraction model.

Category	Feature	Description
baseline features	TFIDF	TF*IDF value
	firstPos	first occurrence position
pattern features	Sup	pattern support
	Len	pattern length
	sup*len	product of sup and len
	closedNum	number of closed patterns
centrality features	Degree	number of edges of a node
	Coreness	importance of a word

2) KEYPHRASE EXTRACTION MODEL

We experimented with a number of different machine learning schemes, and we chose the naïve Bayes technique [48] to train the keyphrase extraction model because it is simple and yields good results in Kea [4]. During the classification phase, the model determines the overall probability that each candidate is a keyphrase, and then selects the top-k candidates as extracted keyphrases. The detailed keyphrase extraction algorithm based on sequential patterns, named Sp_MSMING is described in Algorithm 3.

The main process of extracting keyphrases is illustrated in Algorithm 3. The Sp_MSMING mainly fall into two stages: training phase (steps 1 to 15) and classification phase (steps 16 to 30). Step 3 transforms the document d' into SeqDB. Step 4 calls MSMING (Algorithm 1) to mine all frequent patterns. Steps 6 to 9 extract the baseline and pattern features of patterns. Steps 10 to 11 extract the centrality features. Step 12 combines baseline and pattern features with centrality features. Step 15 trains the keyphrase extraction model by using naïve Bayes on the training set. The classification phase is similar to the training phase. However, in step 27, the classifier calculates the probability of keyphrase candidates. Step 30 ranks keyphrase candidates according to their probability value and selects top-k candidates with the highest probability as keyphrases.

D. KEYPHRASE EXTRACTION BASED ON TOPIC MODEL

In the step of classification for keyphrase extraction, baseline features, pattern features, and centrality features are the statistical or centrality information of frequency words. However, these words may not capture the topics of a document. In some documents, the most important words do not appear frequently. If we only consider the statistical or centrality information of frequency words during the course of keyphrase extraction, some important words will be missed, and the extracted keyphrases will not embody the topics of the document [49]. To improve the correctness of keyphrase extraction, before using sequential pattern mining, we apply latent Dirichlet allocation (LDA) [50], [51] to obtain the word distribution over topics. The generation step is performed in two smaller steps: training and inference.

Algorithm 3 Sp_MSMING

Input: the training set D with keyphrases labeled; the document d for extracting keyphrases; $g = [\min, \max]$, min _ sup

Output: the keyphrases extracted from document d

1. **// Training phase:**
2. **for** each document $d' \in D$ **do**
3. transform d' into a sequence database; // transform d' into SeqDB
4. $FP' \rightarrow$ MSMING; // use general gap constraints and the one-off condition to mine frequent patterns
5. **for** each keyphrase candidate $kc' \in FP'$ in d' **do**
6. $BF'_{kc} \rightarrow$ extract baseline features from FP' for each keyphrase candidate kc' in d';
7. $BF'[] = BF'[] \cup BF'_{kc}$; // extract baseline features
8. $PF'_{kc} \rightarrow$ extract pattern features from FP' for each keyphrase candidate kc' in d';
9. $PF'[] = PF'[] \cup PF'_{kc}$; // extract pattern features
10. $CF'_{kc} \rightarrow$ extract centrality features from FP' for each keyphrase candidate kc' in d';
11. $CF'[] = CF'[] \cup CF'_{kc}$; // extract centrality features
12. $F'[] = BF'[] \cup PF'[] \cup CF'[]$; //combine features
13. **end for**
14. **end for**
15. Sp_MSMING $\leftarrow$ train keyphrase extraction model from D by using naïve Bayes on training set $F'[]$; //train keyphrase extraction model
16. **// Classification phase:**
17. transform d into a sequence database; // transform d into SeqDB
18. $FP \rightarrow$ MSMING; // use general gap constraints and one-off condition to mine frequent patterns
19. **for** each keyphrase candidate $kc \in FP$ in d **do**
20. $BF_{kc} \rightarrow$ extract baseline features from FP for each keyphrase candidate kc in d;
21. $BF[] = BF[] \cup BF_{kc}$; // extract baseline features
22. $PF_{kc} \rightarrow$ extract pattern features from FP for each keyphrase candidate kc in d;
23. $PF[] = PF[] \cup PF_{kc}$; // extract pattern features
24. $CF_{kc} \rightarrow$ extract centrality features from FP for each keyphrase candidate kc in d;
25. $CF[] = CF[] \cup CF_{kc}$; // extract centrality features
26. $F[] = BF[] \cup PF[] \cup CF[]$; //combine features
27. $p(kc) \leftarrow$ calculate probability of kc being a keyphrase by exploiting the trained naïve Bayes classifier Sp_MSMING;
28. $P[] = P[] \cup p(kc)$;
29. **end for**
30. keyphrases $\leftarrow$ select top-k keyphrase candidates with the highest probability value from $P[]$ as keyphrases; //extract keyphrases

Training. Given a corpus D consisting of M documents, LDA assumes that each word W is connected with a latent topic Z. Each topic $Z_k, k \in \{1, 2, \cdots, K\}$ is related to a multinomial distribution θ_m, and each θ_m is chosen from a Dirichlet prior distribution with parameter α. Sample a topic Z_k according to distribution θ_m, and then sample a word according to topic-word distribution ϕ_k, which is drawn from a Dirichlet prior distribution with parameter β. After training, we obtain a word-topic weight matrix and a document-topic distribution matrix used for the inference.

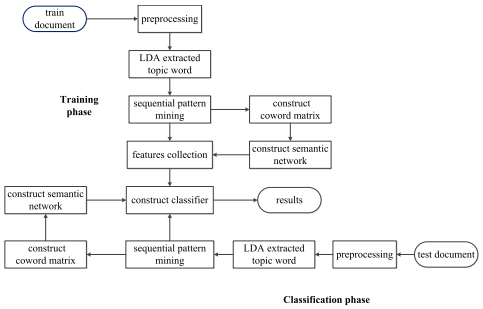


FIGURE 2. The model of Ke_MSMING.

Inference. Given a document d, we first choose the top-k closest topics of d. The closest topics of d are those that have the highest values in the document-topic distribution matrix. For each topic, we select the top-n words, which have the highest weights in the word-topic weight matrix. As a result, the number of topical words for each document is |k|*|n|. The topical model allows our model to select salient words or phrases that contain important topics in the document.

Based on LDA and sequential pattern mining, we propose a keyphrase extraction method Ke_MSMING, which can capture strong semantic relations between words and obtain meaningful topics corresponding to contexts for each extraction. The algorithm model is shown in Fig. 2.

In general, Ke_MSMING can be divided into two stages: training and classification. The training phase mainly constructs the classifier. First, the document is preprocessed; that is, the words are processed by the stop words list and the Porter stemmer [46], and each document is converted into a sequence database SeqDB. Second, before using sequential pattern mining, we use the topic model to filter important candidate words. The filtered candidate words capture the topical meaning of the text. Third, the general gap sequence pattern mining algorithm MSMING is used to calculate the frequency of the words, and other features of the frequent patterns are obtained. Then, for each document, we design a cooccurrence semantic network, construct a cooccurrence network by adding all the word types (i.e., unique words) as nodes, and draw an edge between the words that occur in the same sequence to satisfy the given gap constraints. Finally, the naïve Bayes classifier is used to train and construct the classifier.

The classification phase mainly verifies the performance of the classifier constructed in the training phase. In the classification phase, the effectiveness of the algorithm is calculated by comparing the extracted keyphrases with the real keyphrases.

V. EXPERIMENTAL RESULTS

A. EXPERIMENTAL SETUP

The sequential pattern mining algorithm is implemented in Microsoft Visual Studio 2013 with the language of C++. Other algorithms are implemented in Java. All algorithms are implemented on a PC with an Intel(R) Core(TM)i3-4170 CPU @ 3.70 GHz, 8.0 GB of memory, running the Windows 7 OS.

B. EXPERIMENTAL DATASETS AND EVALUATION METRICS

To comprehensively experimentally evaluate the performance of our methods on document-specific keyphrase extraction, we conduct experiments on two datasets. One dataset is INSPEC [52], which contains 2,000 abstracts (1,000 for training, 500 for development, and 500 for testing). For each abstract, there are two sets of keyphrases: controlled manually assigned keywords and uncontrolled manually assigned keywords. The other dataset is the keyphrase extraction benchmark dataset SemEval-2010, which was published by the ACL SemEval workshop [53]. The dataset includes 244 articles (144 for training and 100 for testing). For each article, there are three sets of keyphrases provided: author-assigned keyphrases, reader-assigned keyphrases, and combined (both author- and reader- assigned) keyphrases.

To evaluate the predictive performance of keyphrase extraction methods, precision (P), recall (R), and the F1-measure are utilized as the evaluation metrics. They are defined as follows:

$$P = \frac{\#correct}{\#extracted} \tag{1}$$

$$R = \frac{\#correct}{\#labeled} \tag{2}$$

$$F_1 = \frac{2 * P * \mathrm{R}}{P + R} \tag{3}$$

where #correct denotes the number of correctly extracted keyphrases, #extracted is the number of extracted keyphrases, and #labeled denotes the number of labeled keyphrases. The values of P and R are between 0 and 1. The closer the value is to 1, the higher the precision or recall rate will be. The F1 value is the harmonic mean of P and R.

C. BASELINE

In the experiment, we compare the performance of Ke_MSMING with different keyphrase extraction methods (supervised based approach KeyEx [30], unsupervised based approaches KeyRank [45] and TextRank [47]). These methods are also the compared methods in [10].

D. FEATURE ANALYSIS

For comparison, we divide the baseline features, pattern features and centrality features into five different groups:

I: sup*len+ pos
II: sup*len+ degree
III: sup*len+ coreness
IV: closedNum + pos
V: TFIDF + pos

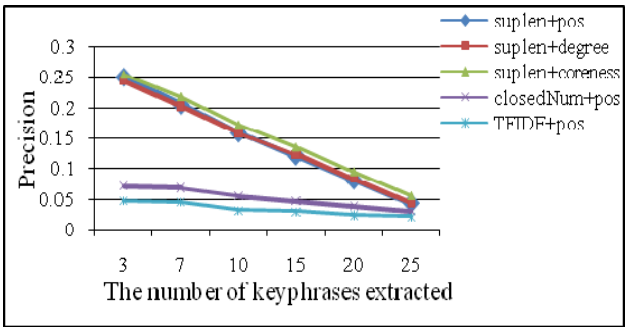


FIGURE 3. The precision of five groups of features with respect to different numbers of keyphrases.

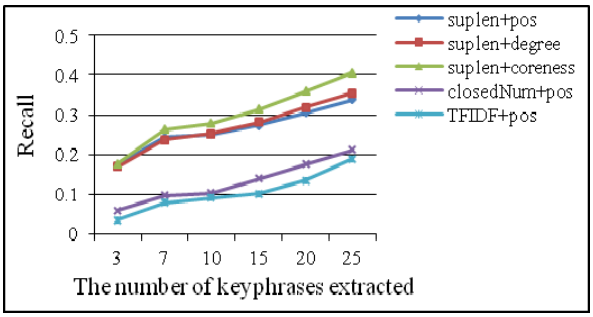


FIGURE 4. The recall of five groups of features with respect to different numbers of keyphrases.

In order to prove that the composite features are superior to the baseline features, we divide features into five groups. The pattern features occur in the first four groups. The centrality features are in the second and third groups. The baseline features are in the first, fourth and fifth groups. The features in the last group are only baseline features, and the first four groups are the composite features. The pattern features combine with baseline features or centrality features to form composite features. The following experimental results show that the composite features are superior to the baseline features. We first compare the five different features groups on the dataset SemEval-2010 in terms of P, R, and F1 scores, and the results are shown in Fig. 3, Fig. 4 and Fig. 5, respectively. Then, we further compare the baseline features, the baseline and pattern features, and the baseline, pattern and centrality features on SemEval-2010 dataset in terms of P, R, and F1 scores, and the results are shown in Fig. 6, Fig. 7 and Fig. 8, respectively. They report the performances of different groups of features with respect to different numbers of extracted keyphrases.

The results in Figs. 3-5 suggest several key observations. The first four groups perform significantly better than the last group. It appears that pattern features and centrality features are both more important than baseline features. Second, when the number of extracted keyphrases increases, the precision of the five curves decreases (see Fig. 3). In addition, we notice that the recall of the five curves increases when the number of extracted keyphrases varies from 3 to 25 (see Fig. 3) because, with the increment of the number of extracted keyphrases,

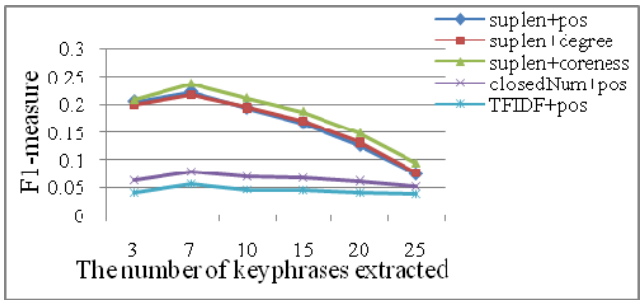


FIGURE 5. The F1-measure of five groups of features with respect to different numbers of keyphrases.

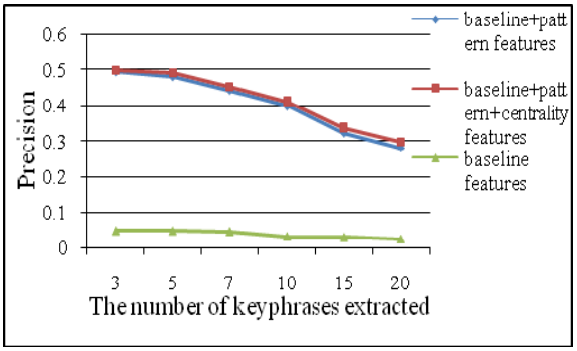


FIGURE 6. The precision of five groups of features with respect to different numbers of keyphrases.

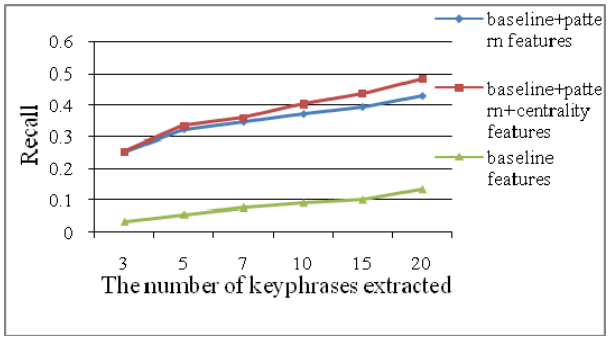


FIGURE 7. The recall of five groups of features with respect to different numbers of keyphrases.

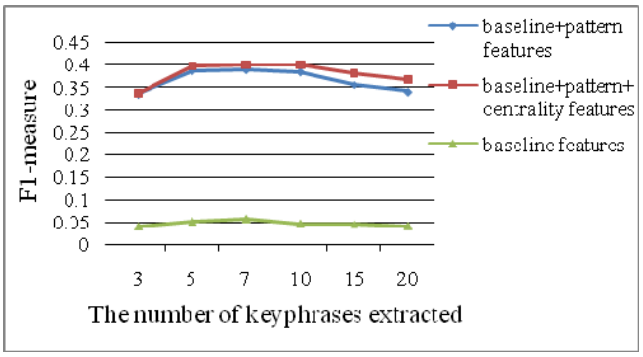


FIGURE 8. The F1-measure of five groups of features with respect to different numbers of keyphrases.

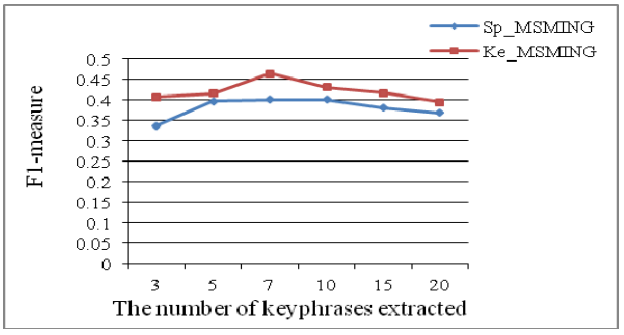


FIGURE 9. The F1-measure of whether using LDA or not with respect to different numbers of keyphrases.

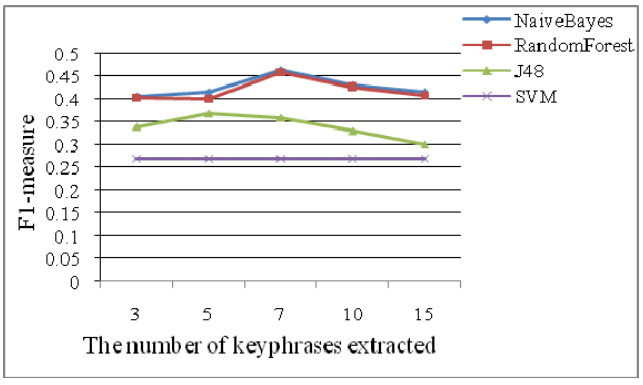


FIGURE 10. The F1-measure of different classifiers with respect to different numbers of keyphrases on the dataset SemEval-2010.

more correct keyphrases are extracted, which increases the recall value. However, the growth rate of the number of correctly extracted keyphrases is less than that of the number of extracted keyphrases. As a result, the precision decreases. The F1-measure is a combined measure integrating both precision and recall values. F1-measures of the five curves increase when the number of extracted keyphrases is small (see Fig. 5), and then slightly decreases with the increase in the number of extracted keyphrases.

Baseline features and pattern features are commonly used in existing methods. However, they only show the statistical information of the words or patterns. Centrality features can accurately capture their semantics in the context of the document, and therefore can improve keyphrase extraction quality. Fig. 6, 7 and 8 report the performances of each group of features in the SemEval-2010 dataset. We notice that the curve of the baseline and pattern features is much better than the baseline curve. The baseline, pattern and centrality features curve always performs better than the baseline and pattern features curve, which is reasonable because compared with the pattern and centrality features, baseline features do not contain enough characteristics to distinguish keyphrases from nonkeyphrases.

E. TOPIC MODEL ANALYSIS

Fig. 9 reports the F1-measure using (or not) the LDA topic model on the SemEval-2010 dataset. We notice that the performance of Ke_MSMING (using the LDA topic model)

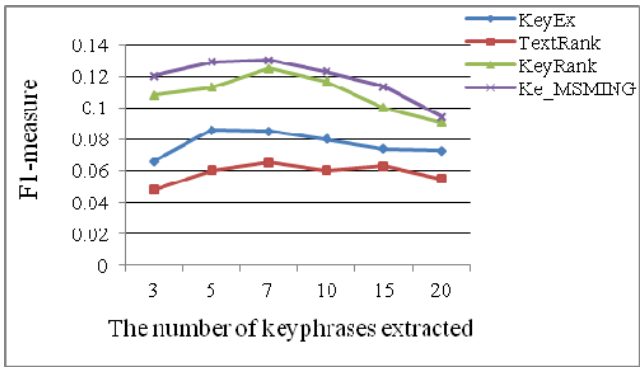


FIGURE 11. The F1-measure of different methods with respect to different numbers of keyphrases on the dataset SemEval-2010.

is much better than that of the Sp_MSMING (not using the LDA topic model). Before sequential pattern mining, Ke-MSMING uses the LDA topic model to train the documents and selects salient words or phrases that contain important topics in the document. While Sp_MSMING extracts keyphrase candidates directly with sequential pattern mining, the keyphrases extracted by Ke-MSMING embody more topical information of the documents and the F1-measure value is higher than Sp-MSMING.

F. CLASSIFICATION ALGORITHM COMPARISON

Fig. 10 reports the performance of Ke_MSMING using different classification algorithms, including naïve Bayes, J48, random forest, and SVM. We use the open source machine learning software Weka [54]. It can be seen from Fig. 10 that the highest (best) classification F1-measure value is obtained by the naïve Bayes classifier, and this is in accordance with the result in reference [4].

G. COMPARISON WITH OTHER KEYPHRASE EXTRACTION METHODS

The experimental results on the dataset SemEval-2010 and INSPEC are shown in Fig. 11 and Fig. 12. From Fig. 11 and Fig. 12, we can see that Ke-MSMING performs significantly better than TextRank, KeyRank and KeyEx. The main reason is Ke-MSMING exploits LDA and sequential pattern mining to obtain keyphrase candidates and uses several features, such as baseline features, pattern features and centrality features to train the keyphrases extraction model. The keyphrases extracted with Ke-MSMING capture more semantic meaning of the documents.

In Fig. 11, when the number of extracted keyphrases is 7, Ke-MSMING performs significantly well. However, in Fig. 12, it achieves the optimal value when the number of extracted keyphrases is 5, mainly because the dataset INSPEC only contains titles and abstracts, whereas the dataset SemEval-2010 contains whole documents.

To compare the experimental results of the proposed algorithm with those of its comparison algorithms in two datasets, student's t-test with a saliency level of 5% is adopted. When Ke-MSMING is compared with other algorithms, the

TABLE 5. P-value of Student's T-test results (Bold indicates that P-value is greater than 0.05).

		SemEval-2010	INSPEC
		F1-measure	
Ke-MSMING	KeyRank	**0.143348**	0.002158
	TextRank	0.000641	0.000338
	KeyEx	0.000107	0.000049

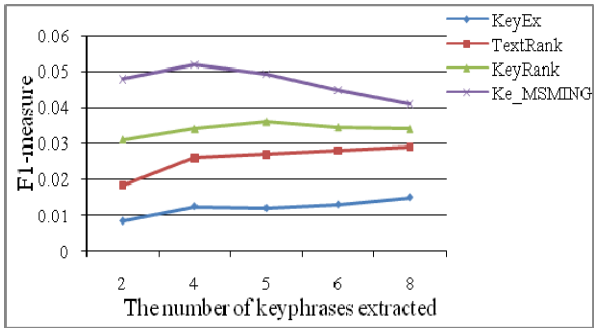


FIGURE 12. The F1-measure of different methods with respect to different numbers of keyphrases on the dataset INSPEC.

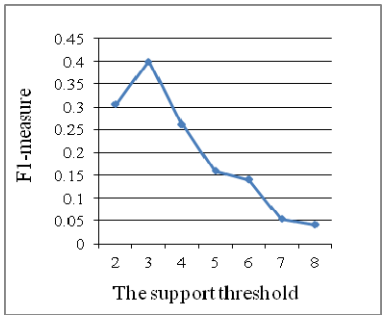


FIGURE 13. The F1-measure with respect to different numbers of support threshold on the dataset SemEval-2010.

performance of Ke-MSMING is inferior to or equal to that of its comparison algorithms and is taken as a zero hypothesis. In addition, the performance of Ke-MSMING is better than that of the contrast algorithms and is taken as an alternative hypothesis. From Table 5, it can be seen that only one P-value of Ke-MSMING and each comparison algorithm on two datasets is greater than 0.05 (i.e., supporting the original hypothesis). Comprehensive analysis shows that the performance of Ke-MSMING is better than other algorithms.

H. PARAMETER ANALYSIS

In this experiment, we examine the influence of several parameters, including the support threshold, the maximal gap size, the topic number and the number of extracted keyphrases on keyphrase extraction performance. For this purpose, we test our method varying the value of one parameter, but fix all the other parameters values.

The resulting F1-measure of the support threshold and that of the maximal gap size are summarized in Fig. 13 and Fig. 14, respectively. We first let the maximal gap size be 9 and then vary the support threshold from 2 to 8, and report

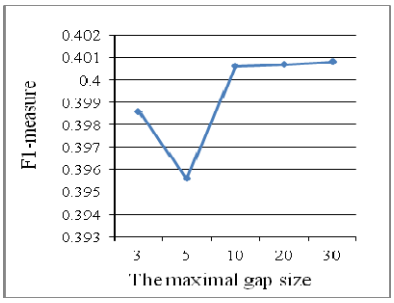


FIGURE 14. The F1-measure with respect to different numbers of maximal gap size on the dataset SemEval-2010.

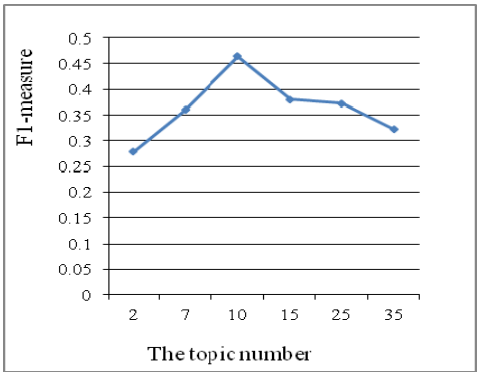


FIGURE 15. The F1-measure with respect to different numbers of topics on SemEval-2010.

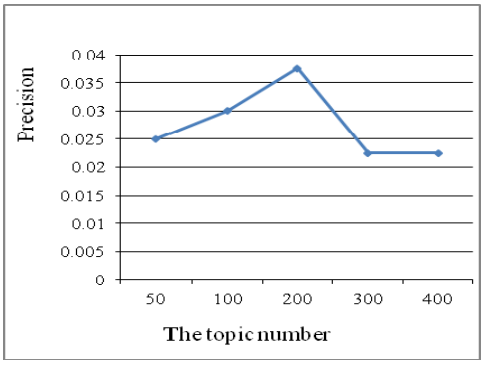


FIGURE 16. The precision with respect to different numbers of topics on INSPEC.

the F1-score value in Fig. 13. According to Fig. 13, with the support threshold increasing, the performance gradually improves. However, when the support threshold is above 3, the F1-measure score reaches the peak, and then it decreases with the increase in the value of the support threshold. Another important parameter is the maximal gap size. We set the support threshold to 3, and then vary the maximal gap size from 3 to 30, and report the F1-score values in Fig. 14. Intuitively, the curve reaches the peak when the maximal gap size is set to 10. When the gap size is above 10, the F1-measure score remains unchanged.

The resulting F1-measure on SemEval-2010 with respect to the different number of topics is shown in Fig. 15. We first let the maximal gap size be 9, the support threshold be 3 and

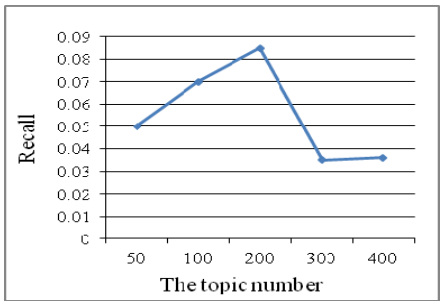


FIGURE 17. The recall with respect to different numbers of topics on INSPEC.

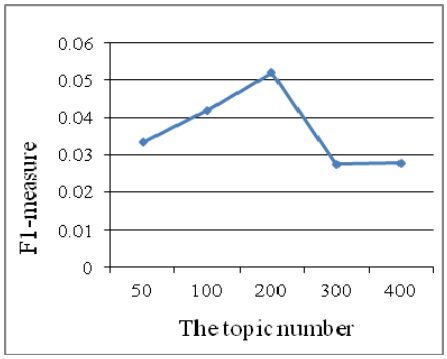


FIGURE 18. The F1-measure with respect different numbers of topics on INSPEC.

the number of extracted keyphrase words be 7. According to Fig. 15, with the topic number increasing, the performance gradually improves. However, when the topic number is above 10, the F1-measure score reaches the peak and then, it decreases with the increase in the topic number.

Fig. 16, 17 and 18 report the performance of different numbers of topics on the INSPEC dataset. The precision related to the topic number is illustrated in Fig. 16. We first let the maximal gap size be 9, the support threshold be 2 and the number of extracted keyphrases be 4. According to Fig. 16, the performance gradually improves when the topic number grows. However, when the topic number is above 200, the precision reaches the peak, and then, it decreases with the increase in the topic number. In Fig. 17, when the topic number increases, the recall gradually improves. However, when the topic number is above 200, the recall reaches the peak, and then, it decreases with the increase in the topic number. In Fig. 18, with the topic number increasing, the F1-measure score gradually improves. However, when the topic number is above 200, the F1-measure score reaches the peak, and then, it decreases with the increase in the topic number.

Fig. 19 reports the F1-measure with respect to different numbers of extracted keyphrases on INSPEC. We first let the maximal gap size be 9, the support threshold be 2, and the topic number be 200. According to Fig. 19, with the number of extracted keyphrases increasing, the performance gradually improves. However, when the number of keyphrases extracted is above 3, the F1-measure score reaches the peak,

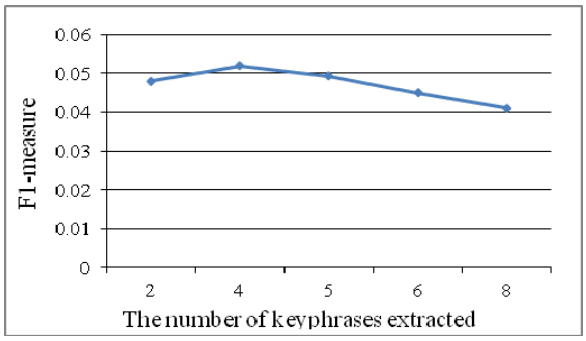


FIGURE 19. The F1-measure with respect to different numbers of extracted keyphrases on INSPEC.

and then, it decreases with the increase in the number of extracted keyphrases.

VI. CONCLUSION

In this paper, we propose a new approach Ke-MSMING for keyphrase extraction. Ke-MSMING is a document-specific keyphrase extraction algorithm using topic model and sequential pattern mining with general gap constraints. Two main problems have been solved. The first is how to improve the quality of extracted keyphases. The second is how to capture semantic relationships between words. To solve these two problems, we use LDA and sequential pattern mining with different general gap constraints to discover a rich set of keyphrase candidates for each individual document, and further use a supervised learning approach to build a classification model for keyphrase extraction. To improve the classification efficiency, we integrate three types of features into the learning process: (1) baseline features, (2) pattern features, and (3) centrality features. To consider the topics of documents in the step of searching keyphrase candidates, we use topic model LDA before using sequential pattern mining. Experiments demonstrate that the proposed keyphrase extraction method is effective in improving the quality of extracted keyphrases. Currently, deep neural networks are being used for keyphrase extraction, as they can obtain an effective representation of text. In the future, we will extend our work to apply deep neural networks to obtain word embeddings of keyphrase candidates to improve the accuracy of keyphrase extraction.

ACKNOWLEDGMENTS

The authors would like to thank Mr. Qingren Wang for assisting us with the experiment setting up.

REFERENCES

[1] Y. Song, S. Liu, X. Liu, and H. Wang, "Automatic taxonomy construction from keywords via scalable Bayesian rose trees," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 7, pp. 1861–1874, Jul. 2015.

[2] G. Ercan and I. Cicekli, "Using lexical chains for keyword extraction," *Inf. Process. Manage.*, vol. 43, no. 6, pp. 1705–1714, Nov. 2007.

[3] S. Xu, S. Yang, and L. F. Keyword, "Keyword extraction and headline generation using novel word features," in *Proc. 24th AAAI Conf. Artif. Intell.*, Atlanta, GA, USA, Jul. 2010, pp. 1461–1466.

[4] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning, "Domain-specific keyphrase extraction," in *Proc. 16th Int. Joint Conf. Artif. Intell.*, Stockholm, Sweden, Jul./Aug. 1999, pp. 668–673.

[5] Z. Fu, F. Huang, K. Ren, J. Weng, and C. Wang, "Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 8, pp. 1874–1884, Aug. 2017.

[6] J. Li, X. Li, B. Yang, and X. Sun, "Segmentation-based image copy-move forgery detection scheme," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 507–518, Mar. 2015.

[7] 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.

[8] 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.

[9] F. Xie, X. Wu, and X. Zhu, "Document-specific keyphrase extraction using sequential patterns with wildcards," in *Proc. IEEE Int. Conf. Data Mining*, Shenzhen, China, Dec. 2014, pp. 1055–1060.

[10] Q. Wang, V. S. Sheng, and X. Wu, "Document-specific keyphrase candidate search and ranking," *Expert Syst. Appl.*, vol. 97, pp. 163–176, May 2018.

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

[12] P. D. Turney, "Coherent keyphrase extraction via Web mining," in *Proc. 18th Int. Joint Conf. Artif. Intell.*, Aug. 2003, pp. 434–439.

[13] O. Medelyan and I. H. Witten, "Thesaurus based automatic keyphrase indexing," in *Proc. 6th ACM/IEEE-CS Joint Conf. Digit. Libraries (JCDL)*, Chapel Hill, NC, USA, Jun. 2006, pp. 296–297.

[14] S. Xu, S. Yang, and F. Lau, "Keyword extraction and headline generation using novel word features," in *Proc. 24th AAAI Conf. Artif. Intell.*, Atlanta, GA, USA, Jul. 2010, pp. 1461–1466.

[15] R. Tadano, K. Shimada, and T. Endo, "Multi-aspects review summarization based on identification of important opinions and their similarity," in *Proc. 24th Pacific Asia Conf. Lang., Inf. Comput.* Sendai, Japan: Tohoku Univ., Nov. 2010, pp. 685–692.

[16] P. Ren, Z. Chen, Z. Ren, F. Wei, J. Ma, and M. de Rijke, "Leveraging contextual sentence relations for extractive summarization using a neural attention model," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Shinjuku, Tokyo, Japan, Aug. 2017, pp. 95–104.

[17] X. Wan, "An exploration of document impact on graph-based multi-document summarization," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Honolulu, HI, USA, Oct. 2008, pp. 755–762.

[18] Z. Yang, K. Cai, J. Tang, L. Zhang, Z. Su, and J. Li, "Social context summarization," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Beijing, China, Jul. 2011, pp. 255–264.

[19] J. Zhu, C. Wang, X. He, J. Bu, C. Chen, S. Shang, M. Qu, and G. Lu, "Tag-oriented document summarization," in *Proc. 18th Int. Conf. World Wide Web*, Madrid, Spain, Apr. 2009, pp. 1195–1196.

[20] M. T. Nguyen, D. V. Tran, L. M. Nguyen, and X. H. Phan, "Exploiting user posts for Web document summarization," *ACM Trans. Knowl. Discovery Data*, vol. 12, no. 4, p. 49, Jul. 2018.

[21] Y. Gong and X. Liu, "Generic text summarization using relevance measure and latent semantic analysis," in *Proc. 24th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, New Orleans, LA, USA, Sep. 2001, pp. 19–25.

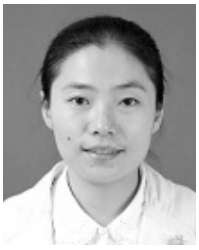
[22] D. Bollegala, N. Okazaki, and M. Ishizuka, "A bottom-up approach to sentence ordering for multi-document summarization," *Inf. Process. Manage.*, vol. 46, no. 1, pp. 89–109, Jan. 2010.

[23] D. M. Zajic, B. J. Dorr, J. Lin, and R. Schwartz, "Multi-candidate reduction: Sentence compression as a tool for document summarization tasks," *Inf. Process. Manage.*, vol. 43, no. 6, pp. 1549–1570, Nov. 2007.

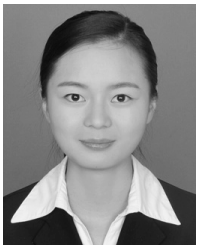
[24] Y. Chali, M. Tanvee, and M. T. Nayeem, "Towards abstractive multi-document summarization using submodular function-based framework, sentence compression and merging," in *Proc. 8th Int. Joint Conf. Natural Lang. Process. (IJCNLP)*, Taipei, Taiwan, Nov./Dec. 2017, pp. 418–424.

[25] R. E. L. Condori and T. A. S. Pardo, "Opinion summarization methods: Comparing and extending extractive and abstractive approaches," *Expert Syst. Appl.*, vol. 78, pp. 124–134, Jul. 2017.

[26] I. Augenstein, M. Das, S. Riedel, L. Vikraman, and A. McCallum, "Scienceie-extracting keyphrases and relations from scientific publications," in *Proc. 11th Int. Workshop Semantic Eval.*, Aug. 2017, pp. 546–555.
[27] K. B. Smires, C. Musat, A. Hossmann, M. Baeriswyl, and M. Jaggi, "Simple unsupervised keyphrase extraction using sentence embeddings," in *Proc. 22nd Conf. Comput. Natural Lang. Learn.*, Brussels, Belgium, Oct./Nov. 2018, pp. 221–229.
[28] W. You, D. Fontaine, and J. P. A. Barthés, "An automatic keyphrase extraction system for scientific documents," *Knowl. Inf. Syst.*, vol. 34, no. 3, pp. 691–724, Mar. 2013.
[29] P. D. Turney, "Learning to extract keyphrases from text," 2002, *arXiv:cs/0212013*. [Online]. Available: https://arxiv.org/abs/cs/0212013
[30] F. Xie, X. Wu, and X. Zhu, "Efficient sequential pattern mining with wildcards for keyphrase extraction," *Knowl.-Based Syst.*, vol. 115, pp. 27–39, Jan. 2017.
[31] A. Onan, S. Korukoğlu, and H. Bulut, "A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification," *Expert Syst. Appl.*, vol. 62, pp. 1–16, Nov. 2016.
[32] M. Yang, Y. Liang, W. Zhao, W. Xu, J. Zhu, and Q. Qu, "Task-oriented keyphrase extraction from social media," *Multimedia Tools Appl.*, vol. 77, no. 3, pp. 3171–3187, Feb. 2018.
[33] T. Van, B. Vo, and B. Le, "Mining sequential patterns with itemset constraints," *Knowl. Inf. Syst.*, vol. 57, no. 2, pp. 311–330, Nov. 2018.
[34] H. C. Neto and R. M. S. Julia, "ACE-RL-Checkers: Decision-making adaptability through integration of automatic case elicitation, reinforcement learning, and sequential pattern mining," *Knowl. Inf. Syst.*, vol. 57, no. 3, pp. 603–634, Dec. 2018.
[35] C. Zhang, Y. Zheng, X. Ma, and J. Han, "Assembler: Efficient discovery of spatial co-evolving patterns in massive geo-sensory data," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Sydney, NSW, Australia, Aug. 2015, pp. 1415–1424.
[36] R. Mishra, P. Kumar, and B. Bhasker, "A Web recommendation system considering sequential information," *Decis. Support Syst.*, vol. 75, pp. 1–10, Jul. 2015.
[37] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proc. 11th Int. Conf. Data Eng.*, Taipei, Taiwan, Mar. 1995, pp. 3–14.
[38] M. F. Garrido, J. A. C. Ochoa, and J. F. M. Trinidad, "AGrap: An algorithm for mining frequent patterns in a single graph using inexact matching," *Knowl. Inf. Syst.*, vol. 44, no. 2, pp. 385–406, Aug. 2015.
[39] T. Pang, L. Duan, J. L. Ling, and G. Dong, "Mining similarity-aware distinguishing sequential patterns from biomedical sequences," in *Proc. IEEE 2nd Int. Conf. Data Sci. Cyberspace*, Shenzhen, China, Jun. 2017, pp. 43–52.
[40] H. Yang, L. Duan, B. Hu, S. Deng, W. T. Wang, and P. Qin, "Mining top-k distinguishing sequential patterns with gap constraint," *J. Softw*, vol. 26, no. 11, pp. 2994–3009, 2015.
[41] J. Zhang, Y. Wang, and D. Yang, "CCSpan: Mining closed contiguous sequential patterns," *Knowl.-Based Syst.*, vol. 89, pp. 1–13, Nov. 2015.
[42] X. Chai, X. Jia, Y. Wu, H. Jiang, and X. Wu, "Strict pattern matching with general gaps and one-off condition," *J. Softw.*, vol. 26, no. 5, pp. 1096–1112, 2015.
[43] Y. Wu, K. Zhou, J. Liu, H. Jiang, and X. Wu, "Ming sequential pattern with periodic gap constraints," *Chin. J. Comput.*, vol. 40, no. 6, pp. 1338–1352, 2017.
[44] H. Liu, Z. Liu, H. Huang, and X. Wu, "Sequential pattern matching with general gap and one-off condition," *J. Softw.*, vol. 29, no. 2, pp. 363–382, 2018.
[45] Q. Wang, V. S. Sheng, and X. Wu, "Keyphrase extraction with sequential pattern mining," in *Proc. 31st AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, 2017, pp. 5003–5004.
[46] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.
[47] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Barcelona, Spain, 2004, pp. 404–411.
[48] W. Hadi, Q. A. Al-Radaideh, and S. Alhawari, "Integrating associative rule-based classification with Naïve Bayes for text classification," *Appl. Soft Comput*, vol. 69, pp. 344–356, Aug. 2018.
[49] W. Gan, J. Lin, P. Fournier-Viger, H. Chao, J. Zhan, and J. Zhang, "Exploiting highly qualified pattern with frequency and weight occupancy," *Knowl. Inf. Syst.*, vol. 56, no. 1, pp. 165–196, Jul. 2018.
[50] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.
[51] L. Yang, M. Qiu, S. Gottipati, F. Zhu, J. Jiang, H. Sun, and Z. Chen, "CQArank: Jointly model topics and expertise in community question answering," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage.*, San Francisco, CA, USA, Nov. 2013, pp. 99–108.
[52] [Online]. Available: https://github.com/snkim/AutomaticKeyphraseExtraction
[53] [Online]. Available: https://en.wikipedia.org/wiki/SemEval
[54] [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/



HUITING LIU received the Ph.D. degree from the Hefei University of Technology, Hefei, China, in 2008. She is currently an Associate Professor with the School of Computer Science and Technology, Anhui University, Hefei. Her current research interests include personalized recommendation, text information processing, and data mining.



LILI WANG is a currently pursuing the degree with the School of Computer Science and Technology, Anhui University, Hefei, China. Her main research interests include text information processing and sequential pattern mining.

PENG ZHAO received the Ph.D. degree from the Department of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China, in 2006. She is currently an Associate Professor with the Department of Software Engineering, School of Computer Science and Technology, Anhui University, Hefei. Her current research interests include intelligent information processing, image annotation, and machine learning.

XINDONG WU received the Ph.D. degree in artificial intelligence from The University of Edinburgh, Edinburgh, U.K. He is currently a Yangtze River Scholar with the Hefei University of Technology, Hefei, China, and the Dean of the Mininglamp Academy of Sciences, Mininglamp Technology, Beijing, China. His current research interests include data mining and big data analytics.

Dr. Wu is a Fellow of AAAS. He is the Steering Committee Chair of ICDM. He is the Editor-in-Chief of *Knowledge and Information Systems*.