Knowl Inf Syst (2013) 34:691–724
DOI 10.1007/s10115-012-0480-2

REGULAR PAPER

An automatic keyphrase extraction system for scientific documents

Wei You · Dominique Fontaine · Jean-Paul Barthès

Received: 4 October 2010 / Revised: 15 November 2011 / Accepted: 4 March 2012 /
Published online: 7 April 2012

Abstract Automatic keyphrase extraction techniques play an important role for many tasks including indexing, categorizing, summarizing, and searching. In this paper, we develop and evaluate an automatic keyphrase extraction system for scientific documents. Compared with previous work, our system concentrates on two important issues: (1) more precise location for potential keyphrases: a new candidate phrase generation method is proposed based on the core word expansion algorithm, which can reduce the size of the candidate set by about 75 % without increasing the computational complexity; (2) overlap elimination for the output list: when a phrase and its sub-phrases coexist as candidates, an inverse document frequency feature is introduced for selecting the proper granularity. Additional new features are added for phrase weighting. Experiments based on real-world datasets were carried out to evaluate the proposed system. The results show the efficiency and effectiveness of the refined candidate set and demonstrate that the new features improve the accuracy of the system. The overall performance of our system compares favorably with other state-of-the-art keyphrase extraction systems.

Keywords Information retrieval · Automatic indexing · Keyphrases extraction · Candidate phrase identification · Scientific document processing

A preliminary version of this paper appears in the 2009 IEEE/WIC/ACM International Conference on Web Intelligence [41].

W. You (✉) · D. Fontaine · J.-P. Barthès
HEUDIASYC UMR CNRS 6599, Université de Technologiede Compiègne, Centre de Recherches de Royallieu, BP 20529, Compiègne Cedex, France
e-mail: youwei.utc@gmail.com

D. Fontaine
e-mail: fontaine@hds.utc.fr

J.-P. Barthès
e-mail: barthes@utc.fr

1 Introduction

A keyphrases is a simplex word or a sequence of words that characterizes the theme and content of a document. Automatic keyphrase extraction is defined as the automatic selection of important, topical phrases from within the body of a document[1] [34]. The technique is useful in a variety of applications including text indexing, summarization [39], categorization [12], topic detection [38], or improvement of search results [9,27].

In this paper, we focus on keyphrase extraction for scientific documents. Currently the growing collections of research papers in digital libraries make online search more difficult to meet individual needs. A well-chosen keyphrase list can improve the search precision and help the readers to quickly decide whether the document is related to their domain of interest. Assigning keyphrases manually is time consuming and particularly difficult for scientific papers because of their technical content. Therefore, an automatic keyphrase extraction technique is very important for this type of documents.

Generally speaking, two different approaches can be applied for the task of keyphrases extraction: free indexing or controlled indexing. The first approach freely chooses representative phrases appearing in the text, while the second one selects keyphrases from a controlled vocabulary of related domains. Some previous works in which the MEDLINE database or some other domain-specific thesaurus were used [21,31,32] have proved that controlled indexing tends to yield more precise and more readable results, but it is highly dependent on the quality of the vocabulary. Considering that a complete list of technical terms is not easy to obtain for scientific domains, free indexing is more practical in that case.

The task of keyphrase extraction includes three main steps. First, candidate identification: candidate phrases are identified from the full text document. Second, feature engineering: a set of features are used to calculate the weight of each candidate. Third, keyphrase selection: a supervised or unsupervised model is built to pick up the most important candidates to constitute a list of keyphrases.

Our work mainly improves the first two steps. In the first step, we concentrate on more precise locations for candidate phrases. A newly-designed algorithm based on core word expansion is used to generate a refined candidate set. As the weight of each candidate must be calculated in the second step, this small but effective candidate set leads to great run time savings. During the feature engineering, some new features are introduced, especially a feature related to inverse document frequency, which helps to decide the proper granularity when selecting among a phrase or its sub-phrases. This feature successfully solves the overlap problem that is not rare in the candidate set. Finally, a model is built for ranking and picking up the final list of keyphrases. Experimental results show that our system outperforms other state-of-the-art systems in both accuracy and run-time efficiency.

Our paper is organized as follows: Sect. 2 provides a brief background on the task of keyphrase extraction. Section 3 introduces a new algorithm for generating candidate phrases. Section 4 is mainly about the feature engineering and the keyphrase selection. Section 5 presents the evaluation experiments and results. Section 6 discusses some open issues, and Sect. 7 concludes our work and presents future research directions.

[1] Notice that although some of author-assigned keyphrases are out of text of a document, this paper focuses on developing a technique that extracts keyphrases from within the body of a document.

2 Related work

In the preceding section, we have mentioned that the task of keyphrase extraction can be divided into three steps. This section presents the related works for each step.

The first step is called *candidate phrase identification* or *candidate phrase generation*. Free indexing and controlled indexing adopt different strategies in this step. The main difference is whether the source of terminology is restricted or not: Free indexing freely identifies candidates from the text of the document, while controlled indexing determines candidates by mapping document phrases to vocabulary terms.

In keyphrase extraction, a phrase replaces a word to become the linguistic unit. For free indexing, intuitively, all word sequences appearing in the text can be candidate phrases, and keyphrases are selected from them. Existing approaches usually fall into two categories: n-gram based and part-of-speech (POS) sequence based.

Typical n-gram based approach was adopted in pioneering systems [8,34]: first, the input text is split up according to phrase boundaries (e.g., punctuation marks, dashes, brackets, and numbers). All subsequences of these initial phrases up to a limit length (known as n-grams) are taken as candidate phrases; then, some simple rules are set up to filter meaningless subsequences (e.g., candidate phrases cannot begin or end with a stopword). Some later systems use refined n-gram based approaches: Huang et al. [10] add a filtering phase in which phrases are divided into groups by the number of distinct non-stopwords. Only the winner of each group can remain as a candidate. Wang et al. [38] first extract keyword candidates and then combine them into keyphrases using position information. LZ78 compression algorithm is used by Kumar and Srinathan [18] to prepare a dictionary of distinct n-grams.

POS sequence-based approaches are also widely used. Using POS tagger,[2] Barker and Cornacchia [1] skim a document for base noun phrases (NP) as candidates, which are non-recursive structures consisting of a head noun and zero or more premodifying adjectives and/or nouns. Hulth [11] compares three different phrase selection approaches and concludes that extracting NP-chunks and word sequences matching any of a set of POS tag patterns both give a better precision than n-grams. Kim and Kan [15] analyze the nature and variation of manually provided keyphrases and then select candidates using regular expressions.

The strength of free indexing methods is that they do not require vocabularies or thesauri as external resources and are usually easy to realize. But free indexing may results into a large candidate set that contains a lot of ill-formed phrases. Controlled indexing is also applied by some systems, to avoid ill-formed or inappropriate extracted phrases. Representative examples include the KEA++ system [21] that obtains candidate phrases from a domain-specific thesaurus and Song et al.'s system [31] that studies whether the use of MeSH improves the retrieval performance.

The second step concerns feature engineering. In this step, a number of features are calculated for each candidate phrase to assess its importance. The foremost features are known as TF-IDF and First Occurrence in the KEA system [8]. These two features are considered as two baseline features and are employed in most of the existing systems.

Many other features were developed later: The KEA++ system [21] adds the feature of phrase length in words and the node degree (representing the number of thesaurus links that connect the term to other candidate phrases) as a refinement of KEA. The use of semantic information about terms encoded in a structured controlled vocabulary results in a lowered requirement for training data. Concept subsumption and boosting are used in the KX system

[2] A tagger assigns the most likely single POS tag (noun, adjective, verb, etc.) to each word in a sentence, i.e., the widely used Brill tagger [3].

[28] to merge or re-rank a couple of candidates when one is a specification of the other. Aiming at solving the problem of incoherence, Turney [35] introduces two new sets of features based on the use of web mining to measure the statistical association among candidate phrases.

Document type, context, and structure are also considered as helpful information for feature calculation. Zhang et al. [43] propose to use not only 'global' but also 'local' context information, such as the linkage feature. Kumar and Srinathan [18] use the position of sentence and position of phrases in a sentence. Chen et al. [4] use three new features considering the structure of web pages. For scientific publications, Nguyen and Kan [25] propose the section occurrence vectors based on the idea that keyphrases are distributed non-uniformly in different logical sections of a paper, favoring sections such as introduction, and related work. Similar structural features are also used in some later systems [15,20,26] by adding additional section information. Especially, Berend and Farkas [2] invent features concerning not only the document level but also the corpus level and the knowledge-based level. For instance, they consider whether a phrase is an author-assigned keyphrase in other documents of the corpus. They also rely on external knowledge sources (i.e., Wikipedia) to achieve further enhancements in performance.

Some features have also been proposed, based on the semantic relations among candidate phrases. Kelleher and Luz [14] use the feature Semantic Ratio, which takes the hyperlink information of web documents into account. Huang et al. [10] treat each document as a semantic network and analyze its connectedness and compactness to symbolize the importance of a node which represents a candidate phrase.

Distinctively, Wan and Xiao [37] use a set of neighborhood documents to enhance the single document keyphrase extraction. The system builds a global affinity graph based on all candidate words restricted by syntactic filters in all the documents of the expanded document set, and employs the graph-based ranking algorithm to compute the global saliency score for each word.

Generally speaking, the information such as document structure, hyperlink analysis, or nearest neighbor documents helps to improve the effectiveness of automatic keyphrase extraction. But it is noteworthy that the use of extended information always involves a higher computational complexity than the baseline approach.

The last step is related to building the model for keyphrase extraction. The basic idea of this step is to make use of the results obtained from feature engineering to evaluate the importance of the candidate phrases and to rank them. There are both supervised and unsupervised methods. Supervised models were proposed in pioneering systems and are widely used in later systems. Unsupervised models start to attract interest recently, aiming at achieving comparable result to supervised systems through a deep analysis of the general nature of documents and keyphrases.

In supervised machine learning approaches, the training phase usually includes a classification task: each phrase in the document is either a keyphrase or not. GenEx [34] and KEA [8] are two typical systems based upon this idea: GenEx consists of a set of parameterized heuristic rules that are tuned to the training corpus by a genetic algorithm; KEA employs a Naïve Bayes learning method to induce a probabilistic model from the training corpus. Experiments with the same datasets show comparable performance of these two influential supervised systems. Other supervised methods include Support Vector Machine (SVM) model and Conditional Random Fields (CFR) model. Zhang et al. [43] show that the use of SVM model helps their system to outperform not only baseline methods such as TF-IDF, but also KEA system. The effectiveness of the CFR model are proved in [42] by comparing the accuracy with five different methods including SVM, logistic regression, multiple linear

regression, etc. Lopez and Romary [20] experiment different machine learning models such as Decision tree (C4.5), Multi-Layer perceptron and SVM, and combine these models with boosting and bagging techniques. Supervised methods can adapt to the specific nature of the documents at hand, based on a representative training set. They always achieve satisfactory accuracy and have excellent stability, but they require an annotated corpus and have difficulties in extracting unknown keyphrases.

Unsupervised approaches usually involve assigning a saliency score to each candidate phrases by considering various features [37]. The keyphrase extraction technique proposed by Kumar and Srinathan [18] uses filtration algorithms based upon statistical observations, simple grammatical facts, heuristics, and lexical information. A semantic network structure analysis method is employed by Huang et al. [10]. They use structural dynamics of the network to extract keyphrases. El-Beltagy and Rafea [5] argue that an understanding of the process is helpful to improve keyphrase extraction. Based on this idea, they look into the general nature of documents and keyphrases and build the KP-Miner system. The designers of the three aforementioned unsupervised keyphrase extraction systems conduct experiments to demonstrate that their systems have similar performances than typical supervised systems (Extractor [36] and/or KEA [45]) even without training samples.

3 Identification of candidate phrases

In this section, we present our method for identifying candidate phrases. Compared with simplex words, phrases have complex grammatical structure. Generally, the meanings of a phrase cannot be inferred from the meanings of the words that make it up and we believe that phrases have stronger ability for representing text topics. However, when a phrase is chosen as the language unit to extract, the first problem we have to deal with is phrase identification. Because phrases do not have clear boundaries, it is hard to let the algorithm know what a phrase is.

Previous works usually include all possible n-grams. Although a few rules are then applied to eliminate some ill-formed n-grams (e.g., n-grams that begin or end with a stopword), the number of remaining candidates is still very large.

Some other systems use syntactically motivated boundaries like NP-chunks and POS tags. Such methods indeed decrease the number of candidate phrases; however, they require external supports such as a tagger or a large list of words with all possible parts of speech for each word. The quality of these supports directly affects the performance of the system.

Our method aims at obtaining a refined candidate set without resorting to any external support. To achieve this goal, a two-step method is employed: first, a core word set is proposed to predict potential positions of keyphrases; then, core words are expanded to obtain candidate phrases.

3.1 Text preprocessing

We start our approach by preprocessing the text. The purpose of text preprocessing is to regularize the input files and set phrase boundaries for word sequences. The following modifications are made to the input stream:

- Set tags to mark the text in the document title, the abstract and the first paragraph (see Sect. 4.2 for more details).
- Punctuation marks which are used to separate different semantic fragments, such as comma, full stop, and semicolon, are replaced by phrase boundaries.

- Apostrophes, brackets, numbers, and non-alphanumeric characters are removed.
- Hyphens are kept, and the hyphenated words are considered to be single words.
- Make a special treatment of abbreviations: if a string of capital letters appears in a pair of brackets, we assume it is an abbreviation. All the occurrences of this string in the document are given a special mark. If the abbreviation is included in the final list of keyphrases, the mark can be used to find the expanded form of the abbreviation.
- The entire input stream is converted to lower case except for abbreviations.[3]
- Stopwords are also replaced by phrase boundaries. Because there is not such an *oracle* stopword list that all text processing tools incorporate, to minimize the impact of a special-chosen list of stopwords, we use a widely used list that contains 429 most common stopwords [48]. In particular, we exclude some words that may occur as part of keyphrases (e.g. "of", "for", "and") from the list. The reason for keeping such prepositions and conjunctions is that the lack of such words may cause grammatical mistakes and impact the readability of the final result.

Stemming is also applied during the preprocessing step in many keyphrase extraction systems, because it is a usual approach to recognize terms of equivalent meaning. However, in our system we decided to do this operation in the later steps of core word selection, creating child nodes for expansion trees (see Sects. 3.2 and 3.3). This is because we want to retain the original unstemmed input stream from which the final keyphrase can be picked up, to avoid the grammatical problems caused by stemming.

3.2 Locate candidate phrases using core words

A large number of candidates lead to inefficiency, because the feature weight of each candidate must be calculated in the following step, accounting for a high proportion of system run time. In fact, when analyzing the feature weight of each candidate, we find that some candidates have too low a weight to participate in the feature weight competition. Such weak competitors should not be included in the candidate set.

The problem happens because all positions in the document are considered equal for producing candidate phrases. To address the problem, previous works tried to take into account some position-related information. For example, Witten et al. [40] considered the first occurrence position of a candidate phrase and believed that keyphrases tend to occur in the beginning of documents. Nguyen and Kan [25] proposed that different logical sections of scientific publications contribute keyphrases at different rates. Similar section locality were used by Kim and Kan [15] to identify key sections. Results of these applications suggest the hypothesis that candidates produced at different positions have a different potential to be keyphrases. However, we notice that most of the position-related information is used in the step of feature engineering. We argue that distinguishing strong candidates from weak ones should not only rely on feature engineering. Some filtering can be done as early as in the step of candidate identification.

An intuitive idea is that keyphrases obviously should contain key terms. Here, key terms mean important single words in the document. Key terms can be used as the basis for finding candidate phrases, which will reduce the size of the candidate set to a great extent. The idea of using key terms can be observed in some related works. For example, Turney [35] proposed to use top K most probable phrases (ranked according to baseline features such as

[3] Capitalizations usually carry useful information. That is why we keep abbreviations in upper case. However, we do not think a word in the beginning of a sentence is more important than a word in the middle of a sentence. We convert other input stream to lower case, to normalize the n-grams for the calculation of frequency.

phrase frequency) as a standard for evaluating the top L most probable phrases ($K < L$). The hypothesis is that candidates that are semantically related to one or more of the top K phrases will tend to be more coherent, higher quality keyphrases. Also, in the task of terminology extraction, simplex terms, or so-called "reference terms", are selected first. Then terminologies are extracted by accumulating neighboring words according to some grammar rules [13,22]. During the system design, we notice that the difficulty of using key terms consists in identifying key terms. Therefore, we introduce an important concept—the *core word set*. In the following, we will give the definition and prove that the proposed core word set can work as an alternative of key terms:

Definition 1 For a given document d, the core word set S_{cw} is defined as the top-k most frequent words in d, and S_{cw} should satisfy two additional conditions:

1. S_{cw} does not contain stopwords.[4]
2. All words in S_{cw} map to different stems.

According to our definition, the primary factor for core word selection is the term frequency (TF). The use of TF may evoke another statistical measure which is usually used together with the TF, namely the inverse document frequency (IDF). Although the TF-IDF model works well for many information retrieval problems, when applied to our task, the simple TF-based core word set approach performs better than the TF-IDF model. There are two possible reasons: first, the stopword list we used is complete enough to exclude words with low IDF values; second, in a collection of scientific documents, IDF may eliminate some important single terms, particularly when the documents come from the same or similar domains. For example, if all the documents in the collection are related to web search, the term 'search' may appear in most documents and obtain a low TF-IDF value, although this term might appear in a keyphrase.

The following statistical study proves that our core word set can replace key terms to highlight those positions in the document where keyphrases may appear. A total of 500 scientific papers with 3 to 15 author-assigned keyphrases were randomly selected for our preliminary experiment.[5] We use k to represent the number of core words. Two measures are analyzed when k varies: (i) What is the percentage of author-assigned keyphrases that begin or end with a core word? (ii) What proportion does the total frequency of core words take in the preprocessed document? The results reported in Fig. 1 show that the majority of keyphrases begin or end with words in S_{cw}. When the value of k grows, this proportion increases. Especially, when k increases to 50, no less than 97 % of the keyphrases conform to this rule. On the other hand, compared with this high percentage, the total frequency of core words in the preprocessed document is low. For example, when k equals 50, core words take only 42 % of the positions in the preprocessed document.[6] In other words, we can cover almost all the keyphrases using candidate phrases produced by only 42 % positions of the preprocessed document.

This experiment shows that the core word set is very helpful for locating candidate phrases, pointing out positions more likely to produce possible keyphrases. 50 is suggested as an

[4] A complete stopword list in [48] is used here. The retained stopwords (mentioned in Sect. 3.1, such as "of" and "for") will not appear in S_{cw}.

[5] The papers used in the preliminary experiment and the dataset described in Sect. 5.1 come from the same source. They have similar structure and length.

[6] Here, 'position' means the position taken by a word. For example, if the length of a document is 6,000 words, then the document has 6,000 positions.

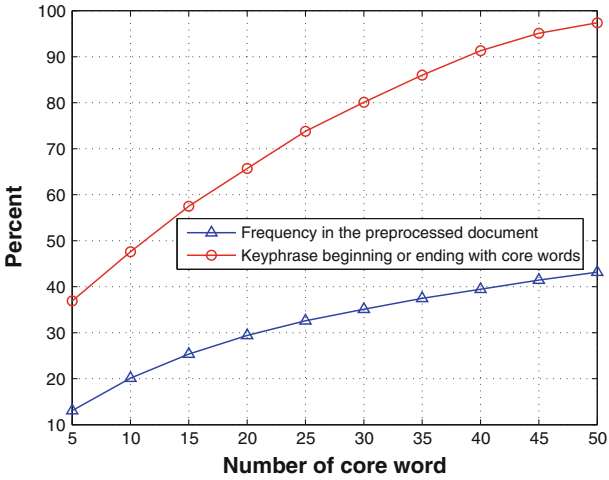


Fig. 1 Comparison between core word frequency and core word covering ability for keyphrase extraction

Table 1 Potential candidates for core word w_p

1-gram	w_p
2-gram	$w_{p-1}w_p$ / w_pw_{p+1}
3-gram	$w_{p-2}w_{p-1}w_p$ / $w_pw_{p+1}w_{p+2}$
4-gram	$w_{p-3}w_{p-2}w_{p-1}w_p$ / $w_pw_{p+1}w_{p+2}w_{p+3}$

ideal number of core words to have a good enough covering ability for generating candidate phrases. This value is applied in the following calculations and experiments.

With the core word set, we can only focus on the core words and their surrounding positions. Statistics show that a keyphrase with length greater than four is quite rare [8], so it is reasonable to set 4 as the max length of the keyphrases. In our system, all n-grams ($1 \leq n \leq 4$) beginning or ending with a core word can be potential keyphrases.[7] For example, if w_p is a core word, where p is its position, then all the candidates will appear within the following word sequence:

$$w_{p-3}w_{p-2}w_{p-1}w_pw_{p+1}w_{p+2}w_{p+3}$$

and the number of potential candidates produced by position p is 7, including the core word itself, two 2-grams, two 3-grams, and two 4-grams. Table 1 lists all such candidates. Thus, by removing all word sequences without core words, we successfully decrease the number of candidate phrases.

To obtain the core word set for a given document, all the non-stopwords in the document are ranked by their frequency. Porter stemming algorithm [29] is then applied to combine words with the same stem. The top-k word stems are picked up as the result we need. To facilitate the subsequent expansion algorithm (see Sect. 3.3), all occurrence positions for each core word are also recorded.

[7] The above experiment proves the covering ability of this n-grams set, so we do not consider n-grams containing a core word in the middle (noted as G_m). In fact, further analysis shows that if an G_m is important, it is likely to contain more than one core word, which means the beginning or ending word of the G_m may also be a core word. Consequently, most of the G_m are already included in the current n-gram set.

Algorithm 1: phrase generation based on the forward expansion of core words

Input: preprocessed input file, core word set S_{cw} and the position information of each core word
Output: candidate phrases with frequency and position information

for each core word W_{core} in S_{cw} **do**
  create the root node of the expansion tree:
    {$key\ value \leftarrow W_{core}$;
     $position\ array \leftarrow$ positions of W_{core};
     $tag \leftarrow$ non-leaf;}
  while the on-building tree contains non-leaf nodes which were not expanded **do**
    pick up a non-leaf node to expand, according to Depth- First order;
    if the depth of this node is <4 **then**
      add the word in the next position of each element in the position array into a word set $S_{children}$;
      apply Porter stemming to $S_{children}$;
      for each different stem W_i in $S_{children}$ **do**
        create a child node:
          {$key\ value \leftarrow W_i$;
           $position\ array \leftarrow$ positions of W_i appearing after the word of its father node;
           if count [$position\ array$] $< \delta$ **or** W_i is a phrase boundary **then**
             $tag \leftarrow$ leaf;
           else
             $tag \leftarrow$ non-leaf;
           end if }
      end for
    else
      create an empty leaf node as the child node;
      continue;
    end if
  end while
  for each leaf node of the tree **do**
    backtrack its ancestors until a non-stopword node N_{end} is found;
    add the word sequence from the root node to node N_{end} as a candidate phrase;
  end for
end for

3.3 Phrase generation algorithm based on core word expansion

With the help of core words, the scale of candidate phrase searching is evidently reduced. From Sect. 3.2, we know that, for each occurrence of a core word, the maximum number of candidates is seven (when the consecutive positions do not contain phrase boundaries). But many of the candidate n-grams may be semantically meaningless due to incompleteness or redundancy. Actually, according to the rules of phrase segmentation, each occurrence

of a core word belongs to only one most likely candidate phrase. Therefore, we propose a new algorithm, called *core word expansion algorithm*, for further reducing the number of candidates.

The core word expansion is inspired by the compound noun bracketing technique in which the syntactic bracketing should be decide to make clear the semantic relations among the composite nouns [16,23,30]. Similar to the bracketing task that usually has either left-branching or right-branching, our core word expansion algorithm works in two directions: forward expansion for finding the most probable candidates beginning with core words, and backward expansion for finding the most probable candidates ending with core words. However, the core word expansion differs from the compound noun bracketing in that the input is a word sequence but not a given noun phrase; therefore, our algorithm needs to decide the length of the phrase automatically.

The main idea of this algorithm is based on the core word expansion tree. The expansion tree has the core word as the root node and its subsequent or preceding words as child nodes. The branches of the tree correspond to different word sequences beginning or ending with this core word. Each node of the tree has three components: (i) a word as the key value of the node; (ii) an array to record the positions of the word; and (iii) a binary tag to indicate whether the node is a leaf node or not.

A threshold δ is set to filter word sequences with low frequency, which is known as the least allowable seen frequency factor. In our system, the value of δ is set to 3, meaning that candidate phrases occurring <3 times in the document are removed. The same value was used in [5], which was shown to work well for scientific documents.

Algorithm 1 describes the steps of the forward expansion algorithm in details. When a new node is created, we use the operations in "{ }" to assign values for the three components of this node. The operation in step 27 ensures that a candidate phrase will not end with a stopword.

Here we use an example to explain the process of Algorithm 1. To make it easy to follow, we use the real text extracted from the preceding part of this paper. In this example, 'keyphrase' is a core word. In Table 2, the second column lists four contexts of the occurrences of the core word. The third column of each occurrence contains three sub-rows, and each of them is a different form of the related 4 grams. The first sub-row is the related word sequence beginning with the core word in the context (the underlined part in the second column, e.g. 'keyphrases from a controlled vocabulary' for the occurrence ♯2); The second sub-row is the 4-gram after preprocessing (e.g. 'keyphrases | controlled vocabulary' for the occurrence ♯2); And the third sub-row is the symbolized 4-grams (e.g. 'x | yz' for the occurrence ♯2). We use symbol '|' to represent a phrase boundary and different letters to represent different word stems, for example, letter 'x' for 'keyphrase' (the stem of 'keyphrase' and 'keyphrases'), letter 'y' for 'extract' (the stem of 'extraction'), and so on.

Figure 2 gives the forward expansion tree of this example, where δ is set to 2, tag '$\wedge$' is used to indicate a non-leaf node and '!' for a leaf node. In the following, we explain the process of the generation of the tree and the candidates. Our aim is to find candidate phrases beginning with x. We record the positions of x, that is p_1, q_1, r_1 and s_1, and make $x(p_1, q_1, s_1, r_1)$ to be the root node of the tree. Then we check p_2, q_2, s_2 and r_2, and get two child nodes of $x(p_1, q_1, s_1, r_1)$: $y(p_2, r_2, s_2)$ and | (q_2). As the frequency of y is more than 2, $y(p_2, q_2, r_2)$ is determined to be a non-leaf node. And | (q_2) is a leaf node because the word in position q_2 is a phrase boundary. With similar operations for the root node, we get nodes $z(p_3, r_3)$ and $w(s_3)$ as the child nodes of $y(p_2, r_2, s_2)$. Because w appears only once, $w(s_3)$ is a leaf node. And non-leaf node $z(p_3, r_3)$ has two child nodes $u(p_4)$ and $v(r_4)$. Both these two nodes have a depth of 4, so they are leaf nodes. Until now, all the non-leaf

Table 2 A real-text example for explaining the forward expansion of core word: the context of core word and different forms of related 4-grams of forward expansion

Occurrence	Context	4-gram
♯ 1	Automatic keyphrase extraction techniques play an important role in many tasks including indexing, categorizing, summarizing, and searching.	keyphrase extraction techniques play keyphrase extraction techniques play *x* *y* *z* *u*
♯ 2	The first approach freely chooses representative phrases appearing in the text, while the second one selects keyphrases from a controlled vocabulary of related domains.	keyphrase from a controlled vocabulary keyphrase | controlled vocabulary *x* | *a* *b*
♯ 3	The keyphrase extraction technique proposed by Kumar and Srinathan uses filtration algorithms that are based upon statistical observations, simple grammatical facts, heuristics, and lexical information.	keyphrase extraction technique proposed keyphrase extraction technique proposed *x* *y* *z* *v*
♯ 4	Stemming is also applied in the preprocessing step for many keyphrase extraction systems, because it is a usual approach to recognize terms with equivalent meaning.	keyphrase extraction systems, keyphrase extraction systems | *x* *y* *w* |

nodes of the on-building tree have been expanded. We obtain a whole tree for x-beginning candidate phrases. The backtrack for leaf nodes $u(p_4)$ and $v(r_4)$ generates the candidate 'xyz'. Similarly, we can catch the other two candidates: 'xy' and 'x'. Coming back to the original text, this example generate three candidates, 'keyphrase extraction technique' for the occurrence ♯1 and ♯3, 'keyphrase extraction' for the occurrence ♯2, and 'keyphrase' for the occurrence ♯4.

The backward expansion algorithm is quite similar to Algorithm 1. But we use 'last position' instead of 'next position' in step 9, and add a reverse-order operation in step 28, that is, from the node N_{end} to the root node.

The forward expansion algorithm creates a forward expansion tree for each core word. For a given core word W, each of its occurrences produces a word sequence beginning with W. From Table 1, we know that the maximum length of the word sequence is four, and the sequence may contain at most four potential candidates.[8] Each word sequence corresponds

[8] In Table 1, w_p is considered as an occurrence of the core word W. w_p, $w_p w_{p+1}$, $w_p w_{p+1} w_{p+2}$ and $w_p w_{p+1} w_{p+2} w_{p+3}$ are four potential candidates beginning with W. Forward expansion selects one among these four candidates.

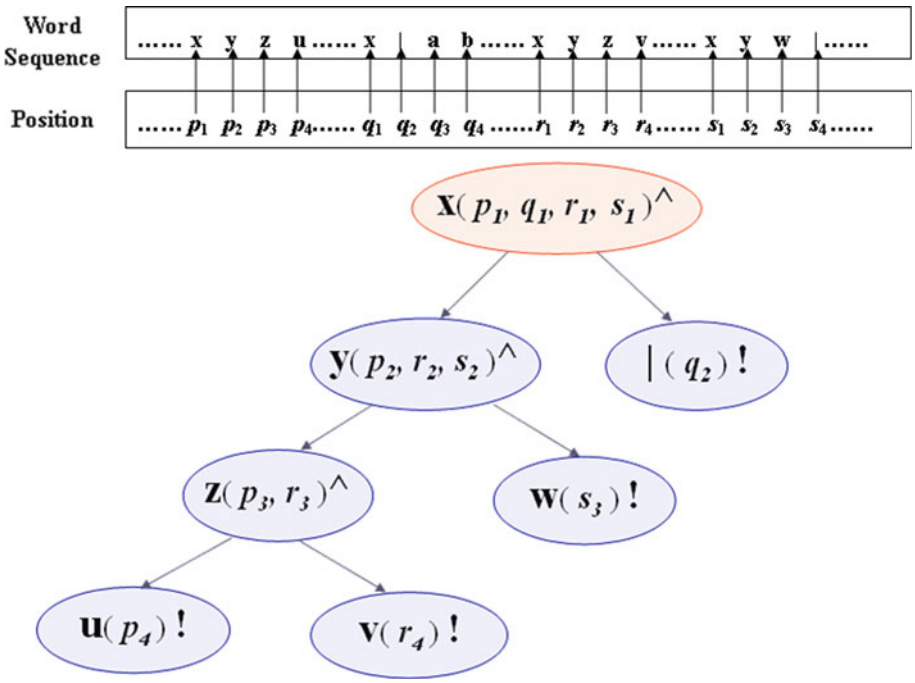


Fig. 2 Example of forward core word expansion algorithm

to a branch of the tree. More precisely, if we use set A to represent all the occurrence positions of W, set B to represent all the different word sequences, and set C to represent all the branches, then the function $f : A \rightarrow B$ is a surjection, and the function $g : B \rightarrow C$ is a bijection. According to the algorithm, each branch will generate only one candidate phrase, which ensures that each occurrence of W has only one candidate for forward expansion. In other words, for each occurrence of W, forward expansion helps to reduce the number of potential candidates from four to one. Looking back to the example in Table 2, for the occurrence ♯1 of the core word, four potential candidates are 'keyphrase', 'keyphrase extraction', 'keyphrase extraction techniques', and 'keyphrase extraction techniques play'. The corresponding branch in Fig. 2 is '$xyzu$'. Steps 26–28 of Algorithm 1 ensure that the only candidate generated by the branch is 'xyz', which means the algorithm chooses 'keyphrase extraction techniques' among the four potential candidates. The backward expansion algorithm works in a similar way, which helps to choose the most likely candidate among four potential candidates ending with W.[9] Therefore, using the expansion tree, each occurrence of a core word will produce at most two candidates, one for forward expansion and the other for backward expansion. Hence, the maximum number of candidates for each occurrence of a core word can be reduced from seven to two.

For a given document, the number of core words is a constant. We need two expansion trees for each core word. The maximum depth of an expansion tree is four. And for each node of the tree, the size of its position array is equal to the sum of all its children's position array

[9] Similar to forward expansion, backward expansion chooses a candidate among four potential candidates ending with W: w_p, $w_{p-1}w_p$, $w_{p-2}w_{p-1}w_p$ and $w_{p-3}w_{p-2}w_{p-1}w_p$. Therefore, from the seven potential candidates the expansion process will choose at most two of them.

size. Therefore, the maximum number of nodes for an expansion tree is $(3 \times freq_{root} + 1)$, where $freq_{root}$ is the frequency of the core word as the root. However, in practical applications, the average number of nodes is much smaller than this maximum value because of the repeated occurrences of candidate phrases. The time complexity of the expansion tree construction equals the complexity of depth-first traversal, which is $O(n)$. This complexity is comparable to other keyphrase extraction systems, meaning that our algorithm successfully reduces the size of the candidate set without increasing the computational complexity.

4 Feature engineering and keyphrase selection

From Sect. 3, we obtain the resulting candidate set consisting of stemmed word sequences. This section aims at identifying a subset containing the most important candidates, ranking them and generating the final result. The main method we use is feature weight calculation.

For features, we mean some properties or attributes that can reflect how well a candidate phrase represents the topic and content of the document. Feature weight is an index of phrase importance and is used for ranking and selecting the final keyphrases. In our system, the features used are divided into three groups: position features, statistical features, and a granularity-related feature.

4.1 Position features

Position features are related to the position of the given phrase in the text. The basic idea is that different positions in scientific publications contribute keyphrases at different rates [25,40]. Two features, *first occurrence position* and *paragraph position*, are included in this group.

fop *First occurrence position* is based on the belief that the more important a phrase is, the sooner it will appear in the document. Proposed in pioneering systems [34,40], this widely used feature represents the position where a candidate phrase first appears within an input document. It is measured by how many words appear in front of the phrase. In our system, this feature is calculated by Eq. (1)

$$fop(p) = lg(nw_{pre}(p, d) + \alpha) \tag{1}$$

where $nw_{pre}(p, d)$ is the number of words preceding the first occurrence of phrase p in document d. α is an adjustable parameter that tunes the contribution of the fop feature. As the minimum value of $nw_{pre}(p, d)$ is 0, and the value of $\lg(nw_{pre}(p, d)+\alpha)$ should be >0, α should be >1. When the value of α increases, the fop difference between phrases with different $nw_{pre}(p, d)$ values decreases, which means the bigger the α, the smaller effect the fop feature has in the final weight calculation. The setting of α will be reported later in Sect. 5.2.

pp *Paragraph position* is a feature that distinguishes the phrase position in different parts of a document. Academic publications tend to follow a consistent sequential structure: with a title, followed by an abstract, an introduction, related work, methods, evaluation, conclusions, and references [25]. pp can be considered as a feature attempting to identify key sections. Statistics show that the title, the abstract, and the first paragraph usually highlight the topic and contain the core concepts of the whole text. Phrases appearing in the three parts have higher chances to be keyphrases [38]. Therefore, we believe that occurrences in the title, abstract, and first paragraph are more important than in other parts of the document and

differentiate them by assigning a higher pp score.[10]

$$score(pp(p)) = \begin{cases} 7 & pp(p) = title, \\ 3 & pp(p) = abstract, \\ 2 & pp(p) = first\ paragraph, \\ 1 & otherwise. \end{cases} \tag{2}$$

In Sect. 3.1, we have mentioned that tags are set to identify the title, the abstract, and the first paragraph of a document. In our system, a program was developed to complete this operation automatically and the accuracy was test by a preliminary experiment. The experiment was carried out on the dataset PreExp (see more details about the dataset in Sect. 5.2). The results showed that the accuracy of the recognition of title, abstract, and first paragraph was 97.6, 92.6, and 90.7 % respectively. The erroneous cases were corrected manually.

4.2 Statistical features

Statistical features are based on statistical observations. It is believed that different frequency of occurrence in the document conveys different importance. Three features, *phrase frequency*, *length in words*, and *core word-related feature*, are included in this group.

pf *Phrase frequency* is a feature concerning the number of times a phrase occurs in a document. We assume that keyphrases will tend to have a higher frequency than other phrases.

$$pf(p) = \sum_{i=1}^{num_p} freq(p_i) \tag{3}$$

where p_i is one of the forms of the phrase p, num_p is the number of different forms of p in the document, and $freq(p_i)$ is the frequency of p_i in the document. All the p_i map to the same word-stem sequence. And $pf(p)$ finally equals the sum of the frequency of all associated phrases in the document that have the same stemming result to p.

lw *Length in words* is a feature related to the number of words contained in the phrase.

$$lw(p) = \log(n_w(p) + 1) \tag{4}$$

where $n_w(p)$ equals the number of single word stems appearing in the phrase p. The reason for introducing this feature is to boost the frequency value of the compound terms. Looking at almost any document, one can observe that the occurrence of compound terms is less frequent than the occurrence of single terms within the same document [5]. Therefore, ranking based on phrase frequency tends to assign higher score to single or shorter terms. In fact, the lw feature can be seen as a boosting factor to balance the bias towards compound terms with higher $n_w(p)$ values.

cwr *Core word related* feature is a weighting function reflecting the importance of the core words contained in a phrase.

$$cwr(p) = \prod_{i=1}^{n_c(p)} \lg(k + \beta - r_i) \tag{5}$$

[10] Conclusions may also be an important section, but we do not include them in this feature. Conclusions do not always appear in a single paragraph, and are usually difficult to separate them from the future work. This causes trouble for identifying conclusions automatically.

where $n_c(p)$ is the number of core words contained in phrase p, k is the number of core words in the document (set to 50 in our system), and r_i is the frequency rank in the core word set of the ith core word. In this feature, we use the rank rather than the frequency to reflect the importance of core words. Core words with adjacent ranks may have different frequencies, which may reduce or even conflict with the impact of the pf feature. By comparison, rank can differentiate the core words with equal interval, and therefore, it is more appropriate for calculating cwr.

In Eq. (5), β is an adjustable parameter tuning the contribution of the cwr feature. The setting of β will be reported later in Sect. 5.2. As the minimum value of $(k - r_i)$ is 0, and the value of $lg(k+\beta-r_i)$ should be >1, β should be >10. When the value of β increases, the cwr difference between phrases containing different core words decreases, which means the higher the β, the smaller effect the cwr feature has in the final weight calculation. The setting of β will be reported later in Sect. 5.2.

4.3 Granularity-related feature for overlap elimination

The position features and statistical features we mentioned before are used to rank the candidate phrases. However, candidates with high ranks cannot be directly extracted to constitute the final keyphrase list. One problem we have to deal with is the overlap phenomenon, which is not rare in the candidate set. The overlap means one phrase and its sub-phrases appear in the candidate set at the same time. For example, in the example given in Sect. 3.3, the final result contains both 'keyphrase extraction' and 'keyphrase extraction technique', the first phrase being part of the second one. In our system, such overlap phrases are considered as one phrase with different granularities. Our task is to select a proper granularity that can represent the meaning of the phrase most precisely.

To deal with this problem, methods based on frequency differences were adopted in previous systems [5,18]. Supposing p_1 and p_2 coexist in the candidate set, and p_1 is a sub-phrase of p_2, the frequency of p_1 is decremented by the frequency of p_2, and both of their weights are re-calculated.

The same operation is also applied in our system. However, although this method makes the statistic of frequency fairer and more precise, there is still the possibility to have both p_1 and p_2 in the final result. Such undesirable coexistences cause redundance. Moreover, as overlap phrases deprive of the opportunities for other important candidates, it also results in an information loss for the users, especially in a keyphrase list with limited size. Therefore, a new feature is proposed to eliminate overlap:

$idfd$ *Inverse document frequency difference* is used to help us make the decision for granularity selection.

When facing the overlap phenomenon, on the one hand, we prefer to select the longer phrases, because the more words a phrase contains, the more information it provides to users. On the other hand, longer word sequences run the risk to have unrelated words to the meaning at hand or to have some redundant words that do not provide practical information. Concretely, when p_2 and its sub-phrase p_1 occur at the same time, we tend to retain p_2 if the words in p_2 that are absent from p_1 can help the users gain more knowledge specific closer to the given document.

We have mentioned the inverse document frequency (IDF) measure in Sect. 3.2. Although it is not suited for core word selection, it is helpful in the overlap elimination task. IDF measures the document frequency of terms in a collection of documents. It reflects the general importance of terms. In a given corpus, a term with high IDF value means only a few number

of documents contain this term. Therefore, this special term is likely to distinguish these documents from others, which implies a better representativeness for documents.

The $idfd$ feature focuses on the different words between p_1 and p_2 evaluate their influence on the whole phrase.

$$idfd(p_1, p_2) = \frac{\sum_{j=1}^{nwd(p_1,p_2)} idf(dw_j(p_1, p_2))}{nwd(p_1, p_2)} - \frac{\sum_{i=1}^{lw(p_1)} idf(w_i(p_1))}{lw(p_1)} \tag{6}$$

where $ndw(p_1, p_2)$ is the number of different words between p_1 and p_2, $dw_j(p_1, p_2)$ is the jth different word between p_1 and p_2. $lw(p)$ is the number of words in the phrase p. And in Eq. (6)

$$idf(w) = \log \frac{N}{df(w)} \tag{7}$$

where N is the total number of documents in the corpus and $df(w)$ is the number of documents containing the term w in the corpus.

In fact, Eq. (6) compares the average IDF value of the new added words in p_2 with the average word IDF value of p_1. If $idfd(p_1, p_2) > 0$, we know that the new added words bring more specific information to the original phrase. In this case p_2 is chosen; otherwise, we retain p_1.

4.4 Ranking and keyphrase selection

From Sect. 4.1 to 4.3, we listed and explained all the features we use in our system. In this subsection, we will describe how we use the feature values to rank all the candidate phrases. In this step, some previous works choose a separate linear weighting coefficient per used feature [18]. This method has a problem of coefficient determination. Another well-known solution is the supervised machine learning approach [8,11,34]. For a training set of labeled instances, a vector containing all the features is used to represent a candidate. The ranking function compares the confidence of belonging to the keyphrase class for each candidate. Although this method is beneficial, it involves additional phases of preparing the training set and the learning process.

In our system, a simple ranking scheme is applied. The statistical features in conjunction with the position features are used to rank the candidate phrases. The granularity related feature helps to filter the sorted candidate list. Similar schemes have been used in [5], which have shown to yield performances comparable to supervised approach.

According to the property of the feature, a higher pf value indicates a higher importance. This also applies to the lw feature and the cwr feature. Conversely, the lower fop value a candidate has, the more important this candidate is. In consequence, Eq. (8) is used to calculate the merged feature weight of a phrase p:

$$weight(p) = \frac{\sum_{i=1}^{pf(p)} score(pp_i(p)) \times lw(p) \times cwr(p)}{fop(p)} \tag{8}$$

The final output of our system is a sorted list with a certain number of keyphrases according to the user's requirement. Assuming that the number of keyphrases specified by the user is n, the process for producing results can be described as follows:

1. All the candidate phrases are sorted in descending order of their weight.
2. Top n candidates are selected.

3. If there are overlaps among these n candidates, granularity-related features are used to eliminate such redundancies.
4. The subsequent phrases in the sorted candidate set are used to fill the vacancies.
5. Repeat step 3 and 4 until n non-overlap phrases are obtained.
6. In case a phrase is determined to be a keyphrase, its unstemmed form in the original document is presented to the user. When several variations of a phrase occur, the most frequent version is chosen.

5 Evaluation

Experiments were designed and conducted to evaluate the system described in this paper. The experiments mainly include three parts: (i) test the quality of the refined candidate set; (ii) analyze different features we use for feature engineering; and (iii) make comparisons among different systems to see whether the overall performance of our system is comparable to other keyphrase extraction systems.

5.1 The datasets

Two datasets are used in the experiments. The first one is the SemEval-2010 data for the task of 'Automatic Keyphrase Extraction from Scientific Articles', published by the ACL SemEval workshop [17]. The dataset contains a total of 244 articles and is divided into three parts: 144 for training set, 40 for trial set (trial documents are the subset of training documents), and 100 for test set. All the articles are conference and workshop papers collected from ACM digital library. The average length of a paper is between six and eight pages including tables and pictures. For all collected papers, three sets of answers were provided: author-assigned keyphrases, reader-assigned keyphrases, and finally a set that is simply a combination between the 2 previous sets. To collect reader-assigned keyphrases, annotators were hired and each annotator was assigned an average of 5 papers. The annotation were done within 10–15 min per paper. All reader-assigned keyphrases were extracted from the papers whereas some of author-assigned keyphrases do not occur in the content [44,47].

The advantage of using the SemEval-2010 is that it is a standard dataset for testing keyphrase extraction systems. However, because the test set of SemEval-2010 only contains 100 conference papers in the domain of computer science, the scale is a little too small and there is a lack of variety in the type and the research area of documents. Consequently, we collected another dataset for our experiments, called VarTypes, containing a total 765 documents. The dataset is composed of five types of scientific documents: book chapter, conference paper, journal articles, technical report and scientific web pages. All the documents were obtained from well-known web sites. ScienceDirect (http://www.sciencedirect.com), ACM digital library (http://portal.acm.org), Cogprints (http://cogprints.org) and Springerlink (http://www.springerlink.com) are the sources of the first four types of documents. Scientific web pages were downloaded from Wikipedia (http://en.wikipedia.org), and five experts were asked to assign manual keyphrase lists for the web pages. The dataset covers more than ten research areas, including Biology, Biomedicine, Chemistry, Psychology, Linguistics, or Computer Science. Some detailed statistics for both datasets are reported in Table 3. For the experiments in this section, our system only run the test set of the SemEval-2010 but do not make use of the trial and training data. In the remaining part of the section, unless mentioned otherwise, the experimental results of the dataset SemEval-2010 are based on its test set.

Table 3 Detailed statistics for both datasets

Dataset	Number of documents	Min. size (in number of words)	Max. size (in number of words)	Average document size (in number of words)
SemEval–2010	100	4,063	14,170	8,014
VarType				
Book chapter	105	4,893	21,677	11,511
Conference paper	200	1,957	9,551	4,682
Journal article	350	2,491	18,455	7,073
Technical report	60	2,001	8,678	4,902
Scientific web page	50	508	3,689	1,775

When the experiments begin, we make sure that the author-assigned keyphrase lists have been removed from the documents.

5.2 Parameter setting

The proposed keyphrase extraction technique contains two parameters in the step of feature engineering. In this subsection, we describe how the values of the parameters are set in our system.

As we have explained in Sect. 4, the parameter α used in Eq. (1) tunes the contribution of the fop feature in calculating the weight of candidates. The value of α should be >1. Similarly, the parameter β used in Eq. (5) tunes the contribution of the cwr feature in calculating the weight of candidates. The value of β should be >10.

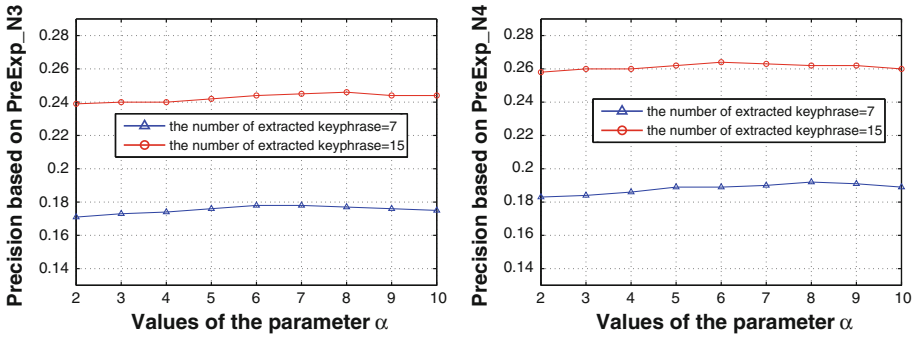
To set the values of α and β, we carried out preliminary experiments in which a data set of 500 documents was constructed. The dataset, called PreExp, was randomly selected from the same source of the dataset VarTypes. We specified that PreExp should contain all the five types of scientific documents that exist in VarTypes. However, the number of documents of each type was randomly decided. We also make sure that PreExp and VarTypes do not have overlapping documents. When the preliminary experiments begin, we first randomly divided PreExp into four equal-size subsets, each containing 125 documents. We used two of them (denoted as PreExp_N1 and PreExp_N2) to select the value of α and β and used the other two (denoted as PreExp_N3 and PreExp_N4) to test whether the obtained value works generally well with different document sets.

Our system was then used to extract seven and fifteen keyphrases from the documents of PreExp_N1 and PreExp_N2 over a set of iterations. The precision in each iteration was logged. These iterations were carried out using a data range of [2, 10] for α (with 2 increments) and [11, 19] for β (with 2 increments). We recorded the (α, β) pairs that resulted in maximum precision across all iterations and such pairs are reported in Table 4. Notice that when a cell of the table contains more than one (α, β) pair, it means all these pairs result in the same maximum precision (with accuracy of three decimal points, e.g., the precision equals to 0.235).

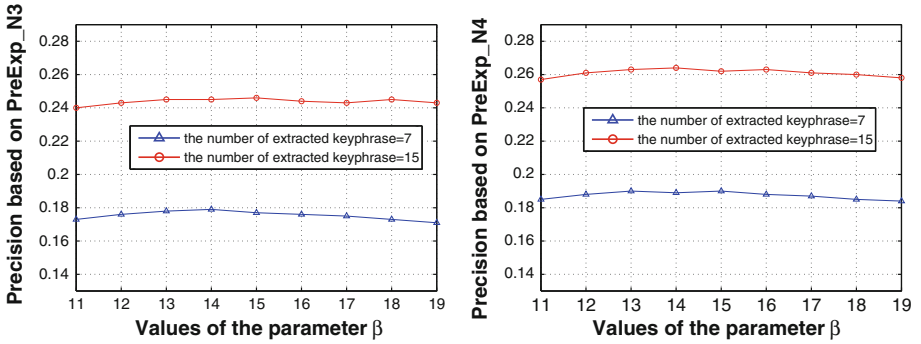
From the results in Table 4, we may explore that the best combination of α and β do not change a lot across all iterations. In another words, neither the change of dataset nor the change of the number of extracted keyphrase seems to have significant effect on the best value of the parameter pair. Beacuse the frequently repeated pairs include (6, 13), (8, 13) and (8, 15), it is reasonable to set α to 7 and set β to 13.

Table 4 Best (α, β) pairs based on the dataset PreExp_N1 and PreExp_N1 when 7 and 15 keyphrases are extracted

Dataset	Number of extracted keyphrases	
	7	15
PreExp_N1	(6,13), (6,15)	(4,17), (6,13), (8,13), (8,15)
PreExp_N2	(8,13), (8,15), (10,13)	(6,13), (6,15), (8,13), (8,15), (10,15)



(a) *the first round: the parameter* β *is set to a fixed value of 13*



(b) *the second round: the parameter* α *is set to a fixed value of 7*

Fig. 3 The generality of the parameter α and β: the change of average precision when one of the parameter is set to a fixed value, based on the dataset PreExp_N3 and PreExp_N4

To further test the generality of the obtained value, another experiment was performed based on the dataset PreExp_N3 and PreExp_N4. In this experiment, for the first round, we let β be the fixed value of 13 and we change α from 2 to 10, with a step size of 1. The precision in each iteration was logged and reported in subfigure (a) of Fig. 3. The second round is similar to the first round, but with a fixed α of 7 and a data range of [11, 19] for β. The subfigure (b) of Fig. 3 reported the results of the second round.

The results presented in Fig. 3 show a good generality of the obtained parameter pair $(\alpha, \beta) = (7, 13)$. When α changes with a fixed β of 13, the best precision was usually achieved when α equals 7 or a value near to 7 (e.g., 6 or 8). This conclusion is supported by both the results based on the dataset PreExp_N3 and PreExp_N4. Similarly, when β changes with a fixed α of 7, the best precision was usually achieved when β equals 13 or

Table 5 Efficiency of the refined candidate set

Document information	Number of document	Average words per document	Average words per preprocessed document
	865	6710	3921
Result of candidate set generation	Number of candidate phrase produced by		
	Our system		KEA
	412		1659

a value near to 13 (e.g., 14). Moreover, observing all the curves in the figure, we find that the average precision does not show significant change within the given range of parameter. Consequently, we may conclude that the performance of our system does not closely rely on the setting of the parameters. The system performs stably, provided the parameters are set within a proper range.

In summary, based on the preliminary experiments we have carried out, the parameter α and β can be set to constant 7 and 13, respectively. Such constants work generally well in the task of keyphrase extraction for scientific documents.

5.3 Evaluating the refined candidate set

The most obvious advantage of our system is the refined candidate phrase set generated by the core word expansion-based algorithm. The experiment in this subsection is designed to test whether the method in Sect. 3 can really reduce the size of the candidate set. Comparisons are made between our system and the KEA system[11] [40,45] designed by Witten et al. We chose KEA for two reasons: (I) KEA is an open-source system and it is able to track the results of each processing step, i.e., the result produced by the candidate identification step. (II) As a pioneering system, KEA proposes baseline method of candidate identification that are stilled applied in most existing systems. The result reported in Table 5 is based on the integration of two datasets, 865 documents in total. It shows that our system successfully reduces the number of candidate size by about 75 %. As the weight of each candidate should be calculated in the subsequent step, this smaller candidate set means great run time savings.

Another issue is related to the effectiveness of the refined candidate set. Studies are made to test how many author-assigned keyphrases actually exist in full text and how many are collected in our refined candidate set. Statistics for both datasets are given in Tables 6 and 7. The percentage of coverage shows that if the author-assigned keyphrases or the reader-assigned keyphrases appear in the full texts, more than 89 % of them will be included in our candidate set. Moreover, the reported results only count the exact matches. In fact, if we extend the granularity limitation,[12] the candidate set will cover more author-assigned keyphrases.

[11] The 3.0 version of KEA is used for all the related experiments in this section.

[12] When we extend the granularity limitation, phrases that have different granularities but the same main meaning can be considered as a match. For example, the phrase pair "support vector machine" (assigned by the author) and "support vector machine algorithm" (extracted by the system) are not an exact match. However, as the two phrases convey almost the same concept, this pair will be counted as a match in the extended condition.

Table 6 Effectiveness of the refined candidate set: based on the dataset SemEval

Keyphrase type	Average number of keyphrases for each document	Percentage of keyphrase in		Percentage of coverage
		Full text (%)	Candidate set (%)	
Author-assigned	3.88	84.3	**78.1**	**92.6**
Reader-assigned	12.04	100	**89.0**	**89.0**
Combined	14.66	95.8	**85.5**	**89.3**

Table 7 Effectiveness of the refined candidate set: based on the dataset VarTypes

Document type	Average number of author-assigned keyphrases	Percentage of author-assigned keyphrase in		Percentage of coverage
		Full text (%)	Candidate set (%)	
Book chapter	5.97	88.7	**83.4**	**94.1**
Conference paper	5.41	76.4	**68.9**	**90.0**
Journal article	6.34	81.8	**73.1**	**89.4**
Technical report	4.78	69.0	**64.1**	**92.9**
Scientific web page	4.22	73.5	**65.9**	**89.7**

Table 8 A summary of features used for feature weight calculation

Feature name	Feature description
fop	Number of words appearing in front of the phrase in the document
score (pp)	Position score according to the phrase occurrence in different parts of document
pf	Phrase frequency in the document
lw	Number of single word stems contained in the phrase
cwr	Importance of the core words contained in the phrase
idfd	Inverse document frequency difference between two overlapping phrases

5.4 Analysis of individual features

In this subsection, we analyze the utility of the various features used for feature engineering described in Sect. 4. Here, we use Table 8 to give a summary of all these features.

We repeat the keyphrase extraction experiment with the output number of 7. During the experiments, we vary the features we use. At each iteration, we remove a single feature from the full feature set. Assuming independence between the different features,[13] a decrease in accuracy indicates how much the removed feature contributes to the overall accuracy. Table 9 reports the of the feature analysis experiments.

Most numbers in Table 9 are negative, which means that most of the time each feature makes a positive contribution to the system precision. From the results, we notice that the pf feature and $idfd$ feature contribute much to the system precision. We also observe that feature contribution varies for different types of documents. A typical example is the fop feature. A possible reason is that fop promotes the phrases appearing in the front part of

[13] There is a exception for the $score(pp)$ feature. Since $score(pp)$ assigns the position score for each occurrence of a phrase, this feature works only if the phrase frequency is taken into account. When pf is removed, $score(pp)$ is removed at the same time. Conversely, the calculation of pf is independent from $score(pp)$. When we remove $score(pp)$, we just set $score(pp_i(p))$ to 1 in Eq. (8).

Table 9 Feature analysis

Dataset	Feature name					
	fop	$score(pp)$	pf	lw	cwr	$idfd$
SemEval-2010	−6.73	−4.92	−12.56	−2.11	−4.67	−13.98
VarTypes						
Book chapter	−1.56	−2.79	−18.86	−1.55	+1.21	−13.09
Conference paper	−8.69	−5.60	−12.78	−2.48	−6.44	−15.64
Journal article	−5.77	−4.71	−13.16	−2.17	−4.32	−11.45
Technical report	−3.69	−3.91	−10.03	−3.70	−3.37	−10.63
Scientific web page	+2.96	−3.29	−9.17	−4.12	−3.04	−14.31

% of change in precision when a single feature is removed. Negative value for a feature indicates that precision has decreased after feature removal and vice versa

the document. Documents in which title and abstract contain most of the keyphrases, e.g. conference papers and journal papers, benefit from this feature. However, for book chapters and scientific webpages that do not always have an abstract, this feature contributes little or even does more harm than good. Such features with negative impact are removed for the particular type of documents in the experiments in Sect. 5.6.

5.5 Evaluation of overlap elimination

To overcome the problem of overlap, we have proposed a new feature called $idfd$. The experiments in this subsection are designed to test the usefulness of this new feature.

We first gather statistics to study the frequency of overlap for pre-assigned keyphrases. We find that for the dataset SemEval-2010, there are only 2 overlap groups[14] among the 388 author-assigned keyphrases, and 31 overlap groups among the 1217 reader-assigned keyphrases. For the dataset VarTypes, only 23 overlap groups exist within a total 4,426 author-assigned keyphrases. The study shows that the overlap phenomenon is very rare in a human-assigned keyphrase list: the proportion is smaller than 1 % for the author-assigned set, and even for the reader-assigned set that has more keyphrases for each document, the proportion is not higher than 3 %.

Then, to study the overlap in candidate sets, we perform an experiment based on a subset of the two datasets. The subset contains 200 randomly selected documents. For each document, top-n candidate phrases obtained from the ranked candidate list are used for statistical calculation. We repeat the experiment when n has different values. We want to discover (i) if overlap is a frequent phenomenon in the candidate set. (ii) Among those overlap groups, how many of them contain author-assigned keyphrases? And (iii) Using the $idfd$ feature, can we correctly select the author-assigned keyphrase from a keyphrase-related overlap group?[15]

The results reported in Table 10 show that overlap is very common in a candidate set. More than half of the test documents have an overlap problem among the top-5 candidate phrases. When the number of extracted candidate phrases grows to 15, all the test documents have one or more overlap groups. Among such overlap groups, more than 30 % of them contain

[14] An overlap group is defined as a set of phrases in which any two elements p_1 and p_2 meet the following condition: p_1 is a sub-phrase of p_2, or p_2 is a sub-phrase of p_1.

[15] A overlap group is keyphrase-related when it contains an author-assigned keyphrase. And we consider the author-assigned keyphrase to be the most representative one in a keyphrase-related overlap group.

Table 10 Evaluation of overlap elimination

Top-n candidate phrase	Number of document with overlap problem	Number of overlap group	Number of keyphrase-related overlap group	Number of correctly extracted keyphrase	Precision
5	109	120	49	35	0.714
10	192	354	150	109	0.727
15	200	481	189	132	0.698
20	200	586	232	158	0.681

an author-assigned keyphrase. The $idfd$ feature can help us to select the author-assigned keyphrase from a keyphrase-related overlap group with an average precision of 70%. This experiment proves the necessity of overlap elimination and the effectiveness of our $idfd$ feature for selecting the most representative phrases.

5.6 Comparison of overall performance

To evaluate the overall performance of our system, experiments are conducted on two datasets.

For the dataset SemEval-2010, there was a competition of keyphrase extraction from scientific articles organized by SemEval. The participants were asked to produce the top 15 keyphrases for each article in the test document set. The evaluation was carried out by matching the extracted keyphrase sets against the answer sets (the reader-assigned set and the combined set). Three metrics are used: precision, recall, and F-score. The results of all the 19 participants are reported in [17] and ranked by F-score. We test our system in the same way and compare our results with top-ranked participants in Table 11. Our system ranks second for both answer sets. Observing the competing systems, we find that the majority of them adopt or partly adopt supervised learning approaches. Especially, among the top-ranked systems we listed in Table 11, only KP-Miner [5] and KX_FBK [28] are based on unsupervised approaches. Other systems use a number of different supervised learners. For example, SZTERGAK [2] and WINGUNS [26] use the classic Naïve Bayes model; HUMB [20] combines three models (Decision tree (C4.5), Multi-Layer perceptron and Support Vector Machine) with boosting and bagging techniques. SEERLAB [33] uses another supervised ensemble classifier called Random Forest. Looking at the results, on the one hand, it is encouraging to find that our approach outperforms most of the supervised systems. On the other hand, we notice the effect of training data on supervised systems. The highest ranked HUMB system exceeds in both answer sets for all the three measures. Especially, for the combined set, it yielded a 5.8% improvement than our system in F-Score. As reported in the publication [20], HUMB not only uses the training set of SemEval-2010 but also an additional "National University of Singapore (NUS) corpus" with 156 ACM articles from all computing domains. Adding the additional NUS training data helps to improve the final results. An in-depth discussion about this point is given later in Sect. 6.

For the dataset VarTypes, extracted results are compared with author-assigned keyphrases. Three metrics are used: precision, recall, and the average number of correctly extracted keyphrases per document. We compare the results from our system with three representative existing systems: the KEA system [45] designed by Witten et al. [8,40], the Extractor system [36] developed based on Turney's work [34,35], and the KP-Miner system [46] designed by El-Beltagy and Rafea [5]. Some details of the methodology of the three systems have been

Table 11 Performance comparison of our system and top-ranked participants when extracting 15 keyphrases

Systems	Combined keyphrases			Systems	Reader-assigned keyphrases		
	Precision	Recall	F-Score		Precision	Recall	F-Score
HUMB	0.272	0.278	0.275	HUMB	0.212	0.264	0.235
Our System	**0.262**	**0.268**	**0.260**	Our System	**0.205**	**0.255**	**0.227**
WINGNUS	0.249	0.255	0.252	KX_FBK	0.203	0.253	0.226
KP-Miner	0.249	0.255	0.252	SZTERGAK	0.199	0.248	0.211
SZTERGAK	0.248	0.254	0.251	WINGNUS	0.198	0.247	0.220
ICL	0.246	0.252	0.249	ICL	0.195	0.243	0.216
SEERLAB	0.241	0.246	0.243	SEERLAB	0.193	0.241	0.215
KX_FBK	0.236	0.242	0.239	KP-Miner	0.193	0.241	0.215

given in the Sect. 2. All the three systems are downloadable or available online so that the experiments have good repeatability. KP-Miner is a unsupervised system, while Extractor and KEA employ supervised methods. Thus, we can test whether, without applying any training step, the performance of our algorithm is comparable to both supervised and unsupervised extractors.

The available version of the Extractor system uses a fix training set which include 55 journal articles. More details about this dataset are described in Turney's paper [34]. A corpus collected from the same journal source[16] is used as the training set for the KEA system. The test of all the four systems are carried out on VarTypes.[17]

Since the available Extractor system is a demo one that only allows the extraction of up to seven keyphrases, a comparison among all four systems was only possible when seven keyphrases are extracted. The result are reported in Table 12 that gives detailed data for each type of document. When the extracting number of keyphrases increases, comparisons can only be carried out among our system, KP-Miner, and KEA. Table 13 shows the result when the number of extracted keyphrase are 10, 15, and 20.

The results demonstrate that the accuracy of our system compares favorably with other systems. Specifically, when extracting 7 keyphrases, our system returns 5 % more correct keyphrases on average than KP-Miner, 13 % more than KEA, and 18 % more than Extractor. When extracting 10 keyphrases, our system returns 7 % more correct keyphrases on average than KP-Miner and 10 % more than KEA. We notice that when the extracted keyphrases increase to 15 and 20, the improvement of system performance is slight. A possible reason is that when the number of extracted keyphrases is far bigger than the number of keyphrases in answer set, the effect of overlap attenuate. In this experiment, the average number of author-assigned keyphrases is about six. We observe that when 15 or 20 keyphrases are extracted, even if the output lists of other systems may contain several overlap phrases, it is quite possible that the six required answers are included in the final result. We may conclude that our system has the potential to perform better for applications in which a large number of keyphrases are required, such as text summarization. The results of the experiment on

[16] Because we do not have access to Turney's training set, we approximately duplicate this corpus to train the KEY system. We collect the same number of articles from the same journal source (6 from the *Journal of the International Academy of Hospitality Research*, 2 from *The Neuroscientist*, 14 from the *Journal of Computer-Aided Molecular Design*, and 33 from *Behavioral and Brain Sciences*) [8].

[17] We make sure that the training set and the test set do not have overlapping documents.

Table 12 Overall performance comparison among four systems when 7 keyphrases are extracted

Different metrics	Document type					
	Book chapter	Conference paper	Journal paper	Technical report	Scientific webpage	Total documents
Average precision						
Our system	**0.244**	**0.221**	**0.258**	**0.202**	**0.192**	**0.238**
KP-Miner	0.223	0.212	0.249	0.198	0.184	0.227
Extractor	0.172	0.199	0.214	0.201	0.193	0.202
KEA	0.203	0.197	0.225	0.195	0.192	0.210
Average recall						
Our system	**0.287**	**0.286**	**0.288**	**0.296**	**0.320**	**0.290**
KP-Miner	0.262	0.272	0.275	0.291	0.303	0.275
Extractor	0.203	0.260	0.235	0.294	0.325	0.247
KEA	0.236	0.257	0.248	0.286	0.319	0.256
Average match						
Our system	**1.71**	**1.55**	**1.81**	**1.41**	**1.34**	**1.67**
KP-Miner	1.56	1.48	1.74	1.39	1.29	1.59
Extractor	1.20	1.39	1.50	1.41	1.35	1.41
KEA	1.42	1.38	1.58	1.37	1.34	1.47

Table 13 Comparison of the overall performance among three systems

Number of extracted keyphrase	Average precision			Average recall			Average match		
	Our system	KP-Miner	KEA	Our system	KP-Miner	KEA	Our system	KP-Miner	KEA
10	**0.197**	0.184	0.179	**0.341**	0.317	0.312	**1.97**	1.84	1.79
15	**0.169**	0.165	0.160	**0.414**	0.392	0.379	**2.53**	2.48	2.40
20	**0.142**	0.138	0.136	**0.440**	0.426	0.419	**2.84**	2.76	2.72

SemEval-2010 substantiate this argument. The reader-assigned answer set and the combined set contain 12.04 and 14.66 keyphrases, respectively. When 15 keyphrases are extracted, the improvement from our system is obvious and better than KP-miner and other competing systems.

The run-time efficiency is also evaluated. To compare the efficiency of different systems, each was used to extract keyphrases from the same dataset, using the same machine and the same running conditions, and the extraction time was logged. Two experiments are performed for this purpose.

The first experiment is based on the whole VarTypes dataset (765 documents in total) and seven keyphrases are extracted. As far as we know, Extractor and KP-Miner do not provide downloadable open source implementations. Therefore, to make sure that the comparison is carried out in the same running conditions, we followed the detailed descriptions in the publications [5,34] to implement these two systems. Although our implementation cannot be exactly the same as the original systems, there should not be a substantial difference in

run-time efficiency.[18] We log the run-time of the whole process of keyphrase extraction for the comparison. The average results showed that the developed system completed the keyphrase extraction process in 0.25 the time taken up by KEA, in 0.34 the time taken up by Extractor and in 0.42 the time required by KP-Miner. In other words, our system is 4 times faster than KEA, 2.9 times faster than Extractor, and 2.3 times faster than KP-Miner.

The second experiment aims to further prove that time savings of our system improve as the document size increases. We perform this experiment using the subset of VarTypes that contains all the documents with a size larger than 10,000 words (163 documents in total) and repeat the evaluation described in the first experiment. The result for the picked-up long document set is that our system is 5.1 times faster than KEA, 3.7 times faster than Extractor, and 3.2 times faster than KP-Miner.

Finally, a case study is made to compare some sample outputs from four systems. Three different types of documents are taken as examples. Table 14 shows that our system has comparable or even superior performance. From the examples, we can also observe some additional advantages of our system:

(i) The special treatment of abbreviations is rewarding. Statistics were done on 100 abbreviation cases randomly chosen from our datasets. 68 % of the abbreviations are among the top-20 of the ranked candidate list. And 27 % of them appear in the author-assigned keyphrase list. The result shows that if an abbreviation is used, it is likely to have high importance in the document. The keyphrase extractors usually extract the abbreviated form. However, many authors tend to use the complete form as a keyphrase because it is more understandable for users. For example, in the first document, for the author-assigned keyphrase "ventral tegmental area," our system can give the original version, whereas the other systems only extract the abbreviation "VTA".

(ii) The elimination of overlap benefits those important candidates with not very high feature weights. For example, in the third document, before we use the $idfd$ feature, the candidate list contained many overlap phrases, such as "fitness" and "fitness landscape", "multi-objective," and "multi-objective optimization". The overlap elimination removes such redundant phrases so that some room is left for subsequent candidates in the ranking list. The candidate "dual phase evolution" stands out relying on this operation.

Inevitably, during the experiment, we also observed some cases for which our system does not perform as well as the other systems. We collected a set of document with poor outputs. Observations indicate that our system tends to have relatively poor performance especially when the majority of the author-assigned keyphrase list is composed of single words. The reason may be related to the lw and cwr feature we use. Both of the two features encourage those compound terms.[19] Analysis for some typical examples manifests that compound terms may have higher rank than their elemental single term even if the frequency of the single terms is three times that of the compound terms. A further research about the proportion of single and compound keyphrases may help us to make a better use of these two features.

[18] El-Beltagy and Rafea [5] also reported similar comparison among KEA, Extractor and KP-Miner, which showed consistent results with our experiment. Therefore, we believe that our reported data of run-time efficiency is reliable.

[19] Although cwr is used to encourage phrases containing high-rank core words, high-rank compound candidate phrases benefit greatly from this feature because they are always mainly composed by high-rank core words. In other words, high-rank compound candidates tend to have high value for both lw and cwr features.

Table 14 Top 7 keyphrases extracted by 4 systems from three different document *(The matches are presented in boldface type)*

Document Source: http://cogprints.org/6385/
Document Title : Ingestion of amniotic fluid enhances the facilitative effect of VTA morphine on the onset of maternal behavior in virgin rats)
Document Type : Journal article
Author-assigned keyphrases: Amniotic fluid, Interpeduncular nucleus, Maternal behavior, Opioid, POEF, Ventral tegmental area, Morphine, Rat

Our system *(Six matches)*	**KP-Miner** *(Four matches)*	**Extractor** *(Four matches)*	**KEA** *(Four matches)*
1. **amniotic fluid**	1. **maternal behavior**	1. **maternal behavior**	1. **maternal behavior**
2. **maternal behavior**	2. **amniotic fluid**	2. **morphine**	2. **morphine**
3. **morphine**	3. vta morphine	3. **rats**	3. dose
4. **rats**	4. onset	4. onset	4. onset
5. **ventral tegmental area**	5. **rats**	5. VTA	5. **rats**
6. **opioid**	6. **morphine**	6. **amniotic fluid**	6. **amniotic fluid**
7. infusion	7. ingestion	7. morphine injection	7. ingestion

Document Source: http://cogprints.org/4533/
Document Title : Evolution and Mirror Neurons: An Introduction to the nature of Self-Consciousness
Document Type : Conference paper
Author-assigned keyphrases: self-awareness, self-consciousness, evolution, mirror neuron, non human primate, intersubjectivity

Our system *(Four matches)*	**KP-Miner** *(Four matches)*	**Extractor** *(Three matches)*	**KEA** *(Three matches)*
1. **mirror neuron**	1. **mirror neuron**	1. **evolution**	1. **mirror neurons**
2. **evolution**	2. **self-consciousnes**	2. body self-awareness	2. **evolution**
3. **self-consciousness**	3. **non human primate**	3. **self-consciousness**	3. body self-awareness
4. human primate	4. body self-awareness	4. **mirror neurons**	4. **self-consciousness**
5. anxiety limitation	5. evolutionary	5. anxiety limitation	5. anxiety
6. **self-awareness**	6. anxiety limitation	6. consciousness	6. group life
7. group life	7. **intersubjectivity**	7. humans	7. basic life

Document Source: http://cogprints.org/6573/
Document Title : Evidence of coevolution in multi-objective evolutionary algorithms
Document Type : Book chapter
Author-assigned keyphrases: coevolution, dual phase evolution, evolutionary algorithms, multi-objective optimization, self-organized criticality

Our system *(Four matches)*	**KP-Miner** *(Two matches)*	**Extractor** *(Three matches)*	**KEA** *(Three matches)*
1. **coevolution**	1. **evolutionary algorithms**	1. fitness	1. **evolutionary algorithms**
2. **evolutionary algorithms**	2. **multi-objective optimization**	2. species	2. fitness
3. fitness landscape	3. species	3. evolution	3. **coevolution**
4. **multi-objective optimization**	4. takes place	4. **coevolution**	4. take place

Table 14 continued

Document Source: http://cogprints.org/6573/

Document Title : Evidence of coevolution in multi-objective evolutionary algorithms

Document Type : Book chapter

Author-assigned keyphrases: coevolution, dual phase evolution, evolutionary algorithms, multi-objective optimization, self-organized criticality

Our system *(Four matches)*	**KP-Miner** *(Two matches)*	**Extractor** *(Three matches)*	**KEA** *(Three matches)*
5. species	5. coevolutionary behavior	5. **multi-objective optimization**	5. **multi-objective optimization**
6. multi-objective environment	6. multi-objective environment	6. **evolutionary algorithms**	6. evolution
7. **dual phase evolution**	7. fitness	7. coevolutionary behavior	7. species

6 Discussion of some open issues

In this section, we want to discuss some open issues we encountered during the system design and evaluation.

(i) Supervised approaches versus unsupervised approaches: At the beginning of system design, we intend to build an unsupervised system for keyphrase extraction. But it is a pity that we are not able to find a fundamental reasoning for choosing the parameters. Because of the manually-done learning process of parameter setting, our system should be categorized into supervised systems. However, since the obtained parameters are robust enough, in practical application, we can consider them as fixed parameters that do not require for any training data or learning phase. In this sense, our system has many characteristics similar to those unsupervised systems. Although our system does not apply traditional machine learning method, we are impressed by the strength of supervised learning sometimes during the experiments. We noticed that although our system exceeds the competing supervised system in most cases of the evaluation, there exists exceptions: when the HUMB system uses a 300-document training set, it shows a better performance than our system. The reason is that SemEval-2010 can be considered as a domain-specific dataset in computer science, and the accuracy of supervised systems increases given an appropriate training set with enough samples. Nowadays, digital libraries such as ACM have built a widely-used classification system to index the articles and are able to provide documents in the same domain for training. Therefore, for the domain-focused applications without high-efficiency requirement, we can expect good performance from supervised approaches. However, the domain-specific training data are not always easy to access, because not every domain has an authoritative classification system such as the ACM Classification System for computer science. In comparison, unsupervised approaches are more flexible. They are based on unlabeled corpora and do not exploit any manually tagged corpus [19,24]. In their purest version, they do not make use of any machine-readable resources like dictionaries, thesauri, etc. This main advantage yields high benefits when it is hard to prepare a training set that is oriented enough for supervised learning, for instance, when the test dataset contains documents of diverse topics, or the domain of the test documents is not available in advance. An example task is the on-line automatic text summarization. The user inputs a text such as a book chapter. The on-line system should output the

summarization immediately or at least highlight the important (or topical) sentences in the text. For such non domain-specific applications, as the system cannot be informed any subject-related information, it is more suitable to adopt unsupervised approaches. The experiments in this paper show that with a well-chosen feature set, unsupervised approaches can achieve pretty good accuracy that is comparable to or even better than supervised methods. In addition, without a training phase, unsupervised systems may be superior in time efficiency and are a good solution for some real-time applications.

(ii) In which contexts is run-time efficiency important? Our new algorithm for obtaining a refined candidate set helps to improve the time efficiency of a keyphrase extraction system, especially when the system consumes a lot of computing resources in feature engineering. However, run-time efficiency is not always a significant issue for a keyphrase extraction system, it depends on its application. For applications such as keyphrase search in digital libraries, keyphrase extraction can be processed in advance, so the users pay little attention to its efficiency. But there are also applications requiring high efficiency, for example, keyphrases for interactive query refinement [36]. The search engine takes the user's query, fetches the first round of documents, extracts keyphrases from them,[20] and then displays the first round of documents to the user, along with suggested refinements to the first query, based on combinations of the first query with the extracted keyphrases. In such a real-time application, especially when the application is based on a large searching scale (e.g., the whole web), a fast keyphrase extraction technique is expected, because the time savings in this phase directly reduces the search response time.

(iii) IDF and the end-use of the keyphrases: As we have stated at the very beginning of the paper, the technique of keyphrase extraction has many important applications. All the applications can be divided into two types, according to the end-use of the keyphrases: one is to help indicate what a document is about (i.e., text indexing and summarization), which is more useful in variable-domain collections, and the other is to help discriminate between documents (i.e., text classification and clustering), which is more useful in domain-specific collections. In this paper, we try to develop a robust keyphrase extraction system rather than a system devoted to a specific task. However, we notice that, for different end-use of the keyphrases, the use of IDF in the proposed method may have different effects on the extraction result.

IDF has been mentioned twice during the system design. The first time is in Sect. 3.2, for defining the core word set. We decide not to use IDF for core word selection, which ensures that no important word will be omitted because of its low IDF value especially in domain-specific document sets. This choice pays more attention to the completeness of the keyphrase list that is significant for the end-use of content indication. Differently, the end-use of discrimination calls for keyphrases not only important but also unique for the document. For instance, if we want to distinguish two documents about machine learning, the keyphrase "machine learning" makes little contribution to this task. Therefore, for the purpose of discrimination, we may consider including IDF for core word selection, which helps removing such common topic related phrases.

The second time we mentioned IDF was for overlap elimination. For the same overlap group, the $idfd$ feature may generate different results given different collections. For example, an overlap group contains two candidates 'supervised' and 'supervised algorithm'. In a variable-domain collection, the IDF of 'supervised' is higher than

[20] It is possible that during the keyphrase extraction, some information from the user's first query are used by the system, to obtain keyphrases more oriented to the query.

the IDF of 'algorithm', so $idfd$ tends to retain 'supervised'. On the contrary, given a collection in machine learning domain, we cannot expect a high IDF value for the word 'supervised', and $idfd$ may tend to select 'supervised algorithm'. Although the difference effects of IDF exist, at present, we do not observe that the difference do harm for either the end-use of content identification or discrimination. Furthermore, we are able to avoid the difference by calculating the IDF value from a prepared corpus such as a given digital library.

(iv) Generality of the features and robustness of the parameters: when we choose features for calculating the weight of candidate phrases, we take into account some linguistic knowledge (i.e., the frequency of compound terms requires for a boosting factor) and some characteristics of scientific documents (i.e., different logical sections contribute keyphrases at different rates). On the one hand, all these knowledge and characteristics are domain-independent and thus the selected features can be used as a baseline feature set for the task of this paper. Such a feature set is easy to be combined with some domain-related features or resources when used for domain-specific corpus. On the other hand, the experiments in Sect. 5.4 explore that feature contributions for the system accuracy may vary for different types of documents. This can be a potential drawback of the selected feature set when used for type-specific corpus. The above analysis is made in terms of features. Similarly, with regard to parameters, we know that the two parameters α and β are used to control the effect of the features in the final result. Thus, we may conclude that they are not sensitive to the domain of test documents, but they may be sensitive to the types of test documents. For example, according to the results reported in Table 9, when applied for a set of scientific web pages, we can expect that an appropriate increase of the value of α may reduce the effect of the feature fop and hence help to improve the system accuracy. That is, although the current parameter setting works well for general applications, there may exist better parameter values when the test set is type-specific. To look for possible solutions of this problem, we think of building models of parameter learning by resorting to statistical studies of different types of documents, which may lead to more corpus-adapted parameter settings and improved system performance.

7 Conclusions and future work

In this paper, we presented an effective technique that can automatically produce a keyphrase list for a given scientific document.

One of the main issues in keyphrase extraction is the difficulty of identifying the candidate phrases. Traditional n-gram methods do the same treatment for all the positions in the target document, which usually results in a large candidate set. To be more efficient, we first introduce a core word set that helps pointing out those more competitive positions for producing potential keyphrases. Then a core word expansion algorithm is proposed to generate candidate phrases from the remarked positions. The algorithm is based on core word expansion trees, which ensures that each occurrence of a core word generates at most two candidates. Compared with the traditional one, our current candidate set reduces the size by 75 % on average while keeping almost the same covering ability. This refined candidate set greatly contributes to run time savings for the whole system.

Next, during the step of feature engineering, three groups of features are used for choosing the most important candidate phrases. Some new features are introduced, especially a feature aiming at solving the overlap problem. This feature focuses on the different parts between

two overlap phrases to see if the longer one can help the readers gain more useful information than the root one. The elimination of redundancy substantially increases the information content of the final keyphrase list.

The empirical results of this paper are encouraging. The new algorithm for candidate phrase generation obviously improved the system efficiency. The new added features also proved to be useful in candidate filtering and final keyphrase selection. Our system attains a superior performance to both supervised and unsupervised reference systems.

In addition to better system accuracy, the main advantage over conventional supervised systems is that the presented method does not require for any human-labeled training sets or domain specific thesaurus. This makes the system more flexible and easier to use. The main advantages over unsupervised systems are the dramatic time savings brought about by the refined candidate set and the potential ability of producing a large high-quality keyphrase list. This makes the system more efficient to adapt to real-time tasks and shows promises for the applicability of the technique to more far-reaching tasks.

In future work, we want to take some user-related information into account, to further facilitate users' fast browsing and intensive reading. The technique described in this paper is used to extract topic-related information from user history documents. Some methods [6,7] are then applied to analyze user behaviors and obtain user interest. The results will be combined to construct the user profile, which is an important component for developing a personalized research paper search system.

References

1. Barker K, Cornacchia N (2000) Using noun phrase heads to extract document keyphrases. In: Proceedings of the 13th Biennial conference of the Canadian society on computational studies of intelligence: advances in artificial intelligence, pp 40–52
2. Berend G, Farkas R (2010) SZTERGAK: feature engineering for keyphrase extraction. In: Proceeding of the 5th international workshop on semantic evaluation, ACL, Uppsala, Sweden, pp 186–189
3. Brill E (1995) Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. Comput. Linguistics 21(4):543–566
4. Chen M, Sun JT, Zeng HJ, Lam KY (2005) A practical system of keyphrase extraction for web pages. In: Proceedings of ACM 14th conference on information and knowledge management, Bremen, Germany, pp 277–278
5. El-Beltagy SR, Rafea A (2009) KP-Miner: a keyphrase extraction system for english and arabic documents. Inf Syst 34(1):132–144
6. Enembreck F, Barthès J-P (2007) Multi-agent based internet search. Int J Prod Lifecycle Manage 2(2):135–156
7. Enembreck F, Barthès J-P, Avila BC (2004) Personalizing information retrieval with multi-agent systems. Lecture notes in computer science, vol 3191. Springer, Berlin, pp 71–91
8. Frank E, Paynter GW, Witten IH, et al (1999) Domain-specific keyphrase extraction. In: Proceedings of the 16th international joint conference on artificial intelligence, Stockholm, Sweden, pp 668–673
9. Gong ZG, Liu Q (2008) Improving keyword based web image search with visual feature distribution and term expansion. Knowl Inf Syst 21(1):113–132
10. Huang C, Tian Y, Zhou Z et al (2006) Keyphrase extraction using semantic networks structure analysis. In: Proceedings of the 6th international conference on data mining, Hong Kong, China, pp 275–284
11. Hulth A (2003) Improved automatic keyword extraction given more linguistic knowledge. In: Proceedings of the 2003 conference on empirical methods in natural language processing, Sapporo, Japan, pp 216–223
12. Hulth A, Megyesi B (2006) A study on automatically extracted keywords in text categorization. In: Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the association for computational linguistics, Sydney, Australia, pp 124–154
13. Jacquemin C (1995) A symbolic and surgical acquisition of terms through variation. In: Proceedings of connectionist, statistical and symbolic approaches to learning for natural language processing, pp 425–438
14. Kelleher D, Luz S (2005) Automatic hypertext keyphrase detection. In: Proceedings of 22th international joint conference on artificial intelligence, Edinburgh, Scotland, pp 1608–1609

15. Kim SN, Ken M-Y (2009) Re-examining automatic keyphrase extraction approaches in scientific articles. In: Proceeding of 2009 workshop on multiword expressions: identification, interpretation, disambiguation, applications, Suntec, Singapore, pp 9–16
16. Kordoni V, Zhang Y (2010) Disambiguating compound nouns for a dynamic HPSG treebank of Wall Street Journal texts. In: Proceedings of the 7th international conference on language resources and evaluation, Valetta, Malta
17. Kim SN, Medelyan O, Kan M-Y et al (2010) SemEval-2010 task 5: automatic keyphrase extraction from scientific articles. In: Proceeding of the 5th international workshop on semantic evaluation, ACL, Uppsala, pp 21–26
18. Kumar N, Srinathan K (2008) Automatic keyphrase extraction from scientific documents using N-gram filtration technique. In: Proceedings of the 8th ACM symposium on document engineering, Sao Paulo, pp 199–208
19. Li T, Ogihara M (2005) Semi-supervised learning from different information sources. Knowl Inf Syst 7(3):289–309
20. Lopez P, Romary L (2010) HUMB: automatic key term extraction from scientific articles. In: Proceeding of the 5th international workshop on semantic evaluation, ACL, Uppsala, pp 248–251
21. Medelyan O, Witten IH (2006) Thesaurus based automatic keyphrase indexing. In: Proceedings of the 6th ACM/IEEE-CS joint conference on digital libraries, Chapel Hill, pp 296–297
22. Morin E, Jacquemin C (2004) Automatic acquisition and expansion of hypernym links. Comput Humanities 38(4):363–369
23. Nakov P, Hearst M (2005) Search engine statistics beyond the n-gram: application to noun compound bracketing. In: Proceedings of CoNLL-2005, 9th conference on computational natural language learning, Ann Arbor, pp 17–24
24. Navigli R (2009) Word sense disambiguation: a survey. ACM Comput Surv 41(2):article 10
25. Nguyen TD, Kan M-Y (2007) Keyphrase extraction in scientific publications. In: Proceeding of international conference on Asian digital libraries, Hanoi, pp 317–326
26. Nguyen TD, Luong M-T (2010) WINGNUS: keyphrase extraction utilizing document logical structure. In: Proceeding of the 5th international workshop on semantic evaluation, ACL, Uppsala, pp 166–169
27. Park J, Lee S-G (2011) Keyword search in relational databases. Knowl Inf Syst 26(2):175–193
28. Pianta E, Tonelli S (2010) KX: a flexible system for keyphrase extraction. In: Proceeding of the 5th international workshop on semantic evaluation, ACL, Uppsala, pp 170–173
29. Porter MF (1980) An algorithm for suffix stripping. Program 14(3):130–137
30. Rus V, Moldovan DI, Bolohan O (2002) Bracketing compound nouns for logic form derivation. In: Proceedings of the 15th international florida artificial intelligence research society conference, pp 198–202
31. Song M, Song IY, Allen RB et al (2006) Keyphrase extraction-based query expansion in digital libraries. In: Proceedings of the 6th ACM/IEEE-CS joint conference on digital libraries, Chapel Hill, pp 202-209
32. Srinivasan P (1996) Optimal document-indexing vocabulary for MEDLINE. Inf Process Manage 32(5):503–514
33. Treeratpituk P, Teregowda P, Huang J et al (2010) SEERLAB: a system for extracting keyphrases from scholarly documents. In: Proceeding of the 5th international workshop on semantic evaluation, ACL, Uppsala, pp 182–185
34. Turney PD (2000) Learning algorithms for keyphrase extraction. Inf Retrieval 2(4):303–336
35. Turney PD (2003) Coherent keyphrase extraction via web mining. In: Proceedings of the 20th international joint conference on artificial intelligence, Acapulco, pp 434–439
36. Turney PD (2005) Extractor, http://www.extractor.com
37. Wan XJ, Xiao JG (2008) Single document keyphrase extraction using neighborhood knowledge. In: Proceedings of the 23rd international conference on artificial intelligence, Chicago, pp 855–860
38. Wang CH, Zhang M, Ru LY et al (2008) An automatic online news topic keyphrase extraction system. In: Proceedings of 2008 IEEE/WIC/ACM international conference on web intelligence, Sydney, pp 214–219
39. Wei FR, Li WJ, Lu Q, He YX (2009) A document-sensitive graph model for multi-document summarization. Knowl Inf Syst 22(2):245–259
40. Witten IH, Paynter GW, Frankand E et al (1999) KEA: practical automatic keyphrase extraction. In: Proceedings of the 4th ACM conference on digital libraries, Berkeley, pp 254–255
41. You W, Fontaine D, Barthès J-P (2009) Automatic keyphrase extraction with a refined candidate set. In: Proceedings of the 2009 IEEE/WIC/ACM international conference on web intelligence, Milan, pp 576–579
42. Zhang CZ, Wang HL, Liu T et al (2008) Automatic keyword extraction from documents using conditional random fields. Comput Inf Syst 4(3):1169–1180

43. Zhang K, Xu H, Tang J et al (2006) Keyword extraction using support vector machine. In: Proceedings of the 7th international conference on web-age information management, Hong Kong, pp 86–96
44. Automatic Keyphrase Extraction from Scientific Articles. Task #5 of the 5th workshop on semantic evaluation, 2005. http://semeval2.fbk.eu/semeval2.php?location=tasks&taskid=6
45. KEA. http://www.nzdl.org./Kea/download.html
46. KP-Miner: A Simple System for Effective Keyphrase Extraction (New version Oct. 2007). http://www.claes.sci.eg/coe_wm/kpminer/
47. The SemEval-2010 dataset. http://semeval2.fbk.eu/semeval2.php?location=data
48. Stopword list. http://www.lextek.com/manuals/onix/stopwords1.html

Author Biographies

Wei You is a Ph.D. student in Computer Science at the University of Technology of Compiègne, France. She received a B.S. in Computer Science in 2004 and a M.S. in Computer Application Technology in 2007 from Xiamen University, China. Her research interests include Information Retrieval, Recommender System and Knowledge Management.

Dominique Fontaine is an Associate Professor, in the Department of Computer Science, University of Technology of Compiègne, France. As a member of the UMR CNRS 6599 Heudiasyc Laboratory, his research interests were previously in the areas of Knowledge Base Systems, Knowledge Acquisition, Temporal Reasoning, and more recently are in the areas of Pedagogical Systems, Knowledge Retrieval and Personalized Recommendation.

Jean-Paul Barthès obtained a Ph.D. from Stanford University in 1973. Currently, he is professor emeritus in the department of Computer Science at the University of Technology of Compiègne (UTC) in France. His main research interests are related to Distributed Artificial Intelligence and mixed societies of cognitive artificial and human agents with a focus on Personal Assistant Agents. Prof. Barths is co-chair of the IEEE-SMC Technical Committee on CSCWD (Computer-Supported Cooperative Work in Design). Under his supervision a number of multi-agent applications have been developed during the last 25 years.