Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

Integrating Semantic Relatedness and Words' Intrinsic Features for Keyword Extraction

Wei Zhang, Wei Feng, Jianyong Wang
Department of Computer Sci. and Tech.,
Tsinghua University, Beijing, China
{zwei11, feng-w10}@mails.tsinghua.edu.cn, jianyong@tsinghua.edu.cn

Abstract

Keyword extraction attracts much attention for its significant role in various natural language processing tasks. While some existing methods for keyword extraction have considered using single type of semantic relatedness between words or inherent attributes of words, almost all of them ignore two important issues: 1) how to fuse multiple types of semantic relations between words into a uniform semantic measurement and automatically learn the weights of the edges between the words in the word graph of each document, and 2) how to integrate the relations between words and words' intrinsic features into a unified model. In this work, we tackle the two issues based on the supervised random walk model. We propose a supervised ranking based method for keyword extraction, which is called SEAFARER[1]. It can not only automatically learn the weights of the edges in the unified graph of each document which includes multiple semantic relations but also combine the merits of semantic relations of edges and intrinsic attributes of nodes together. We conducted extensive experimental study on an established benchmark and the experimental results demonstrate that SEAFARER outperforms the state-of-the-art supervised and unsupervised methods.

1 Introduction

Keyword extraction, including extracting keyterms and keyphrases, is popularly used to summarize the core ideas of the original texts. It is beneficial for both users to find text information through keyword search and information publishers to organize content through keywords' taxonomy.

Existing methods for the keyword extraction task are generally classified into two categories: supervised and unsupervised approaches [Liu *et al.*, 2010]. In short, these two approaches focus on different aspects of the task. Generally, the supervised methods [Turney, 2000] take full advantage of words' own attributes, which we regard as node features. They first convert each text unit in training data into a feature vector and then learn to classify or rank candidate keywords. Finally, the trained models can output the labels or ranking scores for candidate keywords. On the other hand, the unsupervised methods [Mihalcea and Tarau, 2004; Grineva *et al.*, 2009] concentrate on the relationships between the words in the word graphs, which we call edge features in this paper. Normally, they first represent the original text documents as directed or undirected graphs based on the relatedness between text units and then utilize the clustering methods or random walk based methods to determine the final keywords.

However, based on our observations, most existing unsupervised methods consider only one type of relationship between the text units in one graph. More specifically, for some works such as [Mihalcea and Tarau, 2004], they compute relatedness based on the local context of the text units in their documents. While some other works [Grineva *et al.*, 2009] measure the relationships based on an external knowledge base like Wikipedia[2]. As these information reveals the relatedness between text units from different levels, it is promising to consider these relations together in constructing graphs. Moreover, one major limitation for both supervised methods and unsupervised methods is that they exploit node features and edge features separately, which lacks a general way to integrate the merits of the two types of features.

Thus, two important issues naturally arise for the keyword extraction task: 1) how to fuse multiple semantic relations into a unified relatedness measurement and automatically learn the weights of the edges in the word graphs, and 2) how to integrate the semantic relations between words and words' intrinsic attributes into a unified model.

To address the above issues, we extend the supervised random walk model [Backstrom and Leskovec, 2011] to keyword extraction task by combining it with a ranking based framework. The proposed method is called SEAFARER. Based on supervision information and optimization principles, SEAFARER is able to learn the weights of the edges between words based on multiple types of relationships. And in particular, we consider three types of relations between words in this paper. These relations come from three text

[1] SEAFARER stands for combining Semantic rElAtedness and words' intrinsic FeAtuRes for kEyword extRaction

[2] http://www.wikipedia.org/

levels, i.e, document level, corpus level, and knowledge base level. Moreover, SEAFARER seamlessly combines the edge features and node features due to its essential advantage of joint optimizing the parameters of the nodes and edges in the word graphs. We conducted experiments on a benchmark dataset and the results demonstrate that automatic edge learning by fusing multiple semantic relations achieves better performance than using them separately. Moreover, considering edge and node features together further improves the keyword extraction results. To the best of our knowledge, this is the first work to tackle the two issues together and combine them in a unified model.

2 Related Work

Unsupervised Methods. Existing unsupervised methods for keyword extraction can be roughly classified into two categories, i.e., random walk based methods [Mihalcea and Tarau, 2004; Wan and Xiao, 2008; Liu *et al.*, 2010] and clustering based methods [Grineva *et al.*, 2009; Liu *et al.*, 2009]. For the random walk based methods, stable ranking scores for each node in the word graphs can be obtained after several random walk iterations. [Mihalcea and Tarau, 2004] first applied random walk method to keyword extraction task. More recently, the authors in [Liu *et al.*, 2010] proposed a biased random walk method which computes the degree of topic matching between a document and the words within it to measure the prior score of each candidate keyterm. The authors in [Zhao *et al.*, 2011] also proposed a context-sensitive topical pagerank method for extracting keywords in microblogs. For the clustering based methods, they often divide the words into clusters and then choose the representative words in each cluster and finally merge them into keywords [Liu *et al.*, 2009; Grineva *et al.*, 2009].

Unlike most previous unsupervised methods which merely utilize one type of relation between words and directly regard the values of the relation as the weights of the edges, we can automatically learn the weights in a unified graph. In Section 5, it is experimentally validated to be useful to improve keyword extraction results.

Supervised Methods. Supervised methods for keyword extraction usually need labeled training data to optimize the model parameters. Existing supervised methods [Turney, 2000; Hulth, 2003; Jiang *et al.*, 2009; Li *et al.*, 2010] mainly focus on two aspects of the task: (1) which model is more effective, and (2) which word features are useful for the task. While works like [Turney, 2000] apply simple supervised methods for keyword extraction, two recent works [Jiang *et al.*, 2009; Li *et al.*, 2010] show that boosting and learning to rank based machine learning methods, e.g., GBM [Friedman, 2000] and Ranking SVM [Joachims, 2006], obtain better results than traditional methods for keyword extraction. On the other hand, the authors in [Hofmann *et al.*, 2009] considered document structure features for words in addition to the traditionally used features, such as term frequency, word position, etc. Besides, the authors in [Xu *et al.*, 2010] exploited the category and infobox information in Wikipedia articles to derive novel word features. These new features can be regraded as complements to traditional word features.

In this paper, we argue that most previous supervised methods for keyword extraction task ignore the semantic relatedness between words, which has been widely used in the unsupervised methods introduced before. Thus we adopt supervised random walk to integrate both types of information to achieve better results.

3 Preliminaries

3.1 Problem Formulation

In this paper, we concentrate on extracting the keywords from text documents. We regard keywords as the union of keyphrases and keyterms. A keyterm is a single word which appears in at least one keyphrase in its corresponding document. Keyterm extraction is an important step for keyphrase extraction, which will be discussed later. Let $D = \{d_1, d_2, \ldots, d_n\}$ be a set of text documents and each document d_i includes a set of words $W = \{w_{i,1}, w_{i,2}, \ldots, w_{i,m}\}$ and several keyphrases $P = \{p_{i,1}, p_{i,2}, \ldots, p_{i,k}\}$ which consist of consecutive words. The words in P form the keyterm set $T = \{t_{i,1}, t_{i,2}, \ldots, t_{i,u}\}$. Then the goal of keyword extraction is to automatically extract T and P in a given document d_i accurately.

3.2 Ranking Framework for Keyword Extraction

Ranking based framework for keyword extraction is popular in random walk based methods [Mihalcea and Tarau, 2004; Wan and Xiao, 2008; Liu *et al.*, 2010; Zhao *et al.*, 2011]. The supervised methods usually need to construct n-grams as candidate keywords before training. However, the determination of the maximal length of candidate keywords is not easy. This is because if the length of candidate keywords is constrained to be small, some long keywords will be missed. However, merely increasing the maximal length may cause the excessive growth of training set size. Ranking based framework is more flexible because it is simple and can extract long keyphrases based on the ranking scores of keyterms.

Unlike the unsupervised ranking based framework for keyword extraction, our supervised ranking based framework incorporates the model learning step. In this work, the ranking framework is defined to have the following five steps:

(1) Candidate keyterm selection. Considering all words in each document to construct a graph may suffer from low efficiency. As most keywords are noun phrases and they commonly consist of nouns and adjectives [Hulth, 2003; Liu *et al.*, 2009], we choose all nouns and adjectives as candidate keyterms.

(2) Semantic graph construction. Different from previous works in which the weights of edges are determined by simply choosing one type of relation in advance, we consider the edges as functions about multiple relations, whose weights can be automatically learned by considering each type of relation as a feature and optimizing the parameters for each feature. Besides, in our constructed graph, each node has its own attributes, which can be regarded as node features and used to compute the prior ranking score for each word.

(3) Model Learning. This step concentrates on learning the optimal model parameters on training data. We regard keyterms as positive nodes and the other words in the same

graph as negative nodes. The optimization goal is to rank the positive nodes higher than the negative nodes. More information about the model parameters and its learning process can be found in Section 4.

(4) Keyterm ranking. After getting the optimal model parameters, we can compute the weights of edges for test data through multiple relations between words and the prior ranking score for each word based on its intrinsic attributes. Then we run standard random walk to get the final ranking score for each candidate keyterm.

(5) Keyphrase extraction. Candidate keyphrases consist of consecutive candidate keyterms. The ranking scores of candidate keyphrases are computed based on the simple strategy adopted in [Wan and Xiao, 2008; Liu *et al.*, 2010; Zhao *et al.*, 2011], which can be defined as follows,

$$S(p) = \sum_{t \in p} S(t) \tag{1}$$

where $S(t)$ represents the score of keyterm t. Then we can choose top-k candidate keyphrases as the final results.

Actually, our main contributions lie in the Step 2 to Step 4. Hence, we concentrate more on keyterm extraction results, which can indeed reveal the effectiveness of our method.

3.3 Supervised Random Walk

Supervised random walk was proposed in [Backstrom and Leskovec, 2011], which has been exploited for predicting links' occurrence in social networks. In short, it incorporates supervision information into standard unsupervised random walk model. The random walk model is defined as follows,

$$\mathbf{S}^{t+1} = \alpha * \overline{\mathbf{A}}\mathbf{S}^t + (1-\alpha) * \overline{\mathbf{P}} \tag{2}$$

where $\mathbf{S}$ denotes the ranking scores for all the nodes in the graph, $\overline{\mathbf{A}}$ indicates the adjacency matrix of the graph with each of its columns normalized to sum to 1. $\overline{\mathbf{P}}$ represents the prior ranking scores for all the nodes, which are usually initialized to be equal. To assign higher prior scores to the nodes which are more likely to be preferred, [Haveliwala, 2002] proposed a topic-sensitive pagerank which can be regarded as a biased random walk.

While the weights of edges are computed before constructing graphs in standard random walk based methods, in supervised random walk based approaches the weights are determined by the feature functions of edges. For each edge $a_{i,j}$, its weight can be expressed as below,

$$\varphi(\mathbf{x}_{i,j}) = \frac{1}{1 + exp(-\boldsymbol{\omega}^T \mathbf{x}_{i,j})} \tag{3}$$

where $\boldsymbol{\omega}$ denotes the model parameters and $\mathbf{x}_{i,j}$ represents the related features of the edge $a_{i,j}$ in the social network. All $\varphi(\mathbf{x}_{i,j})$ should be normalized to sum to 1 in each column j. Recently, a biased supervised random walk was also proposed in [Feng and Wang, 2012].

Denote the positive node set by I_p and negative node set by I_n. To optimize the model parameters $\boldsymbol{\omega}$, supervised random walk attempts to make the ranking score S_i for $i \in I_p$ larger than S_j for $j \in I_n$. This is similar to maximizing the AUC (Area Under the Roc Curve), which is defined as follows,

$$AUC = \frac{\sum_{i \in I_p} \sum_{j \in I_n} \mathbb{I}(S_i - S_j)}{|I_p||I_n|} \tag{4}$$

where $\mathbb{I}(.)$ is an indicator function that is equal to 1 if $S_i > S_j$. As the indicator function is non-differential and many optimization algorithms cannot be directly applied to it, it is usually replaced by other differentiable functions such as the sigmoid function. More details about the parameter optimization will be introduced in Section 4.

4 Methods

4.1 Fusion of Multiple Relations

In this work, we consider three types of semantic relations which come from three levels: document, corpus and knowledge base.

Document. Words' relations derived from a document are calculated based on the number of words' co-occurrence times, which is able to provide local syntactic relatedness between words. To be specific, each document can be viewed as a long word sequence and words' co-occurrence information can be obtained through sliding windows. The length of the sliding window is usually set to 2-10 [Mihalcea and Tarau, 2004]. The more times the two words co-occur in sliding windows, the higher their relatedness is in the word graphs.

Corpus. A corpus is a set of documents which usually belong to similar domains. Based on the effectiveness of the topic model [Blei *et al.*, 2003], we can capture the topic similarity between words. In this work, we first get the topic distribution $T_w = \{t_{w,1}, t_{w,2}, \ldots, t_{w,k}\}$ for each word w in the corpus through GibbsLDA++[3]. Then we compute the topic similarity between words w_i and w_j through cosine similarity, which is defined as follows,

$$sim(w_i, w_j) = \frac{{T_w}^i \cdot {T_w}^j}{||{T_w}^i|| \; ||{T_w}^j||} \tag{5}$$

Empirically, the number of topics is set to 50 after a few attempts.

Knowledge base. Knowledge bases such as Wikipedia or YAGO provide content information as well as linkage information between content units. These link information holds a wealth of semantic information. Wikipedia contains more than four million articles and provides larger coverage to words than other knowledge bases. In order to utilize the linkage information in Wikipedia, we exploit the toolkit, Wikipedia-Miner[4]. Given two words w_i and w_j, the semantic relatedness between them can be computed as follows,

$$sim(w_i, w_j) = 1 - \frac{log(max(|E_i|, |E_j|)) - log(|E_i| \bigcap |E_j|)}{log(|W|) - log(min(|E_i|, |E_j|))} \tag{6}$$

where E_i is the set of documents connected to w_i and $|W|$ is the total number of documents.

[3] http://gibbslda.sourceforge.net/

[4] http://wikipedia-miner.sourceforge.net/index.htm

Finally, we get three levels' relatedness between words. We regard these relations as edge features. Based on Equation 3, we can linearly combine the edge features and get the weights of edges though the sigmoid function just the same as supervised random walk. Thus, we get a unified graph for each document d_i. Finally, given the document d_i, the equation of supervised random walk which fuses multiple relations is defined as below,

$$\mathbf{S}_i^{t+1} = \alpha * \overline{\mathbf{A}}_i(\boldsymbol{\varphi}_{e,i})\mathbf{S}_i^t + (1-\alpha) * \overline{\mathbf{P}}_i \tag{7}$$

where $\overline{\mathbf{A}}_i(\boldsymbol{\varphi}_{e,i})$ denotes the transition matrix for document d_i which is a function matrix here. Its elements are edge functions in the graph for each document, and $\boldsymbol{\varphi}_{e,i}$ denotes all the edge functions in the graph of d_i. The edge functions are defined the same as Equation 3.

4.2 Integration of Edge and Node Features

In this paper, we adopt five common node features that have been widely used in this task. They are: term frequency (TF), document frequency (DF), term frequency-inverse document frequency (TF-IDF), position of term (POS), and length of term (LEN). However, we should be aware that our method is flexible to incorporate arbitrary node and edge features.

We also utilize sigmoid function to combine these features for each word, which is defined as follows,

$$\psi(\mathbf{z}_{i,j}) = \frac{1}{1 + exp(-\boldsymbol{\omega}^T \mathbf{z}_{i,j})} \tag{8}$$

where $\mathbf{z}_{i,j}$ denotes the features of word $w_{i,j}$ in document d_i. Based on the idea of biased random walk, we incorporate the feature functions to the second term of Equation 7. To maintain the ranking scores of all words in each document to sum to 1, we should also normalize the values of node feature functions. At last, given the document d_i, we give the formulation of the whole model by combining with the fusion of multiple semantic relations, which is defined as follows,

$$\mathbf{S}_i^{t+1} = \alpha * \overline{\mathbf{A}}_i(\boldsymbol{\varphi}_{e,i})\mathbf{S}_i^t + (1-\alpha) * \overline{\mathbf{P}}_i(\boldsymbol{\psi}_{n,i}) \tag{9}$$

where $\overline{\mathbf{P}}_i(\boldsymbol{\psi}_{n,i})$ denotes a vector whose elements are node feature functions of the word graph of d_i. This is the final formulation we adopt in our method SEAFARER.

4.3 Parameter Learning

We now discuss the method to get the optimal model parameters $\boldsymbol{\omega}_e$ and $\boldsymbol{\omega}_n$, which denote edges' and nodes' related parameters respectively. We adopt the gradient ascent method to seek the maximal value of objective function ϕ with similar form to Equation 4 except the indicator function is replaced by the sigmoid function. The key to the gradient ascent method is calculating the first-order derivative of the objective function, which is defined as follows,

$$\frac{\partial \phi(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = \sum_{i \in D} \frac{\sum_{j \in I_{i,p} \wedge k \in I_{i,n}} \frac{\sigma(dif_{(i,j,k)})}{dif_{(i,j,k)}} \left(\frac{\partial S_{i,j}}{\partial \boldsymbol{\omega}} - \frac{\partial S_{i,k}}{\partial \boldsymbol{\omega}} \right)}{|I_{i,p}||I_{i,n}|} \tag{10}$$

where $I_{i,p}$ denotes the keyterm set and $S_{i,j}$ is the ranking score of keyterm t_j in the document d_i. $\boldsymbol{\omega}$ represents model parameters which include $\boldsymbol{\omega}_e$ and $\boldsymbol{\omega}_n$. $dif_{(i,j,k)}$ is set to be equal to $S_{i,j} - S_{i,k}$. σ represents the sigmoid function. Note that the above expression usually should add a 2-norm regularization term to avoid overfitting.

Then the problem is to seek $\frac{\partial S_{i,\cdot}}{\partial \boldsymbol{\omega}}$. Based on Equation 9, we get the following two expressions for each document d_i,

$$\frac{\partial \mathbf{S}_i}{\partial \boldsymbol{\omega}_e} = \alpha * \left(\overline{\mathbf{A}}_i(\boldsymbol{\varphi}_{e,i}) \frac{\partial \mathbf{S}_i}{\partial \boldsymbol{\omega}_e} + \frac{\partial \overline{\mathbf{A}}_i(\boldsymbol{\varphi}_{e,i})}{\partial \boldsymbol{\omega}_e} \mathbf{S}_i \right) \tag{11}$$

$$\frac{\partial \mathbf{S}_i}{\partial \boldsymbol{\omega}_n} = \alpha * \overline{\mathbf{A}}_i(\boldsymbol{\varphi}_{e,i}) \frac{\partial \mathbf{S}_i}{\partial \boldsymbol{\omega}_n} + (1-\alpha) * \frac{\partial \overline{\mathbf{P}}_i(\boldsymbol{\psi}_{n,i})}{\partial \boldsymbol{\omega}_n} \tag{12}$$

It is obvious to see that the above two equations have similar forms to the pagerank expression. Based on the algorithm proposed in [Backstrom and Leskovec, 2011], we can iteratively compute $\frac{\partial S_i}{\partial \boldsymbol{\omega}}$ to get the final stable results.

5 Experimental Study

5.1 Dataset and Preprocessing

To evaluate the performance of our method and compare with other state-of-the-art methods for keyword extraction, we chose a benchmark dataset [Hulth, 2003] which has been widely used in other related works [Mihalcea and Tarau, 2004; Liu *et al.*, 2009; 2010]. This dataset consists of 2000 abstracts and their corresponding annotated keywords, which have been already divided into training, validation and test set. Thus it is reliable to compare our results with some already published results on this dataset. Because annotators scanned the whole text to choose the keywords, there are some keywords not occurring in the abstracts and titles, which means the maximal coverage of the extraction results cannot reach to 100%. We did not remove these words when computing the evaluation results.

For data preprocessing, we first removed stopwords[5] from the raw text, then we used Stanford Log-linear Part-Of-Speech Tagger[6] to assign parts of speech to each word. Finally, we stemmed all the saved words by Porter's stemmer[7].

5.2 Evaluation Metrics

Based on our observations, almost all previous works on keyword extraction adopt precision, recall and F1-measure to evaluate the results. Hence, we keep our evaluation metric consistent. Besides, we tune the hyperparameters of our method based on AUC metric on the validation dataset. The results can also reveal the effect of combining edge and node features for keyword extraction.

5.3 Parameter Setting

Damping factor α. In our method, α controls the probability of random jumping from the source node to a random node in the graph. It is the hyperparameter which we tune on the validation set based on AUC metric. As the results showed

[5] http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

[6] http://nlp.stanford.edu/software/tagger.shtml

[7] http://tartarus.org/martin/PorterStemmer/

Method	Top-5			Top-10			Top-15		
	Prec	Recall	F-score	Prec	Recall	F-score	Prec	Recall	F-score
BRW	0.709	0.182	0.289	0.592	0.303	0.401	0.517	0.394	0.447
WRW	0.552	0.142	0.225	0.524	0.269	0.355	0.492	0.375	0.426
LR	0.727	0.186	0.297	0.620	0.318	0.420	0.548	0.417	0.474
RankSVM	0.731	0.187	0.298	0.617	0.316	0.418	0.547	0.417	0.473
ListNet	0.732	0.187	0.298	0.616	0.315	0.417	0.543	0.413	0.469
GBM	0.759	0.194	0.309	0.638	0.326	0.431	0.557	0.424	0.481
SEAFARER	**0.768**	**0.197**	**0.313**	**0.654**	**0.335**	**0.443**	**0.574**	**0.437**	**0.496**

Table 1: Comparison of different methods on keyterm extraction on the Hulth's data.

in Figure 1, α gains better performance in the region of 0.7 to 0.9 and get optimal AUC at 0.8. Therefore, we chose $\alpha = 0.8$ without dedicated tuning. Besides, when α is set to be 0 or 1, the corresponding methods are the special cases of our method for only considering node and edge features separately. As their results are explicitly worse, it indicates the advantage of combining both node and edge features.

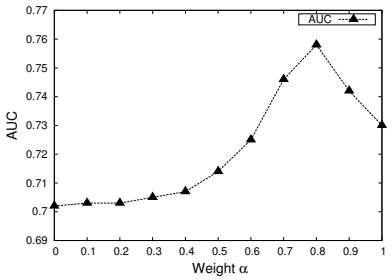

Figure 1: The effect of damping factor α

Learning rate η. In learning process, η determines the convergence speed of parameters. A large η may speed up the training process but it may also cause failures to converge. We chose 10^{-4} as a tradeoff.

Regularization parameter λ. For simplicity, we set $\lambda = 1$, the same as [Backstrom and Leskovec, 2011].

5.4 Baselines

In order to verify the superiority of our method, we compare our method with some supervised and unsupervised methods, which are listed below:

Basic Random Walk (BRW). It constructs the word graphs based on the co-occurrence information in local documents and then run standard random walk model on it to get nodes' ranking scores.

Wiki-based Random Walk (WRW). This is similar to the previous one except that the graphs are constructed based on the linkage information from Wikipedia.

Yahoo! Terms Extractor. It is an implementation of keyword extraction from industry which can be accessed through the web service[8]. As it directly returns the extracted keyphrases without ranking scores, we only compare with it in the last experiment.

Logistic Regression (LR). Logistic regression is a popular supervised approach for its simplicity and effectiveness in many problems.

RankSVM. RankSVM[9] is a state-of-the-art pairwise learning to rank approach which has been experimentally demonstrated to gain better performance than SVM and some other methods in keyword extraction task [Jiang *et al.*, 2009].

ListNet. ListNet [Cao *et al.*, 2007] is a starte-of-the-art listwise learning to rank approach which sometimes performs better than RankSVM.

Gradient Boosting Machine (GBM). Gradient Boosting Machine [Friedman, 2000] is an ensemble learning based approach, which has been shown to perform best in keyword extraction among the methods adopted in [Li *et al.*, 2010].

In addition to the above introduced methods, we also compare our results with some previous works' results on the same dataset, including Bagging in [Hulth, 2003], TextRank in [Mihalcea and Tarau, 2004], clustering-based method (SC) in [Liu *et al.*, 2009], Topical PageRank (TPR) in [Liu *et al.*, 2010]. All the results will be discussed later.

5.5 Experimental Results

(A) Evaluation of relations' fusion. We first show the advantage of fusing multiple relations with automatic learning the weights of edges based on optimization principles. We compare our fusion method with three versions of random walk method, which are different in their used graphs. We use "doc", "corp", and "wiki" to represent the random walk method using graphs at the document, corpus, and Wikipedia levels, respectively. We compare them on keyterm extraction and the results are shown in Table 2.

Relation	Top-5		Top-10		Top-15	
	Prec	Recall	Prec	Recall	Prec	Recall
doc	0.709	0.182	0.592	0.303	0.517	0.394
corp	0.444	0.114	0.434	0.223	0.425	0.324
wiki	0.552	0.142	0.524	0.269	0.492	0.375
fusion	**0.738**	**0.189**	**0.621**	**0.318**	**0.546**	**0.416**

Table 2: Evaluation of the effectiveness of multiple relations' fusion.

The results show that the fusion of several relations and automatic learning of the edge weights indeed outperforms the other three methods. This is consistent with our intuition that the automatic learned weights of edges based on multiple semantic relations can reflect the relationship between words more accurately. Besides, random walk using document level

[8] http://developer.yahoo.com/search/content/V1/termExtraction.html

[9] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

information and knowledge base information obtain better results than using corpus level information. One possible reason is that when computing two words' similarity based on their topic distribution, we ignore their documents' topic distributions, which are important as keywords' topic distributions should be similar with their documents'.

(B) Evaluation of keyterm ranking results. Since keyterm ranking is the essential step for keyphrase ranking and our method is directly targeted to improve the keyterm ranking results. This part of the experimental results can indeed reflect whether our method works.

We conducted experiments on top 5, top 10 and top 15 extracted keyterms. All the results are presented in Table 1. As the results show, SEAFARER get the best results in all metrics which indicates that our method indeed outperforms other baselines. In conjunction with the results shown in Figure 1, we can clearly see the advantage of combining both edge and node features. Among the baselines, GBM performs better than other methods, which is consistent with [Li *et al.*, 2010].

(C) Evaluation of keyphrase ranking results. Based on the ranking scores of candidate keyterms, we can get the ranking scores of keyphrases based on Equation 1. For non-random walk based methods, such as Logistic Regression, RankSVM, ListNet and GBM, we tried two strategies to deal with the raw scores of candidate keyterms. The first one is to normalize their ranking scores in each document to sum to 1 before getting the ranking scores for candidate keyphrases. While the second strategy is to directly use the ranking scores to get the candidate keyphrases' final score. At last, we use the better experimental results to represent their effectiveness. We chose top-15 returned phrases as candidates because the results show it can achieve high recall while retaining relative high precision for all the methods. The results are shown in Table 3,

Method	Prec	Recall	F-score
BRW	0.315	0.457	0.373
WRW	0.325	0.475	0.385
LR	0.329	0.478	0.390
RankSVM	0.314	0.455	0.372
ListNet	0.330	0.478	0.391
GBM	0.329	0.477	0.390
SEAFARER	**0.335**	**0.485**	**0.396**

Table 3: Comparison of different methods on keyphrase extraction of the top 15 candidate keyphrases.

The above results lead to the following observations: (1) the most popular supervised methods achieve a little better performance than the unsupervised methods. However, the performance gap w.r.t. keyphrase extraction is smaller than the keyterm extraction results; and (2) integrating edge and node features in the word graphs can also help keyphrase extraction task which makes an improvement over the-state-of-art methods for keyphrase extraction, such as GBM and ListNet.

(D) Comparison with past works' results. As we introduced in Section 5.1, the test data set is the same for most works testing on Hulth's dataset. Thus, we tried to get the extraction results of keyphrases in the similar settings to previous works and then compare our results with those published results mentioned in Section 5.4 fairly. The comparisons are listed in Table 4.

Method	Assigned	Correct	Precision
Yahoo!'s	6,312	1,407	0.223
Hulth's	7,815	1,973	0.252
TextRank	6,784	2,116	0.312
SC	7,158	2,505	0.350
SEAFARER	7,144	**2,514**	**0.352**

Method	Top5-Prec	Top5-Recall	Top5-Fscore
TPR	0.354	0.183	0.242
SEAFARER	**0.412**	**0.210**	**0.278**

Table 4: Comparison with some previous works' results on Hulth's data set.

To match the published results of those methods, the comparisons consist of two parts. The first part ensures that all methods have similar number of extracted candidate keyphrases. While the second part evaluates the results of our method on the same metric as TPR adopted. The results show that our method performs best in the two parts of the experiments. Note that our method achieves a little improvement over the method SC. However, based on the authors' observations on the test data, SC exploited an additional frequent word list to remove the terms which are too common to be keyphrases in postprocessing process. This may notably improve their final results. As this process involves manual intervention, we argue that our method indeed gain better performance than theirs. In sum, all the comparisons demonstrate the superiority of automatic learning the edge weights and integrating the edge and node features for the keyword extraction task.

6 Conclusion

In this paper, based on the supervised random walk model, we solve two important issues in the keyword extraction task which are ignored by most previous works. First, by regarding each type of semantic relations as edge features, we utilize the sigmoid function to fuse them into a uniform semantic measurement, which is then used in the word graph construction. Second, inspired by the idea of biased random walk, we use node features to model the prior ranking score for each word. Through optimization, we can automatically learn the weights of the edges in the word graphs and seamlessly integrate the edge and node features together for the task. Finally, we have conducted extensive experiments on a standard benchmark dataset and all the results demonstrate the advantage of our method.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China under Grant No. 61272088 and National Basic Research Program of China (973 Program) under Grant No. 2011CB302206.

References

[Backstrom and Leskovec, 2011] Lars Backstrom and Jure Leskovec. Supervised random walks: predicting and recommending links in social networks. In *WSDM*, pages 635–644, 2011.

[Blei *et al.*, 2003] David M. Blei, Andrew Y. NG, and Michael I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, March 2003.

[Cao *et al.*, 2007] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *ICML*, pages 129–136, 2007.

[Feng and Wang, 2012] Wei Feng and Jianyong Wang. Incorporating heterogeneous information for personalized tag recommendation in social tagging systems. In *KDD*, pages 1276–1284, 2012.

[Friedman, 2000] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[Grineva *et al.*, 2009] Maria P. Grineva, Maxim N. Grinev, and Dmitry Lizorkin. Extracting key terms from noisy and multitheme documents. In *WWW*, pages 661–670, 2009.

[Haveliwala, 2002] Taher H. Haveliwala. Topic-sensitive pagerank. In *WWW*, pages 517–526, 2002.

[Hofmann *et al.*, 2009] Katja Hofmann, Manos Tsagkias, Edgar Meij, and Maarten de Rijke. The impact of document structure on keyphrase extraction. In *CIKM*, pages 1725–1728, 2009.

[Hulth, 2003] Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *EMNLP*, pages 216–223, 2003.

[Jiang *et al.*, 2009] Xin Jiang, Yunhua Hu, and Hang Li. A ranking approach to keyphrase extraction. In *SIGIR*, pages 756–757, 2009.

[Joachims, 2006] Thorsten Joachims. Training linear svms in linear time. In *KDD*, pages 217–226, 2006.

[Li *et al.*, 2010] Zhenhui Li, Ding Zhou, Yun-Fang Juan, and Jiawei Han. Keyword extraction for social snippets. In *WWW*, pages 1143–1144, 2010.

[Liu *et al.*, 2009] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. Clustering to find exemplar terms for keyphrase extraction. In *EMNLP*, pages 257–266, 2009.

[Liu *et al.*, 2010] Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. Automatic keyphrase extraction via topic decomposition. In *EMNLP*, pages 366–376, 2010.

[Mihalcea and Tarau, 2004] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *EMNLP*, pages 404–411, 2004.

[Turney, 2000] Peter D. Turney. Learning algorithms for keyphrase extraction. *Inf. Retr.*, 2(4):303–336, May 2000.

[Wan and Xiao, 2008] Xiaojun Wan and Jianguo Xiao. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, pages 855–860, 2008.

[Xu *et al.*, 2010] Songhua Xu, Shaohui Yang, and Francis Chi-Moon Lau. Keyword extraction and headline generation using novel word features. In *AAAI*, 2010.

[Zhao *et al.*, 2011] Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming Li. Topical keyphrase extraction from twitter. In *ACL*, pages 379–388, 2011.