Keyphrase Extraction using Semantic Networks Structure Analysis

Chong Huang\textsuperscript{1}, Yonghong Tian\textsuperscript{2}, Zhi Zhou\textsuperscript{2}, Charles X. Ling\textsuperscript{3}, Tiejun Huang\textsuperscript{1}

\textsuperscript{1} Graduate University, Chinese Academy of Sciences, Beijing 100039, China
\textsuperscript{2} Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China
\textsuperscript{3} Department of Computer Science, University of Western Ontario, N6A 5B7, Canada

\{chuang, yhtian, zhou, tjhuang\}@jdl.ac.cn
cling@csd.uwo.ca

Abstract

Keyphrases play a key role in text indexing, summarization and categorization. However, most of the existing keyphrase extraction approaches require human-labeled training sets. In this paper, we propose an automatic keyphrase extraction algorithm, which can be used in both supervised and unsupervised tasks. This algorithm treats each document as a semantic network. Structural dynamics of the network are used to extract keyphrases (key nodes) unsupervised. Experiments demonstrate the proposed algorithm averagely improves 50\% in effectiveness and 30\% in efficiency in unsupervised tasks and performs comparatively with supervised extractors. Moreover, by applying this algorithm to supervised tasks, we develop a classifier with an overall accuracy up to 80\%.

1. Introduction

As a short list of topical phrases or words, keyphrases briefly describe the contents of a document. They are widely used in text indexing, summarization, and categorization. Faced to roughly 200,000 English digital books in our digital library (DL) project\textsuperscript{1}, we are motivated to summarize documents and measure text similarity. But topical terms in a book’s metadata are too few to fulfill our task. Therefore, we plan to use extracted keyphrases.

Most existing keyphrase extraction algorithms\cite{1, 18} are based on supervised learning on papers or web pages. When applied to digital books, they have several drawbacks. First, a book has longer text but less structural tags, making the extraction more complicated. Second, it is laborious to set up an appropriate training set with sufficient samples, especially in DL. Unsupervised learning is thus more preferable. Last but most important, no single prevailing feature weight\cite{15, 16} is satisfactory for unsupervised keyphrase extraction. A feature weight, also called feature metric or term weighting scheme in text mining, measures a term quantitatively in a representative aspect, and indicates how “key” the term is to the document. Most prevailing feature weights solely rely on term frequency\cite{22} or specific structure\cite{17}. They are either inaccurate or incapable to generalize. As a result, the accuracy of unsupervised extraction using a single weight in diversified datasets is unsatisfactory. Additionally, among these feature weights, some are set-dependent (if a new document is added to the data set, set-dependent feature weights of all documents in the set need rescoring).

In this paper, we propose a keyphrase extraction method, using several novel set-independent feature weights, which can be used in both supervised and unsupervised tasks. This algorithm treats each document as a semantic network that holds syntactic relation in edges and frequency information in nodes. By analyzing the structure of these networks, we notice that they are complex but compact – named Small-World Phenomenon (SWP)\cite{11} – and key nodes make them compact. We use several structural variables of SWP as feature weights to find out these key nodes. Given that the structure of a network represents the syntactic structure of its source document, we select some key nodes as keyphrases. This process requires no training data. Experiments demonstrate the proposed keyphrase extraction algorithm averagely improves 50\% in effectiveness (with up to 1/4 rank and two times of weight) and 30\% in efficiency in unsupervised tasks and performs comparatively with supervised extractors. Moreover, we apply this algorithm to a supervised task. Experiments show that the overall accuracy of the acquired classifier can be up to 80\% and it can automatically determine the number of keyphrases.

Section 2 surveys recent literatures on keyphrase extraction and networks structure analysis. Section 3 outlines our knowledge organization system. Section 4 presents our keyphrase extraction algorithm in both supervised and unsupervised tasks. Section 5 describes experiments on the effectiveness and efficiency of it.

2. Background

2.1 Keyphrase extraction

Keyphrase extraction can be viewed as a supervised or unsupervised learning task. Most of published literatures cover only supervised tasks. Two methods are developed. Intuitively, it is a two-category classification task -- a
networks [11]. Network structure analysis is applied to analyze the global structure of complex networks [11]. Network structure analysis is applied to find key nodes.

As far as we know, we are the first to use collective dynamics to extract keyphrases in English documents. We define three variables to measure a term’s impact on the global structure of the semantic network in three aspects: connectedness, compactness, and the combination of connectedness and compactness, called Connectedness Centrality, Betweenness Centrality, and Relation Centrality respectively. Other authors use traffic [10], clustering coefficient [2], and betweenness for structure analysis. Among them, two most related works are: Zhu et al [4] use $\Delta_L$, to extract keywords of Chinese news web pages, in a relatively small dataset without theoretical reasoning (we have a different definition of $\Delta_L$), y; Fortunato et al [21] use a similar approach with us (they call it information centrality), to analyze edges rather than nodes to discover community structures. However, most of these literatures don’t mention many practical issues, such as unconnectedness and computational complexity.

3. Constructing semantic networks

Figure 1. A sentence and its semantic Network. The slashed words are stopwords to filter.

Given a document, the performance of extraction depends on the richness of reserved information. A knowledge organization system (KOS) is needed to store information. Because of its power to hold different kinds of relationship between lexical units, we choose semantic networks as our KOS. Rather than simply as isolated points or linear sequence, we view the original text as a semantic network — a term (a word or a phrase) as a node and a relation between two terms as an edge. Frequency information is stored in nodes, and other ordinal or de-

2.2 Semantic networks structure analysis

Researchers have recently developed a system of theory on how to analyze the global structure of complex networks [11]. Network structure analysis is applied to different kinds of semantic network in natural language processing, such as lexical pattern analysis [5], ontology of language, and language evolvement [6, 13].

In network structure analysis, SWP is recognized as a key property of networks with a large number of nodes. Watts and Strogatz [8] present two important variables and the way to understand the collective dynamics of SWP. SWP provides several feature weights for structural analysis to find key nodes.

A number of feature weights are developed in text mining. They can be divided into two categories: weights that can be used only in supervised learning and weights that can be used in both supervised and unsupervised learning. The first class of them rely on prors or statistics acquired in training, such as Accuracy, F1-Measure, Information Gain, Mutual Information, Term Strength, Chi-Square, and Odds Ratio [15, 16].

The second class of features can be extracted without training. Most of them rely on structural rules or frequency. Firstly, some weights set up rules on specific structure [17], but their performances fluctuate greatly in different datasets. Methods based on frequency can be Term Frequency (TF), TFIDF [18], or more delicately Okapi’s BM25 [22]. Though widely used, they have several drawbacks. First, they are not accurate enough for unsupervised keyphrase extraction. This is why KEA and Extractor do not extract keyphrases solely on TFIDF or TF. Second, ordinal information and dependency between phrases are lost. Finally, some of them need parameter tuning (as in BM25). To overcome these problems, some authors analyze co-occurrence of words within a fixed distance. Mihalcea and Tarau [12] use a PageRank-like manner to analyze a so-called recommending graph. We store ordinal information in a similar way, but view a relation as a path for information flow.

Phrase identification is a main issue when the linguistic unit is phrase rather than word. Intuitively, all word sequences are candidate phrases and keyphrases are selected from them. In paper [1] and [18], some hypotheses are set up to filter meaningless word sequences, but they the number of remaining candidates is still large.

In a word, our main contributions are: (1) proposing a fast and effective phrase identification algorithm; (2) using effective feature weights such as structural variables of Small-World Phenomenon (SWP).
pendency information is stored in the structure of the network, which we will utilize in the extraction algorithm in the form of feature weights.

3.1 Phrase identification

A phrase is a consecutive sequence of words without intervening punctuations, forming a grammatical constituent of a sentence. A node in the semantic network is a term, but how to distinguish a phrase from meaningless word sequence is an issue. KEA and Extractor filter meaningless word sequences by some rules before they are added into the network, but these rules so weak that most of the identified phrases are still semantically meaningless. Redundancy, noise, and compositional explosion are haunted threats. In graphs, they impair the overall efficiency and precision of extraction, since structural variables are sensitive to redundancy and noise.

After investigating the characteristics of topical terms in our eBook archive, we found a two-phase filtering algorithm more effective.

First, phrases must fulfill three rules:
1. A phrase can’t start or end up with stopwords (similar word list as KEA).
2. In a phrase, only a word sequence of less than four midwords (propositions, nouns, numbers, and some conjunctions in stopword list) can exist between two non-stopwords. Phrases as “wheat and rice” are included, while they are not in Extractor.
3. Frequency of a phrase (PF) is above a minimum value.

Second, we select phrases with relatively higher PFs. They are first divided into societies by the number of non-stopwords. Then phrases with the same word in a society are further assigned to the same group. A phrase is then in \( n \) groups if it has \( n \) distinct non-stopwords. Every group has only one or none winner. A winning keyphrase candidate (named a giant phrase) should have a top PPF (Percent of PF) and a PTF (Percent of TF) above a threshold in all \( n \) groups it belongs to. Finally, only winners can remain. Take phrase \( i \) in group \( k \) (phrase \( i \) consists of words \( d \)) for example,

\[
PTF(phrase_i, group_k) = \frac{PF_i}{TF_i},
\]

\[
PPF(phrase_i, group_k) = \frac{PF_i}{\sum_{phrase_j, group_k} PF_j},
\]

where \( PTF(phrase_i, group_k) \) depicts how often word \( d \) concurs with phrase \( i \), and \( PPF(phrase_i, group_k) \) depicts this comparing other phrases in group \( k \). Note that a single word is not a phrase. Therefore, the TF \( i \) is larger than the sum of all PF in group \( k \).

Caropreso et al [20] report that duplicated information among uni-gram and bi-gram is detrimental to effectiveness. Therefore, we replace a single word with its giant phrase if it is available, on the assumption that the giant phrase is a proper substitute for its words in the context.

As for the phrase length, Mladenic and Grobelnik [19] find that word sequences of length up to three improve the performance of feature weight. Extractor and KEA also support keyphrases consisting of less than four words. Observing 85% topical terms of eBooks are less than two words and the remainder are mostly made up by two key-phrases, we now support keyphrases with two words, and this algorithm is extensive to a longer length.

3.2 Relationship establishment

There are mainly three kinds of relationship between terms: semantic relation, syntactic relation, and co-occurrence. Semantic relation establishment requires a dictionary such as WordNet and disambiguation. Syntactic relation builds networks on the grounds that terms at a certain distance have syntactic or semantic relationship [5, 6]. It needs supervised learning including Part-of-speech (POS) tagging. Co-occurrence presumes concurring within a linguistic unit is syntactically or semantically indicative [4, 7], but it introduce redundant and noises. In addition, co-occurrence networks have most edges and thus a highest computational complexity.

We aim to combine the last two kinds, without a training set and a relative low computational complexity. Lyon et al [6] study that 70% of syntactic dependencies are between neighboring terms, and 17% at a distance of 2. Meanwhile, Cancho et al [5] conclude that all syntactical relation within distance 2. Interestingly, we find that after filtering stopwords, nearly all syntactical relation at a distance of 2 is shortened to 1. Because of this, we consider neighboring in the same sentence as the relationship between nodes. Experiment on the performance of neighboring relation in several datasets (Section 5.4) validate that it is comparative to relations with a larger co-occurrence window, and its structural and computational complexity is lowest.

This neighboring relationship holds two properties: unweighted and undirected. There are mainly two ways to calculate the weight of an edge. First, use POS tags to distinguish kinds of syntactic relation, and assign different weights to them. But training sets with syntactic labels are needed. Alternatively, given a training set, n-gram can calculate probabilistic distribution of the co-occurrence between terms within a distance \( n \), which can be naturally used as weights. One problem of weighted graph is that the computation of structural variables is more time-consuming. Therefore, currently we support unweighted graph. Since neighboring relation does not identify kinds of syntactic relation, the graph is undirected. For example, “eat fish” is a verb-object relation, and “fish eat” is a subject-verb relation, but they are the same in the neighboring relation, since the only concern is the existence rather than what kind of syntactic relation they are.
4. Extracting keyphrases using networks structure analysis

Granted that the structure of a network represents the structure of its source document and the edges represent syntactic relation, to extract a keyphrase is to find a key node in the network. A key node is an important node in the structure of the network.

Unsupervised keyphrase extraction algorithm takes two steps. First, it builds an accurate feature weight that can depict the importance of a node in the network without training. Second, by ordering phrases by this weight, it selects top n phrases as keyphrases. The n is decided manually by the demand of the extraction.

4.1 Network structure analysis

To find key nodes in a network, we must first decide what characteristic of network structure to concentrate on. In traditional graph theory, connectedness is an important issue to study the structure of networks. Connectedness measures how connected a graph is: whether a graph is connected or disconnected into components; whether a graph is still has small average minimum path length structure. Though a network has a large amount of nodes, its neighboring structure, and global ones with the status of the entire network. For instance, local variables are degree, TF, PF, clustering coefficient and the like. Since there are thousands or even millions of terms in one book, the structure of the corresponding network is rather complex with up to 60,000 nodes and 400,000 edges. Table 1 summarizes from 9877 semantic networks, randomly sampled from our eBook archive. Concluded from it, SWP happens in our semantic networks.

Table 1. Statistics of SWP variables. (m - number of edges, n - number of nodes, CC - number of components, EX - expectation of variable X, DX - variance of X)

<table>
<thead>
<tr>
<th>X</th>
<th>L</th>
<th>C</th>
<th>n</th>
<th>m</th>
<th>m/n²</th>
<th>CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>EX</td>
<td>4.209</td>
<td>0.629</td>
<td>4295.1</td>
<td>27690.2</td>
<td>0.0027</td>
<td>1.207</td>
</tr>
<tr>
<td>DX</td>
<td>0.826</td>
<td>0.045</td>
<td>3924.5</td>
<td>32098.0</td>
<td>0.0110</td>
<td>0.763</td>
</tr>
</tbody>
</table>

As mentioned above, two main concerns of networks with SWP are clusteredness and compactness, which are also two characteristics of SWP. Clusteredness (Newman [11] calls it Transitivity) measures the cliquishness within neighborhood. Compactness measures the degrees of separation between every two nodes. It shares some similarity with radius of a network in the traditional graph theory, but with much more structural information. Note that C and L are not the only metrics for clusteredness and compactness respectively.

Properties can be divided into two kinds: one is based on local information (clusteredness), and the other on global structure (connectedness and compactness). A node can learn its local properties with the knowledge of its neighboring structure, and global ones with the status of the entire network. For instance, local variables are degree, TF, PF, clustering coefficient and the like. Since the extraction process is to find a keyphrase to summarize the whole text, we choose global properties such as connectedness and compactness to weight terms.

4.2 Feature weights

To analyze connectedness and compactness mentioned above, there are several variables to symbolize the importance of a node. These variables can be used as feature weights in unsupervised keyphrase extraction.

Take syntactic relation as a kind of information flow. One node connects other nodes through this flow. The cause of SWP lies in the existence of some in-formation hub, which keeps the network connected and compact. The possibility of the existence of certain topics increases when the network goes compacter. Therefore, if a keyphrase exists, it should be a hub that tightens the network.

We use "betweenness centrality" and "relation centrality", two global variables to capture the centrality of a term in the context and the role it plays in the compactness of the network. Keyphrases usually have high centralities. All candidate feature weights are defined below.

Connectedness Centrality \( H(v) \). \( H(v) \) captures the importance of a node in the connectedness of the network. Two nodes are unconnected if there is no path connects
them. $H(v)$ summarizes pairs of nodes that become unconnected if node $v$ is deleted from the graph.

Betweenness Centrality $B(v)$ [11] and Traffic $T(v)$. These two variables capture the role a node plays in the compactness of the network. $T(v)$ [10] sums the number of trajectories passing node $v$, identifying whether it is a hub, while $B(v)$ sums the number of shortest paths that node $v$ is in between. Since shortest paths contribute more to the compactness of networks than usual paths, we prefer $B(v)$ to $T(v)$. $B(v)$ is defined as below,

$$B(v) = \sum_{s \neq t \neq v} \frac{\sigma_{st}(v)}{\sigma_{st}},$$  \hspace{1cm} (3)

where $\sigma_{st}$ is the number of shortest paths between node $s$ and $t$, and $\sigma_{st}(v)$ sums the ones passing node $v$.

Moreover, we want to set up a composite feature weight that measures the importance of a node in both connectedness and compactness. A way to fulfill this task is to use variables to measure the structural change of the network in a dynamic behavior. Watts and Strogatz [8] define a key node as a shortcut in a Growth Model: shortcuts are nodes that decrease $E(v)$ dynamically by adding them. Note that comparing with [11], self-distance is excluded in formula (4) to avoid infinity. In essence, $L$ is a variant harmonic mean of $L(v)$.

$$L = 1/E, \quad L^{-1} = \frac{1}{2} \sum_{y \neq v}^\infty \frac{1}{d(y,v)^2},$$

where $E$ is called Efficiency[9], a measure of the efficiency of information flow in a network. The longer $L$ is, the lower $E$ becomes. $E(v)$ and $E$ are defined as:

$$E(v) = \frac{1}{n-1} \sum\frac{1}{d(v,v')} \quad E = \frac{\sum E(v)}{n}. $$

Note that comparing with [11], self-distance is excluded in formula (4) to avoid infinity. In essence, $L$ is a variant harmonic mean of $L(v)$. Accordingly to our experiment results, this definition unexpectedly outperforms the algebraic mean one of $L$. Moreover, the original definition of $L$ fails for unconnected graphs, but $L$ solves it by taking the infinite distance as zero in harmonic average. Thus, we define the increment of $L$ as $\Delta L_v$,

$$\Delta L_v = L - L_v,$$

where $L_v$ denotes $L$ after removing node $v$. Since relation keeps a network connected, and this variable can capture the role that a term plays in both compactness and connectedness of the network, we define it the feature weight Relation Centrality $S(v)$ of node $v$ as

$$S(v) = \Delta L_v. \hspace{1cm} (5)$$

$S(v)$ and $B(v)$ are intrinsically related. $S(v)$ dynamically measures the contribution a node makes to the compactness and connectedness of the network, and $B(v)$ statically counts how many routes $B(v)$ shortens. Either of them can independently distinguish keyphrases without training. Therefore, in the rest of unsupervised keyphrase extraction, we will concentrate on these two feature weights.

Our preliminary experiments in the effectiveness of these variables prove three facts. First, global variables perform far better than local variables. Second, keyphrases are unessential to act importantly in connectedness (connectedness centrality acts the worst among three centralities) Three, $B(v)$ and $S(v)$ act comparatively. Therefore, we provide two ways to extract keyphrases, one by ranking $B(v)$, the other by ranking $S(v)$.

4.3 Algorithm framework

Three main challenges exist in the algorithm. First, the computational complexity of calculating $L$ and $B$ is quite high. In graph theory, the most efficient algorithm of multi-source shortest paths lengths summarization is Floyd Algorithm, taking $O(n^3)$. Second, Large $n$ makes it worse, both for temporal and special cost. Space complexity should be taken in consideration, too. Finally, unconnectedness is an easily ignored issue in complex network analysis, but confines the use of $L$ as in (1).

![Figure 2. Workflow of keyphrase extraction.](image)

**Preprocessing.** A document is split into word sequences, identifying phrases. Before adding terms into the network, stopwords are filtered, and terms are stemmed. Caldeira et al [7] prove that stopword-filtering does not modify global behaviors of the network.

**Building the graph.** We store nodes and edges in adjacent list, to reduce computational complexity of space from $O(n^2)$ to $O(m+n)$. The possibility of memory overflow plummets. We initialize each node as a connected component. When an edge is introduced between components, the smaller component is merged into the bigger one. Then to avoid unconnectedness, we carry out pruning. We observe that in our data, most networks have a main connected component with more than 95% nodes of it (as in Table 1). Strogatz [3] reports it as a common phenomenon. Steyvers and Tenenbaum [13] have the same discovery in WordNet and Roget’s thesaurus, up to even 99% of 20,000 and 29,000 nodes, and conclude this as a main property of SWP. Note that in statistics in Table 1, only 11.5% has a $CC$ more than 1, and 96.9% among
11.5% has a CC more than 1, and 96.9% among which have \( m/n \) less than 2 -- they are approximately linear graphs. Henceforth, most of the semantic networks can be pruned into a main branch. As a result, we prune branches with less than 5% nodes.

**Algorithm 1: Summarize** \( L \) **for a semantic network**

1: \( L[v][w] \leftarrow 0 \)
2: for \( v \in V \) do
3: \( \text{visited}[u] \leftarrow 0, u \in V; \)
4: \( \text{level}[u] \leftarrow 0, u \in V; \)
5: \( Q \leftarrow \text{empty queue}; \quad \text{enqueue} \ v \rightarrow Q; \)
6: while \( Q \) not empty do
7: \( \text{dequeue} \ t \rightarrow Q; \)
8: if \( L[v][k] \leq \text{level}[t], k \in V \) then break; end if
9: foreach neighbor \( s \) of \( t \) and \( \text{visited}[s] = 0 \) do
10: \( \text{visited}[s] \leftarrow 1; \)
11: \( \text{level}[s] \leftarrow \text{level}[t] + 1; \)
12: if \( s < v \) then
13: for \( w \in V \) and \( L[v][w] > L[s][w] + \text{level}[s] \)
14: \( L[v][w] \leftarrow L[s][w] + \text{level}[s]; \)
15: end for
16: else if \( s > v \) and \( L[v][w] > \text{level}[s] \)
17: \( \text{enqueue} w \rightarrow Q; \)
18: end if
19: end foreach
20: end while
21: end for

Scoring. To overcome the time complexity, it is an intuitional way to delete some nodes before summarization of weights, but it is not a reasonable method since the deletion drastically changes the value of \( L \) and \( B \). However, the semantic networks are sparse \((m<n^2)\), unweighted, and undirected. Calculating \( L \) and \( B \) can be simplified as a task of breath-first searching through an adjacent list. By using this method, we reduce the computational complexity of calculating \( L \) and \( B \) greatly. For \( B \), only \( O(n+m) \) space and \( O(mn) \) time is required. For \( L \), we store and re-use results of previous iteration, disregarding route information. The best time cost will be \( k \frac{n(n+1)}{2} \) with a \( O(n^2) \) space cost, where \( k \) is the average time cost for calculating the distance of one path.

Unstemming. Unstemming is a process that adds the suffix back to stemmed keyphrases, making them understandable. However, unstemming is an ill-posed problem (one-to-many mapping). Currently our unstemming algorithm is: search phrases with the stem, pick up the phrase with highest PF, and partially stem its suffixes (plural form of nouns and –ing, -ed tense forms of verbs).

4.4 Applying to supervised keyphrase extraction

Though the unsupervised method above performs comparatively with existing supervised system (shown in Section 5.3), it still shares two drawbacks with KEA and Extractor. First, the precision is low, usually below 50%. Second, the expected number of keyphrase is fixed. This fulfills only the tasks when the number is clear, otherwise, it acts unstable. Towards these ends, we apply our algorithm to a supervised task, aiming at building a more accurate classifier that dynamically determines the number.

As a supervised learning task, we treat it as a two-category classification task -- a term in the document is a keyphrase (positive instance) or not (negative instance). A feature is a kind of property that is helpful for determining the case (positive or negative) of an instance. Based on some features, we build up a classifier to extract keyphrases. Support Vector Machine (SVM) is a powerful method to build up accurate classifiers.

*Feature selection.* To assemble more information, we choose both statistical and syntactic features from local and global properties. As local properties, degree, PF or TF, and \( C(v) \) (defined in (2)) are included. For global information, \( L(v) \) (defined in (4)), connected centrality \( H(v) \), betweenness centrality \( B(v) \) (in (3)), relation centrality \( S(v) \) (in (5)) are all included. Moreover, we normalize these features with the factors defined in the table below.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Normalization factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>Number of nodes in the graph.</td>
</tr>
<tr>
<td>PF</td>
<td>Number of words in the document.</td>
</tr>
<tr>
<td>C(v)</td>
<td>C of the graph.</td>
</tr>
<tr>
<td>L(v)</td>
<td>L of the graph.</td>
</tr>
<tr>
<td>H(v)</td>
<td>Square of number of nodes in the graph.</td>
</tr>
<tr>
<td>S(v)</td>
<td>L of the graph.</td>
</tr>
<tr>
<td>B(v)</td>
<td>Square of number of nodes in the graph.</td>
</tr>
</tbody>
</table>

Experiments (Section 5.5) show that among most of the tested kernel functions, normalized features have a higher accuracy in both negative instances and the entire test set, but a lower in positive instances.

*Learning an SVM classifier.* Each instance is a vector in the sample space, where one dimension is a feature, valued by the corresponding feature weight.

To facilitate the learning of a classifier, we use LIBSVM [24], an influential code library of SVM. The learning proceeds as below: select training set and testing set, preprocess the data, choose a family of classifier, tune the parameters, train the model, and test it.

The traditional data selection approach is to select data from the set randomly from the whole dataset. However, in keyphrase extraction, numbers of different categories are highly unbalanced. Positive instances constitute a very small part in the sample space (the percentage is 0.2% to
2.4% in Turney’s dataset [1] and far below 0.1% in our eBook archive. Thus positive instances have minute possibility to be chosen. To overcome this problem, we choose half of the training set from positive instances and the rest from negative ones. In either half, each instance is selected randomly the category. As for preprocessing, we scale training set and testing set with L-1 norm. We select C-SVC [24] as the type of SVM, linear kernel, RBF kernel, and sigmoid kernel as candidate kernel. Possibility feedback [24] is included, Grid search algorithm [24] and five-fold cross validation is used in parameter tuning.

We randomly select 1212 eBooks as the source of training set, including 2379 positive instances and 106115 negative instances. Note that only topical terms are selected as positive instances. In data selection, we select 1000 positive instances and 1000 negative instances as the training set. Results are in Section 5.5.

5. Experiments

We carry out three experiments on the effectiveness and efficiency of our keyphrase extraction algorithm, denoted SW. For unsupervised tasks, it provides two results, one is from the ranking of relation centrality \( S(v) \) and the other from betweenness centrality \( B(v) \). Moreover, we detail experiments for relationship establishment and the effectiveness of supervised keyphrase extraction mentioned in Section 3.2 and Section 4.4 respectively.

5.1. A case study

In this experiment, we choose two books, rank candidate keyphrases to \( S(v) \) or \( B(v) \), and select top 7 of the returned list (7 is for page limitation). Moreover, we use Copernic Summarizer™ - a leading keyphrase extractor without training - as the baseline (it may be trained before it is published). Extractor [1] has been integrated into it.

Table 2 shows the result. The title of book B1 is “Price responsiveness of world grain market”, and book B2 is “More milk for more children”. From the table, we can see that from semantic point of view, most topical terms from LOC are included in \( SW \). \( S(v) \) and \( B(v) \) slightly performs better in two aspects. First, fewer unrelated keyphrases, such as “pint” in B2. Second, effective phrase identification finds several useful keyphrases such as “Agricultural Economics” and “wheat and rice” in B1. But two problems remain in all these systems. First, simple keywords should be merged into phrases as in USMARC of B2. Phrase identification or some postprocessing can be a solution. Second, synonym mapping is needed between “grain” and “wheat and rice” in B1. Introduction of semantic information, such as synset might be helpful.

<table>
<thead>
<tr>
<th>Book</th>
<th>System</th>
<th>Keyphrases (top 7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>USMARC</td>
<td>Grain trade, Intervention (Federal government), Elasticity (Economics)</td>
<td></td>
</tr>
<tr>
<td>Summarizer</td>
<td>price, government intervention, elasticity, wheat, rice, trade, consumption</td>
<td></td>
</tr>
<tr>
<td>B1</td>
<td>( S(v) )</td>
<td>countries, import, government intervention, Agricultural Economics, price, wheat and rice, elasticity</td>
</tr>
<tr>
<td></td>
<td>( B(v) )</td>
<td>countries, import, price, wheat and rice, government intervention, elasticity, Agricultural Economics</td>
</tr>
<tr>
<td>USMARC</td>
<td>School milk programs, School children (Food)</td>
<td></td>
</tr>
<tr>
<td>Summarizer</td>
<td>milk, school, Agricultural Marketing Administration, sponsor, farmers, community, pint,</td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>( S(v) )</td>
<td>milk, school, children, program, drink, Agricultural Marketing, farmer price</td>
</tr>
<tr>
<td></td>
<td>( B(v) )</td>
<td>milk, school, children, program, sponsor, Agricultural Marketing, drink</td>
</tr>
</tbody>
</table>

5.2 Comparison among feature weights

In this experiment, we compare \( SW \) with two prevailing features: TFIDF, widely used in document relevance analysis and keyphrase extraction, such as KEA; BM25, a prevailing feature used in text indexing and ranking in Information Retrieval. They are defined as

\[
w(t, d) = TF(t, d) \cdot \ln \frac{|d|}{DF(t)},
\]

\[
w(t, d) = \frac{(k_1 + 1)TF(t, d)}{k_1((1-b)+b\frac{d(t)}{avgdl})+TF(t, d)} \cdot \ln \frac{|d|}{DF(t)+0.5} \frac{DF(t)+0.5}{DF(t)+0.5}
\]

respectively, where \( TF \) stands for term frequency of term \( t \) in document \( d \), \( DF \) for the number of documents that consist of this term, \( |d| \) for the number of documents in the dataset, \( dl \) for the number of terms in the document, \( avgdl \) for the averaged \( dl \) in the dataset, and \( b \) and \( k_1 \) are two parameter need tuning. The first formula is the traditional unnormalized TFIDF and the second one is defined as in Roberson’s paper [22]. Here, we select \( b \) and \( k_1 \) to be 0.75 and 2 empirically as in most application in TREC. TFIDF/BM25 denotes their common performance.

Data Generation. DUC is well-known to test the performance on automatic summarization. We choose all 1600 documents of DUC2005 as a test set. Most of these web pages are from \( FT \) and \( LATimes \). They are labeled into 50 categories with topic terms by experts. Therefore, we use these topic terms as the target output and documents as input to extract keyphrases. All the extraction algorithms use he same candidate keyphrase identification method and no TF filtering on terms.
Figure 3. Result of SW versus TFIDF and BM25. (a, c) Ranks comparison. (b, d) Weights comparison. The x-axis is values returned by TFIDF/BM25 and y-axis is S(v) in blue and B(v) in red. LR stands for Linear Regression of Ranks comparison. (b, d) Weights comparison.

Evaluation Measure. To evaluate the result, we resort to five measures: the ranking of the query term in the feature set, the normalized weight of the term, the size of this feature set, time duration, and miss rate of target keyphrases. Though precision and recall are widely used, they can’t measure precisely the ranking and weight difference between feature metrics. To compare the weights between different metrics, we use the L-2 norm as the normalization factor for all four features, supposing between different metrics, we use the L-2 norm as the difference between feature metrics. To compare the weights, we use similar measures, the average precision within top n keyphrases in a dataset, denoted P@n, taking account of the number of assigned keyphrases, the precision will be very low that makes no significance. So we use n less than 15.

Results. Running on DELL workstation with Intel Xeon 3.0G CPU and 1.0G memory, TFIDF/BM25 finishes in 96’53'' (more than 1.5 hour) and SW in 72’21'', about 2/3 of the previous one. SW filters nodes with only one degree in the returned set, since outskirts of the network won’t have a high S(v) nor B(v). As a result, SW has (71.0 ± 7.1)% the size of candidate phrase set of TFIDF/BM25, at a cost of a (6.7 ± 4.3)% increase in miss rate. Other results are shown in the two figures bellow. We set up baselines in the graph to clarify the differences of performance. If all sample points are on the baseline, it means that the system of x-axis and that of y-axis have the same performance. Seen from the graphs below, in (a) and (c), all sample points are above the baseline, indicating that all ranks of SW are smaller than TFIDF and BM25, and in (b) and (d), most of the sample points are below the baseline, indicating that most weights of SW are bigger than the other two. As a summarization of all samples and a tendency prediction, the result of linear regression functions shows that the performance of S(v) and B(v) are close, while they both outperform TFIDF/BM25 with relatively higher ranks (smaller value, less than a half) and bigger weights (nearly 30% promotion from TFIDF and 80% from BM25).

5.3 Comparison among extraction systems

This experiment aims at the comparison between SW (S(v) and B(v)) and other extraction systems, no matter supervised or unsupervised. Since the source code or service of PhraseRate is unavailable, we choose two influential systems, KEA and Summarizer™ mentioned in Section 2 as baselines. The keyphrases assigned by authors or librarians are also included as an optimal baseline, denoted as Author.

Data Generation. We use three kinds of documents to be datasets, web pages, journal papers, and some eBooks from our archive. The first two kinds of datasets include FIPS, Aliweb, NASA, and Journals are all the same as in Turney’s paper [1]. Each consists of source documents and keys assigned by the authors. We train KEA with the same training set (55 documents of Journal) as in [1]. The eBook dataset includes 101 English books, randomly chosen from all kinds of fields, with 12528.0±4118.4 terms. The number of keyphrases that appears in the document is 1.4752±0.8074. Assigned topical terms are from LOC as in Experiment 1. 90.26% of these keyphrases exist in the text. We regroup these five datasets into three for the length of the documents: Aliweb and NASA are grouped into Short, FIPS and Journal into Mid, and eBook remains.

Evaluation Measure. To make the result more easily to compare with previous work of KEA and Extractor, we use similar measures, the average precision within top n keyphrases in a dataset, denoted P@n, taking account of both accuracy and ranking. If n is far bigger than the number of assigned keyphrases, the precision will be very low that makes no significance. So we use n less than 15.
Figure 4. Precision comparison between four systems in three datasets. (a) In eBook dataset. (b) In Mid dataset. (c) In Short dataset.

Results. Without training, S(v) and B(v) still perform comparatively as supervised extractors. Moreover, in eBook and Mid, they slightly outperform Summarizer in most of the time, and they even perform best in the first five keyphrases. This result validates our assumption that dependency between neighboring phrases helps extract keyphrases. On the other hand, they perform relatively poor in Short. The reason is: in short documents, the relation is so sterile that most nodes with very low PF and degree make no differences in a semantic network, while in TFIDF they may do (KEA uses normalized TFIDF and Summarizer use normalized PF as a feature). Note that in Mid, KEA outperforms other systems when the number of keyphrases is larger than 5. A possible reason is that it is trained on this dataset, but its performance on the first 5 keyphrases is still worst of all. Moreover, we can witness a big precision margin between state-of-art keyphrase extraction systems and optimal baseline Author.

5.4 Experiment for relationship establishment

We carry out this experiment to compare the performance of neighboring with other definition of relationship in the semantic network. Relationships between terms in this comparison verify in the size of co-occurrence window: neighboring (size is 2, denoted Bi), Tri (size is 3), Quad (size is 4), and co-occurrence (Occ) in the same sentence. We use the same dataset and evaluation measure as in Section 5.4. Terms are scored by S(v) and B(v).

Concluded from the results, these four kinds of relationship perform roughly the same. However, Bi has the lowest computational complexity, while the counting iterations of other relation are several times of Bi: Tri 2 times, Quad 3 times, Co-occurrence 4-10 times.

5.5 Experiment for supervised extraction

We carry out this experiment to test the performance of extraction with normalized or unnormalized feature set, and with different kernels functions in SVM. The families of kernels we choose include: linear function, RBF function, sigmoid function, and their functions with probability feedback (denoted L-Prob, R-Prob, and S-Prob). The test set is 10555 instance vectors (191 positive instances and 10364 negative ones) extracted from the eBook dataset mentioned before. Since the data is highly unbalanced, we test the accuracy in the set of positive instances (denoted Pos), the set of negative instances (denoted Neg), and the entire test set (denoted All).

Table 4. Accuracy of different kernels. Unnorm Acc. stands for accuracy with unnormalized feature, and Norm Acc. for accuracy with normalized feature, ' for minute.

<table>
<thead>
<tr>
<th>kernel</th>
<th>Unnorm Acc. (%)</th>
<th>Norm Acc. (%)</th>
<th>Train time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pos</td>
<td>Neg</td>
<td>All</td>
</tr>
<tr>
<td>Linear</td>
<td>74.9</td>
<td>83.4</td>
<td>83.3</td>
</tr>
<tr>
<td>L-Prob</td>
<td>57.1</td>
<td>93.9</td>
<td>93.2</td>
</tr>
<tr>
<td>RBF</td>
<td>77.0</td>
<td>80.5</td>
<td>80.5</td>
</tr>
<tr>
<td>R-Prob</td>
<td>77.0</td>
<td>81.3</td>
<td>81.2</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>1.57</td>
<td>99.9</td>
<td>98.2</td>
</tr>
<tr>
<td>S-Prob</td>
<td>0</td>
<td>100</td>
<td>98.2</td>
</tr>
</tbody>
</table>

The result of this experiment is shown above. For most of the kernels, normalization brings a higher accuracy in both negative instances and the entire test set and a lower in positive instances. On the other hand, for L-Prob and S-Prob, the result is reverse. Linear, L-Prob, RBF, and R-Prob performs comparatively though linear/L-Prob has a better accuracy on Neg and All, and RBF/R-Prob has a better accuracy on Pos. Sigmoid has a very low accuracy on Pos (in fact, this is recall), but a very high precision (since Neg outnumbers Pos greatly, accuracy of Neg is approximately the same with precision).

For efficiency, the training time depends on the sample size and the kernel function, and it has no significant relation with the range of features. Sigmoid takes the least training time, followed by RBF and linear. Feedback of
probability requires a different approach of learning [24], which affects efficiency negatively.

6. Conclusion

Keyphrase extraction is a powerful tool for text summarization and similarity analysis. It is traditionally solved by supervised learning. Due to lack of fast, accurate feature weights that are capable to generalize, unsupervised approach is unpractical. We propose an algorithm, using two network structural variables as feature weights, either of which outperforms traditional ones both in effectiveness and efficiency in unsupervised tasks. Moreover, we apply this algorithm to a supervised task, and develop a classifier with accuracy up to 80%, which can automatically decide the number of keyphrases.

Though the performance is encouraging, different definition of relation and more syntactic and semantic information might greatly enhance the performance as well. Directed and weighted graphs are still choices.

Acknowledgement

This work is supported partially by National Sciences Foundation of China (NSF60605020), the 863 Project of China (2003AA119010), and China-US Digital Academy Library Project (CADAL2004002). We are thankful to Mr. Peter D. Turney for generously sharing his datasets and Prof. Jinhu Lu, Prof. Qun Liu, Prof. Chengqing Zong, Yuanning Li, Fei Yang, Alexander Markowetz, and several researchers of MSRA for their comments.

References