Keyphrase Extraction using Semantic Networks Structure Analysis

Chong Huang[1], Yonghong Tian[2], Zhi Zhou[2], Charles X. Ling[3] , Tiejun Huang[1]
[1] *Graduate University, Chinese Academy of Sciences, Beijing 100039, China*
[2] *Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China*
[3] *Department of Computer Science, University of Western Ontario, N6A 5B7, Canada*
[1, 2] *{chuang, yhtian, tjhuang, zzhou}@jdl.ac.cn*
[3] *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 using two novel feature weights, which can be used in both supervised and unsupervised tasks. This algorithm treats each document as a semantic network that holds both syntactic and statistical information. Structural dynamics of these networks can easily identify key nodes, which can be used to extract keyphrases unsupervisedly. Experiments demonstrate the proposed keyphrase extraction 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 task, we develop a more accurate classifier. In this classifier, we assemble several syntactic and statistical features. Experiments show that the overall precision of supervised extraction can be 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 project[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; instead, we plan to use extracted keyphrases. Therefore, keyphrase extraction is of interest.

Most existing extraction algorithms [1, 18] are based on supervised learning for 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. Unsupervised learning is thus more preferable. Last but most important, most prevailing feature weights [15, 16] are not satisfactory for unsupervised keyphrase extraction. A *feature weight* (also called feature metric or term weighting scheme in text mining) measures the feature value of a term in the source document, which helps determine whether the term is a keyphrase or not. Most prevailing feature weights solely rely on frequency-based [22] or specific structural information [17]. They are either too inaccurate or too hard to generalize. As a result, the accuracy of unsupervised learning using a single weight is unsatisfactory. Additionally, among these feature weights, some are set-dependent [22] (set-dependent means that if a new document is added to the data set, the *set-dependent* feature weights of all the existing documents needs 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 structural variables of these networks, we notice the existence of Small-World Phenomenon (SWP) [11] – the networks are clustered, compact and connected. There are key nodes that play important roles in the structure of these networks, making them compact. We use several structural variables of SWP as feature weights to find out these key nodes. Given that the structure of the network represents the structure of the source document, and the edges represent syntactic relation, we select some key nodes as keyphrases. In the process of calculating these variables, no training set is required. 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, aiming at building a more accurate classifier that can dynamically determine the number of the keyphrase. Experiments show that the overall accuracy of this supervised extraction method can be up to 80%.

Section 2 surveys recent literatures on keyphrase extraction and semantic networks structure analysis. Section

[1] This is the first large international digital library cooperation project, aiming at organizing one million Chinese or English books online.

3 outlines our knowledge organization system. Section 4 presents our unsupervised keyphrase extraction algorithm, and its application to build up a more accurate classifier for keyphrase extraction by assembling several features. Section 5 describes experiments on the effectiveness and efficiency of this keyphrase extraction method. Section 6 concludes this paper.

2. Background

2.1 Keyphrase extraction

In text mining, keyword acquisition falls into two ways: keyword assignment and keyword extraction. They differentiate in the source of the keywords: keyword assignment acquires keywords from predefined corpus or thesaurus, and keyword extraction from the source document. Keyphrase extraction extends the acquisition from words to phrases. Phrase identification is a main issue during this extension. Some hypotheses are set up to select phrases as candidate keyphrases, or all word sequences are candidate keyphrases.

Keyphrase extraction can be viewed as a supervised or unsupervised learning task. Though lots of literatures about keyphrases have been published, most of them cover supervised tasks. Viewing keyphrase extraction as a supervised learning task, researchers develop two methods. Intuitively, it is a two-category classification task -- a term in the document is a keyphrase or not. Based on some features, researchers build up classifiers to extract keyphrases. In a pioneer paper [1], Turney first treat it as a supervised learning task, by using frequency-based and part-of-speech information as features, along with decision tree and an generic algorithm (called Genex) as classifiers. The system is named Extractor. KEA [18] performs in a similar way with Native Bayes as the classifier. On the other hand, researchers in natural language processing build up language models from a training set and select phrases whose feature weights are in accord with the distribution of keyphrases [24]. As an unsupervised learning task, it can be fulfilled by the means of ranking: set up some feature weights, rank the phrases under these metrics, and select phrases with top weights. PhraseRate[14] shares a similar goal with us to set up a feature weight to extract keyphrase without training, but it depends on HTML tags.

Feature weights or feature metrics are essential in the preprocessing of data mining and pattern recognition, for extracting representative features of the objects. 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 feature weights rely on priors or statistics acquired in the training process, 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-based information. Firstly, some weights set up rules on specific structure information [17], such as tags or positions, but they have limited capacity to generalize. Methods based on frequency information can be Term Frequency (TF), TFIDF [18], or more delicately Okapi's BM25 [22]. Though they are widely used in information retrieval, there are several drawbacks. First, they are not accurate enough for unsupervised keyphrase extraction. This is the reason 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 the order or co-occurrence of words within a window in a fixed or dynamic size. Mihalcea and Tarau [23] use a graph to store the co-occurrence in a window of N words, view the relation as recommending, and rank the words in a PageRank-like manner. We store ordinal information in a similar way, but view a relation as a path for information flow.

In a word, our main contributions are: (1) proposing a fast and effective phrase identification algorithm; (2) using structural variables of Small-World Phenomenon (SWP) as feature weights, which make the result of unsupervised keyphrase extraction satisfactory.

2.2 Network structure analysis for natural language processing

Currently, a body of literatures is on the study of structural analysis on kinds of complex networks. Researchers have developed a system of theory on how to analyze the global structure of complex networks [11]. Network structure analysis is applied to kinds of semantic network structure analysis in natural language processing, including some for lexical pattern analysis [5], ontology of language, and language evolvement [6, 13], such as Associative Network, WordNet.

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.

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 [3], and betweenness for structure analysis. Among them, two most related works are: Zhu et al [4] use ΔL_v to extract keywords of Chinese news web pages, in a relatively small dataset without theoretical reasoning (we have a different definition of ΔL_v); Fortunato et al [21] use a similar approach with us (they call it information centrality), to analyze edges rather than nodes to find community structures. However, most of these literatures don't mention many practical issues, such as unconnectedness and computational complexity.

3. Constructing semantic networks for phrases

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 within nodes, and other ordinal or dependency information is stored in the structure of the network, which we will utilize in the extraction algorithm in the form of feature weights.

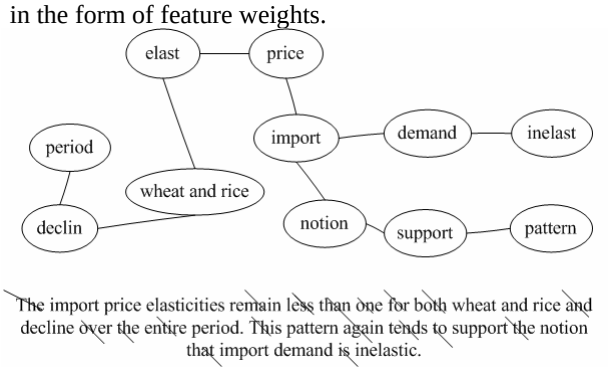


Figure 1. Semantic Networks Model. The sentence on the bottom is the original context and the graph is the corresponding semantic network. Nodes in this network represent stemmed terms (phrases are identified before the construction of the network) and edges represent neighboring relation, which potentially indicates syntactic relation. The slashed words are stopwords to filter.

3.1 Phrase identification

A *phrase* is a consecutive sequence of words without intervening punctuations, forming a grammatical constituent of a sentence. In the process of establishing nodes, we need to identify phrases among meaningless word sequence, and add the candidate phrases and the remainquence, and add the candidate phrases and the remaining single words into the network. To identify phrases, an intuitive way is to combine the candidate set of all possible word sequences with the word set, score them, and then select kephrases from them. Unluckily, this approach introduces compositional explosion. Hence, in KEA and Extractor, hypotheses are set up before scoring to filter meaningless word sequence. Such hypotheses are "neither stopwords (letters, conjunctions, articles, particles, prepositions, pronouns, anomalous verbs, adjectives, and adverbs. We use a similar stopword list as in KEA) nor named entities are in between" and "a phrase has a maximum length (usually three)". But these hypotheses are so weak that most of the phrases in the candidate set are still semantically meaningless. Compositional explosion is still a haunted threat. Moreover, in semantic networks, structural variables are very sensitive to the changing of the structure. The failure of phrase identification leads to error values of these variable, and finally low precision of keyphrases extraction.

After investigating the characteristics of topical terms in our eBook archive, we found a two-phase filtering algorithm more effective. On the first setp, three rules must be fulfilled; otherwise, the word sequence is filtered:

1. A candidate phrases can't start or end up with stopwords.
2. In a candidate phrase, between two non-stopwords, there can be no stopwords except *MidStopwords* (proposition, nouns, and numbers in stopword list), and the number of MidStopwords should be less than four. Phrases as "wheat and rice" are included, while they are not in Extractor.
3. The phrase frequency (PF) of a candidate phrase should be relatively high in the text.

Second, we divide the remaining ones into groups. Candidate phrases are first divided into societies by the number of words they have, and then phrases of a society with the same word are further assigned to the same group. A candidate will be in n groups if it has n distinct non-stopwords. Every group has only one or none winner. A winning keyphrase candidate (we call it 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. Only winners can remain in the candidate set of keyphrases. Take $phrase_i$ in $group_k$ for example ($phrase_i$ consists of $word_k$),

$$PPF(phrase_i, group_k) = \frac{PF_i}{\sum_{phrase_j \in group_k} PF_j},$$

$$PTF(phrase_i, group_k) = \frac{PF_i}{TF_k}.$$

where $PPF(phrase_i, group_k)$ depicts the percentage of frequency of $phrase_i$ among all phrases in $group_k$, and $PTF(phrase_i, group_k)$ depicts the percentage of frequency

of $phrase_i$ versus the frequency of $word_k$. Note that PF is the frequency of a phrase consist of more than one word, TF is the frequency of a single word. Since many word sequences have been filtered in the first phase, PPF in the remaining candidate set is usually much greater than PTF.

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 have only keyphrases with less than two words (in fact most of the remainder 15% are made up by two keyphrases), we now support keyphrases with two words, and this algorithm is extensive to a longer length.

3.2 Relationship establishment

Relationship stores the majority of information in a semantic network, which distinguish it from other types of KOSs. There are mainly three kinds of relationship between words, semantic relation, syntactic relation, and co-occurrence.

Semantic relation can only be established when a dictionary or ontology as WordNet is available, and disambiguation is needed. Syntactic relation builds networks on the grounds that words at a certain distance have syntactic or semantic relationship [5, 6]. It needs a supervised learning process for Part-of-speech (POS) tagging. Co-occurrence presumes that co-occurrence in some linguistic units (chapter, discourse, paragraph, and sentence) is syntactically or semantically indicative [4, 7]. Since co-occurrence is an approximate way to identify relationship, it is prone to introduce redundancy and false relation. Moreover, networks linked up by co-occurrence have the most edges among these three relations, leading to a highest computational complexity.

We aim to combine the last two kinds of relation, without a training set for POS and a relative low computational complexity. Lyon et al [6] study that 70% of syntactic dependencies are between neighboring words, and 17% at a distance of 2. Furthermore, 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 relationship in the same sentence as the relationship between nodes. We carry out an experiment to validate the performance of neighboring relation (see Section 5.4) in several datasets. The result is that its performance is comparative to other relation with a larger co-occurrence window. Also, its computational and structural 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 words 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 distinguish different 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, they are two kinds of syntactic relation, but in the neighboring relation, the only concern is the existence of syntactic relation, rather than what kind of syntactic relation they are. Therefore, they are the same in this graph.

4. Extracting Keyphrases using Semantic Networks Structure Analysis

With semantic networks model and characteristics of candidate keyphrases defined, we now move to extract keyphrases by analyzing the structure of these networks without training. 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 a 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 ranking phrases, it selects top n phrases as keyphrases. The n can be decided by the demand of the extraction.

4.1 Network structure analysis

To kind key nodes in a semantic network, we must first decide what characteristic of network structure we will 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 connected graph will turn into unconnected after removing a node, and if it will, how unconnected it will be. Nodes that link different components are recognized as key nodes, named *cut points*. Alternatively, nodes with highest input and/or output degrees might be of interest.

In complex network analysis, Small-World Phenomenon (SWP) is viewed as an important property of network structure. Though a network has a large amount of nodes, it still has small average minimum path length *L*, and high clustering coefficient *C* [8]. This is called SWP, also known as the Six Degree Separation Rule in Sociology, which is found to be a common phenomenon in complex networks. Characteristic Path Length (average shortest path length) [8] of node *v*, denoted as *L(v)*, demonstrates the average length of all shortest paths started from node *v*, and *L* is the average of *L(v)* among all starting nodes in the network. They are defined as:

$$L(v_i) = \frac{\sum_{v_j \in V} d(v_i, v_j)}{n-1}, L = \frac{\sum_{v_i \in V} L(v_i)}{n}, \quad (1)$$

where $d(v_i, v_j)$ stands for the distance between v_i and v_j in the network. Clustering coefficient [8] of node *v*, denoted as *C(v)*, depicts how fully connected node *v* and all its neighbors behave, and *C* of the network averages *C(v)* in the scope of all nodes. They are defined as bellow, where R_i is the number of pair-bonding links among v_i and its *l* neighbors,

$$C(v_i) = \frac{2R_i}{l(l-1)}, C = \frac{\sum_{v_i \in V} C(v_i)}{n}. \quad (2)$$

Since there are thousands or even millions of words in one book, the structure of the corresponding network is rather complex with up to 60,000 nodes and 400,000 edges. Concluded from Table 1, SWP happens in our semantic networks.

Table 1. Statistics of variables. They are from 9877 semantic networks of books, randomly sampled from our eBook archive, where a network corresponds to a book. L and C are defined above, m stands for the number of edges, n for the number of nodes, CC for the number of connected components, EX for the expectation of variable X, and DX for the variance of X.

X	L	C	n	m	m/n^2	CC
EX	4.209	0.629	4295.1	27690.2	0.0027	1.207
DX	0.826	0.045	3924.5	32098.0	0.0110	0.763

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 of a typical neighbor. *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, while global properties with the knowledge of the status of the entire network. For instance, variables for local properties 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 kind of flow. The cause of SWP lies in the existence of some information 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,

$$B(v) = \sum_{\substack{s \neq v \neq t \\ s,t,v \in V}} \frac{\sigma_{st}(v)}{\sigma_{st}}, \quad (3)$$

where σ_{st} is the number of shortest paths between node *s* and *t*, and $\sigma_{st}(v)$ sums the number of these paths 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 (a network grows from a node to a graph): shortcuts are nodes

that decrease L drastically when they are added into the network. Because we care the status quo more than the evolving process of the network, we view them in a Decadence Model: if these key nodes are removed, L soars, or even the network collapses. In consideration of unconnectedness of our semantic network, we define L as below (a variation from [11]):

$$L' = 1/E,\ L'^{-1} = \frac{1}{\frac{1}{2}n(n-1)}\sum_{i\geq j} d_{ij}^{-1}. \qquad (4)$$

where E is called *Efficiency*[9], which measures how efficiently information flows over a network. The longer L is, the lower E becomes. $E(v)$ and E are defined as:

$$E(v_i) = \frac{1}{n-1}\sum_{v_j \in V} \frac{1}{d(v_i, v_j)},\ E = \frac{\sum_{v_i \in V} E(v_i)}{n}.$$

Note that self-distances are excluded in formula (4) to avoid infinity (while in [11] they are included). In essence, L' is a variant harmonic mean of $L(v)$. According 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

$$S(v) = \Delta L'_v. \qquad (5)$$

*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

To implement the algorithm above, there are three main challenges. First, since n is very large, the computational complexity of calculating L and B is quite high. In graph theory, the most famous and efficient algorithm to summarize lengths of all shortest paths is Floyd Algorithm, which takes $O(n^3)$. It will be an intolerable experience for one to extract metadata from a book when n comes to millions. Space complexity should be taken in consideration, too. Finally, unconnectedness is an easily ignored issue in complex network analysis, confining the use of L.

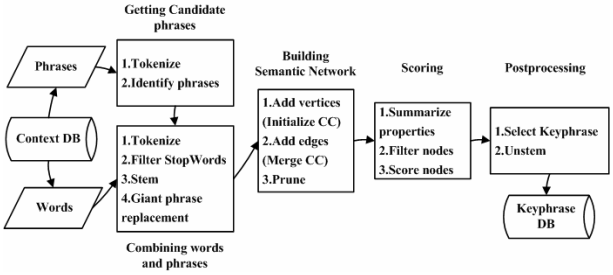


Figure 2. Workflow of keyphrase extraction.

Preprocessing. The algorithm starts in splitting context into word seuqences, identify candidate phrases. Before adding words into semantic network, we omit the words in stopword list, and substitute words with phrases that consist of them. Caldeira et al [7] have proved that this filtering treatment does not modify general behaviors of the network. Since most source books are in English, we use PorterStermmer [12] to map words to their stems.

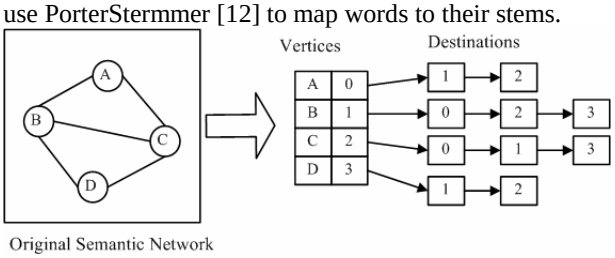


Figure 3. Structure of an adjacent list.

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 [2] 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 which have *m/n* less than 2 -- they are approximately linear graghs. Henceforth, most of the semantic networks can be pruned into a main branch. As a result, we prune branches with less than 5% nodes.

Scoring. To overcome the time complexity, it is an intuitive 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 only 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 reuse results of previous iteration, disregarding route information. The best time cost will be $k\frac{n(n+1)}{2}$ with a O($\frac{n(n+1)}{2}$) space cost, where k is the average time cost for calculating the distance of one path.

Algorithm 1: Summarize *L* of semantic network

```
1: L[v][w]←0
2: for v∈V do
3:   visited[u] ←0, u∈V;
4:   level[u] ←0, u∈V;
5:   Q ←empty queue; enqueue v→Q;
6:   while Q not empty do
7:       dequeue t←Q;
8:       if L[v][k]<= level[t] k∈V then break; end if
9:       foreach neighbor s of t and visited[s]=0 do
10:        visited[s] ←1;
11:        level[s] ←level[t]+1;
12:        if s<v then
13:          for w∈V and L[v][w]>L[s][w]+level[s] do
14:              L[v][w] ←L[s][w]+level[s];
15:          end for
16:        else if s>v and L[v][w]>level[s]
17:          enqueue w→Q;
18:        end if
19:      end foreach
20:  end while
21:end for
```

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 keyphrase extraction above performs comparatively with exsisting supervised keyphrase extraction system (shown in Section 5), it still shares two drawbacks with KEA and Extractor. First, the precision is not high enough, usually below 50%. Second, the number of keyphrase is fixed. This might fulfill the tasks that the user intends to decide the number of returned keyphrases himself, but it fails when the number is unclear. Towards these ends, we apply our unsupervised keyphrase extraction algorithm to a supervised task, aiming at building a more accurate classifier that can dynamically determine the size of the returned keyphrase set.

Viewing it 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). To assign instances, we need a set of features to classify terms. A *feature* is a kind of property that is helpful for determining the case (positive or negative) of an instance. Based on these features, we build up a *classifier* to extract keyphrases. Support Vector Machine (SVM) is a powerful method to build up accurate classifiers for data classification. It provides a model to predict target category or probability of instances. In fact, the model is an accurate hypersphere in the data space (a feature as a dimension) to classify the instances.

Feature selection. To assemble more information as, we choose both statistical and syntactic features from both local and global properties. For local properties, degree, PF or TF, and cluster coefficient $C(v)$ (defined in formula (2)) are included. For global information, averaged shortest path length $L(v)$ (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 normalization factors defined in the table below. For example, $H(v)$ is defined for pairs of nodes in the graph, and the maximum number of pairs is square of the number of nodes. Therefore, we use it as the normalization factor. Note that no normalization is used in unsupervised task, because a feature weight is used for ranking in only one source document, and these factors have no impact on this ranking.

Table 2. Normalization factors of variables.

Variable	Normalization factor
Degree	Number of nodes in the graph.
PF	Number of words in the document.
C(v)	C of the graph.
L(v)	L of the graph.
H(v)	Square of number of nodes in the graph.
S(v)	L of the graph.
B(v)	Square of number of nodes in the graph.

In the experiment of comparison between extraction with original variables and extraction with normalized ones, among RBF, RBF with probability, and sigmoid

kernels, the latter one has a higher accuracy in both negative instances and the entire test set, but a lower in positive instances, but in linear kernel, the result is reverse.

Learning an SVM classifier. Given the features selected above, we view each instance a vector in the sample space, where a dimension is a feature, valued by the corresponding feature weight.

To facilitate the learning of a classifier, we use LIBSVM [25], an influential code library of SVM. The learning proceeds as below: select training set and testing set, preprocess the data, choose a class of classifier, tune the parameters, train the model, and test it.

The traditional data selection approach is to select data from the set randomly, ignoring what category it is. 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). In the traditional sampling approach, positive instances have minute possibility to be chosen into the training set. To overcome this problem, we choose half of the instances in the training set from positive instances and another half from negative instances, and each instance is selected randomly from the instances of the same category. As for preprocessing, we use the L-1 norm scaling schemes as LIBSVM provides. Note that training set and testing set should be scaled in a similar way. We select C-SVC [25] as the type of SVM, linear kernel, RBF kernel, and sigmoid kernel as candidate kernel. Grid search algorithm and five-fold cross validation is used in parameter tuning.

We randomly select 1212 books from our archive as the source of training set, including 2379 positive instances and 106115 negative instances. Note that only topical terms excluding subfields are selected as positive instances. For example, if a LOC-assigned subject is "Trusts Industrial (United States)", and "United States" is a geographical subfield of topical term "Trust Industrial", we only select "Trust Industrial" as a positive instance. In data selection, we select 1000 positive instances and 1000 negative instances as the training set from this data set. The comparison experiment is detailed in Section 5.5.

5. Experiments

We carry out three experiments on the effectiveness and efficiency of our keyphrase extraction system, 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 two experiments for relationship establishment and the effectiveness of supervised keyphrase extraction mentioned in Section 3 and Section 4.

5.1. A case study

In this experiment, we choose two books from our archive, use *S(v)*and *B(v)* individually as the feature weight, rank candidate keyphrases, and select top 7 of the returned list (we choose 7 because of the limitation of pages here). Moreover, we use Copernic Summarizer™ as the baseline to make comparison, since it is a leading keyphrase extractor without training (actually, it may be trained before it is published). The former influential keyphrase extractor system Extractor [1] has been integrated into it.

Table 2 below is 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 are included. *S(v)* and *B(v)* slightly performs better in two aspects. First, fewer unrelated keyphrases are extracted, such as "pint" in B2. Second, candidate keyphrase identification promotes several useful keyphrases such as "Agricultural Economics" and "wheat and rice" in B1. But two problems remain in all these systems. First, simple words can be easily extracted as in B2, but the phrase is hard to get. Second, synonym mapping is needed between "grain" and "wheat and rice" in B1.

Table 3. Returned keyphrases from systems. The first row of each book is the human-assigned topical terms from Library of Congress of USA.

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

5.2 Comparison among feature weights

In this experiment, we compare *S(v)*and *B(v)* with two prevailing features: TFIDF, widely used in document relevance analysis and keyphrase extraction, such as KEA;

BM25, the most prevailing feature used in text indexing and ranking scheme in Information Retrieval. These two features are defined as

$$w(t,d) = TF(t,d) \bullet \ln \frac{|d|}{DF(t)},$$

$$w(t,d) = \frac{(k_1+1)TF(t,d)}{k_1((1-b)+b\frac{dl(d)}{avgdl})+TF(t,d)} \bullet \ln \frac{|d|-DF(t)+0.5}{DF(t)+0.5}$$

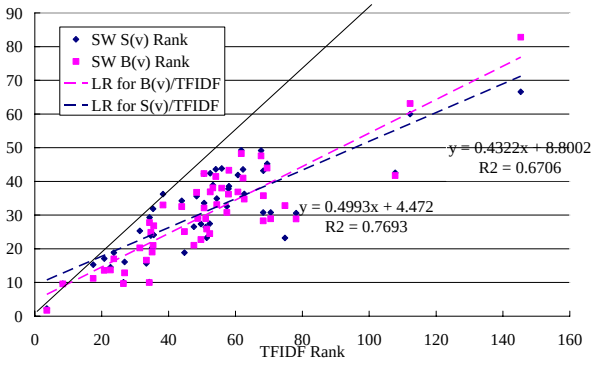
respectively, where TF stands for term frequency of term *t* in document *d*, DF for the number of documents that consist of this term, $|d|$ for the number of documents in the dataset, *dl* for the number of terms in the document, *avgdl* for the averaged *dl* in the dataset, and *b* and k_1 are two parameter need tuning. The first formula is the traditional unnormailized 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 words.

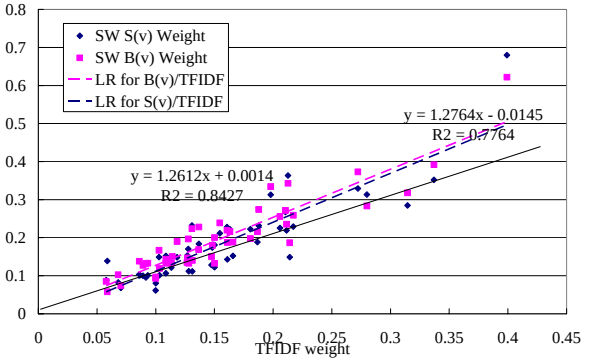
Evaluation Measure. To evaluate the improvement, we retort to five measures: the ranking of the query word in the feature set, the normalized weight of the topical term, the size of this feature set, time duration, and miss rate of target keyphrases. Though precision and recall are widely used in pattern recognition, 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 $f(v_i)$ is the original feature weight for term v_i, and $f'(v_i)$ is the normalized weight for v_i in document *d*,

$$f'(v_i) = \frac{f(v_i)}{\sqrt{\sum_{v_k \in d} f^2(v_k)}}.$$

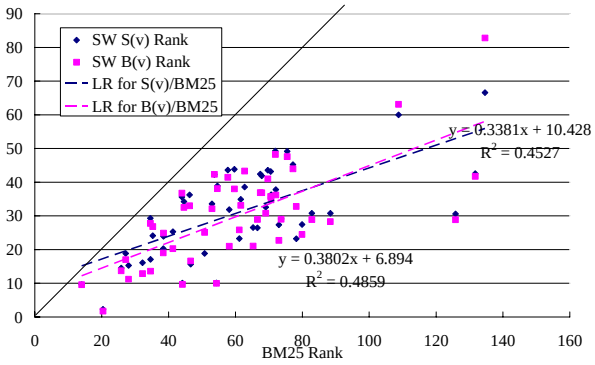
Results. Running on DELL workstation with Intel Xeon 3.0G CPU and 1.0G memory, TFIDF/BM25 finishes in 96'53'' and SW in 72'21'', about 2/3 of the previous one. SW filters nodes with only one degree in the returned set, since outskirts of the network won't have a high *S(v)* nor *B(v)*. As a result, SW has $(71.0 \pm 7.1)\%$ the size of candidate phrase set of TFIDF/BM25, at a cost of an $(6.7 \pm 4.3)\%$ 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 the samples, the result of linear regression functions shows that the performance of *S(v)* and *B(v)* are close, while they both outperforms TFIDF/BM25 with relatively higher ranks (smaller value, less than a half) and bigger weights (nearly 30% promotion from TFIDF and 80% from BM25).



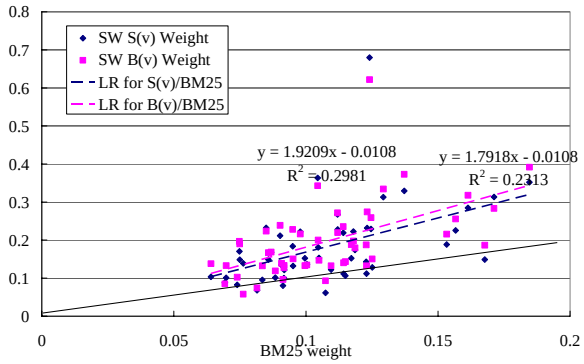
(a)



(b)



(c)



(d)

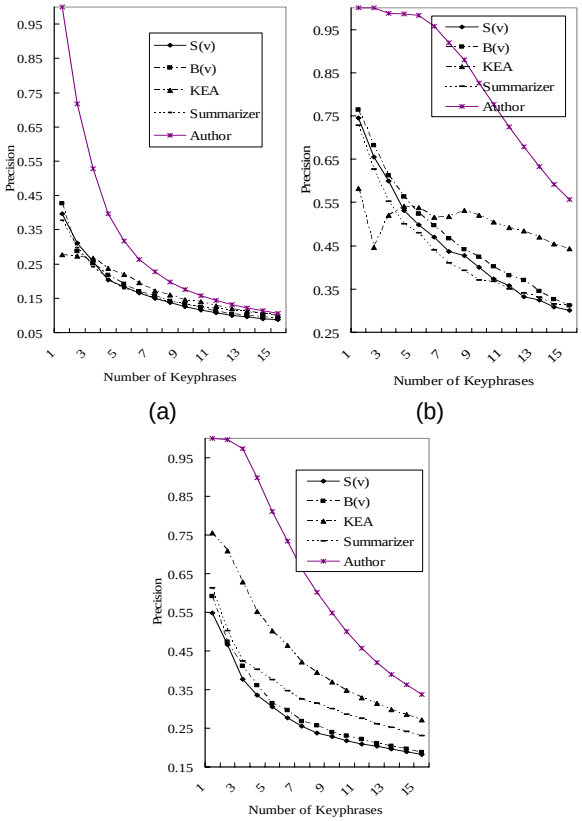
Figure 4. Results of Experiment 2. (a, c) The ranks of topical terms returned by S(v) and B(v) versus TFIDF and BM25. (b, d) The weights of topical terms returned by S(v) and B(v) versus TFIDF and BM25. The x-axis is values returned by TFIDF and y-axis is S(v) in blue and B(v) in red. LR stands for Linear Regression result of these data. The solid skew line is for baseline y=x.

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 eBook 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 context files and keys assigned by author. And 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 words. 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 the similar measures, the average precision within top n keyphrases in a dataset, denoted P@n, taking account of accuracy and a rough measure on 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.



(a) (b)

(c)

Figure 5. 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 on 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 that the relation in the document is so sterile in short documents that most nodes with very low PF and degree make no differences in a semantic network, while in TFIDF may do (KEA uses normalized TFIDF as its only feature 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. The reason might be that it is trained on this dataset, but its performance on the first 5 keyphrases is worst. Moreover, we can witness a big precision margin between state-of-art keyphrase extraction system and optimal baseline Author.

5.4 Experiment for relationship establishment

We carry out this experiment to test the performance of extraction with different kinds of 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)*.

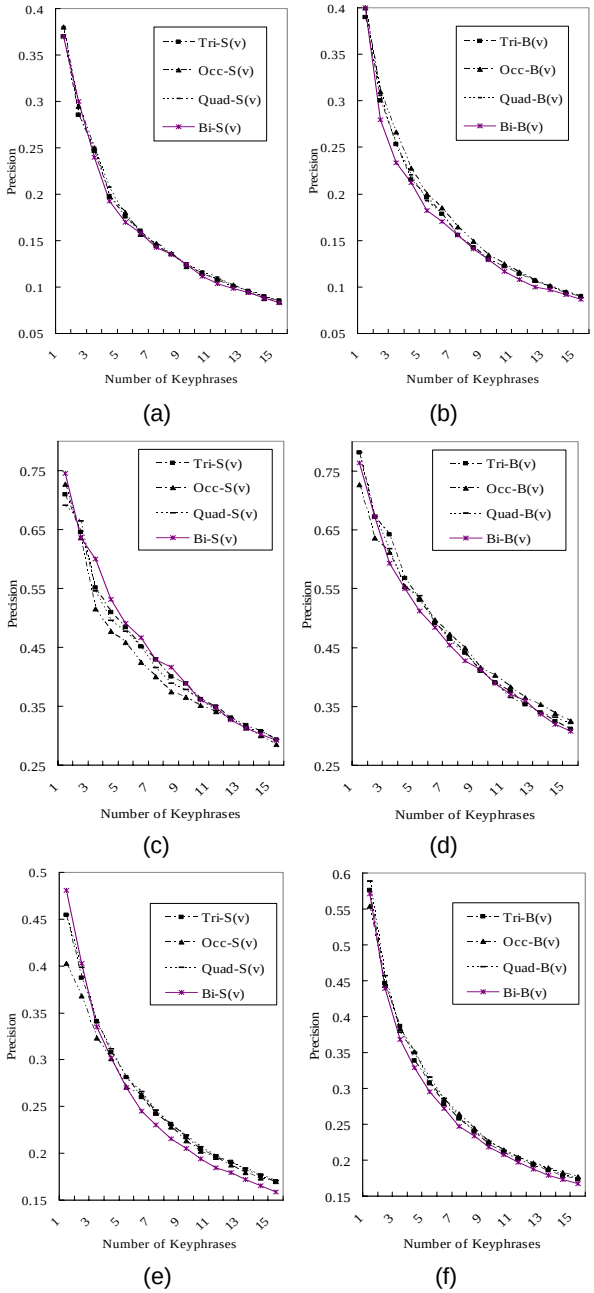


Figure 6. Precision comparison between four kinds of relation in three datasets. (a, b) In *eBook* dataset. (c, d) In *Mid* dataset. (e, f) In *Short* dataset.

Seen from the results of the experiment, 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 keyphrase 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 positive 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.

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

kernel	Unnorm Acc. (%)			Norm Acc. (%)			Train time
	Pos	Neg	All	Pos	Neg	All	
Linear	74.9	83.4	83.3	66.0	90.8	90.3	67’
L-Prob	57.1	93.9	93.2	68.1	89.6	89.2	311’
RBF	**77.0**	80.5	80.5	70.7	86.7	86.4	46’
R-Prob	**77.0**	81.3	81.2	**73.2**	86.2	86.0	181’
Sigmoid	1.57	**99.9**	**98.2**	1.05	**99.9**	**98.1**	**4’**
S-Prob	0	100	**98.2**	24.6	99.3	97.9	299’

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, but a lower in positive instances. For linear kernel, the result is reverse. Linear, RBF, and R-Prob performs comparatively though linear 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, accuracy of Neg is partially 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 [25], which drastically decrease efficiency.

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 also easy to generalize, existing unsupervised learning approach seems to be unpractical. We propose a keyphrase extraction algorithm, using two feature weights. Each of them can extracts keyphrases outperforms traditional feature weights both in effectiveness and efficiency. Moreover, we apply this algorithm to a supervised task, aiming at building a more accurate classifier that can dynamically determine the size of the returned keyphrase set. Experiments show that the overall accuracy of this supervised extraction method can be up to 80%.

Acknowledgement

This work is supported partially by the 863 International Cooperation Project of Technology Ministry of China (No. 2003AA119010), and China-US Digital Academy Library (CADAL) Project (No. CADAL2004002).

We are thankful to Mr. Peter D. Turney for generously sharing his datasets, Prof. Jinhu Lu for his insightful suggestions on the theory of Small-world Model, and several researchers of MSRA for their comments. We thank Yuanning Li and Fei Yang for helpful discussions.

References

[1] Turney, P.D. Learning Algorithms for Keyphrase Extraction. *Information Retrieval*, 2, pp. 303-336 (2000).

[2] Strogatz. S.H. Exploring Complex Networks. *Nature* 410, pp. 268–276 (2001).

[3] Dorogovtscv, S.N. and Mcndes, J.F.F. Exactly solvable analogy of small-world networks. *Euro phys.Lett.* 50, pp. 1 (2000).

[4] Zhu, M., Cai, Z., and Cai, Q. Automatic Keywords Extraction Of Chinese Document Using Small World Structure. In *Proc. of IEEE ICNLPKE'03*, 2003.

[5] i Cancho, R., Sole, R. The small world of human language. In *Proc. R. Soc. London B*, 2001.

[6] Lyon, C., Nehaniv, C., and Dickerson, B. Entropy Indicators for Investigating Early Language Process. In *AISB'05: Proceedings of EELC'05*, pp. 143–150.

[7] Caldeira, S., Lobao, T., et al. The Network of Concepts in Written Texts. *Eur. Phys. J.* B 49, pp. 523–529 (2006).

[8] Watts, D. and Strogatz, S., Collective dynamics of small-world networks, *Nature* 393, pp. 440 (1998).

[9] Latora, V., Marchiori, M. Efficient Behavior of Small-World Networks. *Phys. Rev. Lett.,* 87 (2001), art. No. 198701.

[10] Sigman, M. and Cecchi, G. Global organization of the Wordnet lexicon. *PNAS*, USA, 99 (2002), pp. 1742–1747.

[11] Newman, M. The structure and function of networks, *Comput. Phys. Comm*., 147(2002), pp. 40–45.

[12] Porter, M. The Porter Stemming Algorithm. (2005) (http://www.tartarus.org/~martin/PorterStemmer)

[13] Steyvers, M. and Tenenbaum,J. The Large-Scale Structure of semantic networks: Statistical Analyses and a Model for Semantic Growth, *Cognitive Science,* 29(1, 2005), pp. 41-78

[14] Humphreys, J. PhraseRate: An HTML Keyphrase Extractor. Technical report, University of California, Riverside. June 2002. http://infomine.ucr.edu/

[15] Forman, G. Extensive empirical study of feature selection metrics for text classification. *J. of Machine Learning Research*, 3 (2003) pp. 1289-1305, MIT Press.

[16] Yang, Y. and Pedersen, J.O. A Comparative Study on Feature Selection in Text Categorization. In *Proc. of the ICML'97*, pp. 412–420, 1997

[17] Giuffrida, G., Shek, E., and Yang, J. Knowledge-based metadata extraction from PostScript files. In *Proc. of Fifth ACM Conference on Digital Libraries*, 2000.

[18] Witten, I.H. et al. KEA: practical automatic keyphrase extraction. In *Proc. of Fourth ACM Conference on Digital Libraries*, 1999.

[19] Mladenic, D. and Grobelnik, M. Word sequences as features in text. In *Proc. of ERK'98*, 1998.

[20] Caropreso, M.F., et al. A Leaner-Independent Evaluation of the Usefulness of Statistical Phrase for Automated Text Categorization. *Text Databases and Document Management: Theory and Practice*, A.G.Chin, ed. Idea Group Publishing, Hershey, PA, pp.78–102, 2001.

[21] Fortunato, S., et al. A Method to Find Community Structures Based on Information Centrality. *Phy. Rev. E.* (2004).

[22] Robertson, S., et al. Simple BM25 extension to multiple weighted fields. In *Proc. of the CIKM'04*, pp. 42–49, 2004.

[23] Mihalcea, R. and Tarau P. TextRank: Bringing Order into Texts. In *Proc.of EMNLP 2004*, Barcelona, Spain, July 2004.

[24] Tomokiyo, T. and Hurst M. A language model approach to keyphrase extraction. In *Proc.of the ACL Workshop on Multiword Expressions*, 2003.

[25] Chang, C. and Lin, C. LIBSVM: a library for support vector machines, 2006. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm