Front. Electr. Electron. Eng. 2012, 7(2): 242–253
DOI 10.1007/s11460-011-0174-7

RESEARCH ARTICLE

Zhiyuan LIU, Maosong SUN

Can prior knowledge help graph-based methods for keyword extraction?

Abstract Graph-based methods are one of the widely used unsupervised approaches for keyword extraction. In this approach, words are linked according to their co-occurrences within the document. Afterwards, graph-based ranking algorithms are used to rank words and those with the highest scores are selected as keywords. Although graph-based methods are effective for keyword extraction, they rank words merely based on word graph topology. In fact, we have various prior knowledge to identify how likely the words are keywords. The knowledge of words may be frequency-based, position-based, or semantic-based. In this paper, we propose to incorporate prior knowledge with graph-based methods for keyword extraction and investigate the contributions of the prior knowledge. Experiments reveal that prior knowledge can significantly improve the performance of graph-based keyword extraction. Moreover, by combining prior knowledge with neighborhood knowledge, in experiments we achieve the best results compared to previous graph-based methods.

Keywords keyword extraction, prior knowledge, PageRank, DiffusionRank

1 Introduction

There is tremendous information on the Web. Thus, it is important to effectively seek and manage information. The major difficulty for users is to find what they want from numerous documents. One alternative approach to solving the difficulty is to use keywords to represent a document. Keywords are a small set of words that can best represent a document, which can be identified by some statistical or linguistic properties, such as frequency, length, part of speech (POS), and so on. Keywords are useful in many applications of seeking and managing information for people, such as query refinement of search engine, summarization, social tagging, indexing/cataloging, feature selection for classification, and clustering.

As summary of document, a keyword can be either a single word or a phrase. Keywords in articles of journals and books are generally assigned by authors. However, most articles on the Web do not have human-assigned keywords. Therefore, automatic keyword extraction is an important research task. Existing methods for keyword extraction can roughly be divided into supervised classification approach and unsupervised ranking approach.

Supervised approach [1] regards keyword extraction as a classification task. In this approach, a model is trained to determine whether a word of the document is a keyword or not based on statistical and linguistic features. In supervised approach, a document set with human-assigned keywords is required for training. Since human labeling is usually both tedious and time-consuming, in this study, we focus on unsupervised approach.

Unsupervised ranking approach does not need manually labeled training set [2–6]. Instead, this approach regards keyword extraction as a ranking task and applies several measures to rank the words of a document and extract the top-ranked ones as keywords. Among various unsupervised methods, graph-based methods [7] are most widely used and attract many research attentions [8–11]. In graph-based methods, a document is represented as a word graph based on word relatedness, and then a graph-based ranking algorithm (e.g., PageRank [12]) is used to assign importance scores to each word.

Existing graph-based methods conduct the keyword extraction task only depending on the graph topology of document. A belief behind graph-based ranking algorithms

Received April 28, 2011; accepted August 30, 2011

Zhiyuan LIU (✉), Maosong SUN
Department of Computer Science and Technology, State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China
E-mail: lzy.thu@gmail.com

(e.g., PageRank) is that all words are born equal and have the same ability for voting and standing for election. In fact, in many other methods for keyword extraction, many useful measures of words have been introduced to identify how likely the word is a keyword. That is to say, before running graph-based ranking algorithms, we have already had prior knowledge on which words are more likely to be keywords. Thus, one important intuition is to introduce prior knowledge to help graph-based methods for keyword extraction. If it is possible, how do we incorporate the prior knowledge with graph-based method?

In this paper, we divide the prior knowledge of words into three types: frequency-based (*tfidf* values), position-based (first appearance positions of the words within the document), and semantic-based (semantic relatedness with the document). We propose to use DiffusionRank to incorporate prior knowledge. We systematically investigate the efficiency of prior knowledge in DiffusionRank for keyword extraction. Experiments show that DiffusionRank with prior knowledge outperform traditional PageRank for keyword extraction on precision, recall, and F-measure. It should also be noted that the proposed method is also unsupervised, which is applicable in Web era with enormous information.

Furthermore, in Refs. [8,9], Wan and Xiao proposed to use a small number of nearest neighbor documents of the given document to provide more knowledge to improve keyword extraction. The method with neighborhood knowledge outperforms traditional graph-based methods with more than 2.5% improvement in F-measure. In this paper, we investigate the combination of prior knowledge and neighborhood knowledge for keyword extraction. Experiment results show that the new method with combined prior knowledge and neighborhood knowledge obtains more than 2% improvement in F-measure than the method with only neighborhood knowledge.

The contributions of this paper are as follows: 1) For the first time, we raise the problem of applying prior knowledge in graph-based methods for keyword extraction and further propose a novel and practical framework to solve the problem. 2) We divide various knowledge of words into three types and systematically investigate their contributions to graph-based keyword extraction as prior knowledge. 3) By combining prior knowledge and neighborhood knowledge, we obtain the best results compared to previous graph-based keyword extraction methods in our experiments.

2 PageRank and DiffusionRank

For convenience, we first give some notations. A document is usually represented as an undirected word graph. Here, we denote $G=(V,E)$ as an undirected graph with the set of vertices $V=\{v_1,v_2,...,v_N\}$ and the set of edges $E=(v_i,v_j)$ if there is an edge between v_i and v_j. For a given vertex v_i, let $N(v_i)$ be the set of vertices that are neighbors of v_i. We have $w(v_i,v_j)$ as the weight of the edge (v_i,v_j). For an undirected graph, $w(v_i,v_j)=w(v_j,v_i)$. Let $w(v_i)$ be the degree of v_i, then we have

$$w(v_i)=\sum_{v_j\in N(v_i)} w(v_j,v_i)=\sum_{v_j\in N(v_i)} w(v_i,v_j). \quad (1)$$

2.1 PageRank

PageRank is a well-known algorithm that uses link information to assign global importance scores to all pages on the Web. The basic idea of PageRank is that a web page is important if some other important web pages point to it. PageRank is an iterative algorithm. In this paper, we focus to build word graph as an undirected graph.

In PageRank, a surfer on a given vertex v_i with probability λ chooses to select one of its neighbors and with probability $1-\lambda$ to jump to a random vertex. Here, λ is a damping factor with $0<\lambda<1$, and in practice, we usually set $\lambda=0.85$. Thus, the PageRank score $PR(p)$ of a vertex v_i is defined as

$$PR(v_i)=\lambda\sum_{v_j\in N(v_i)}\frac{w(v_j,v_i)}{w(v_j)}PR(v_j)+(1-\lambda)\frac{1}{N}, \quad (2)$$

where N is the number of vertices. Since the definition of PageRank is recursive, it must be iteratively refined until convergence. We first introduce an adjacent matrix $\boldsymbol{A}$ to represent the graph for PageRank:

$$A(i,j)=\begin{cases}0, & \text{if}(v_i,v_j)\notin E,\\ \dfrac{w(v_i,v_j)}{w(v_j)}, & \text{if}(v_i,v_j)\in E.\end{cases}$$

With the matrix, we can rewrite Eq. (2) as

$$\boldsymbol{r}=\lambda\boldsymbol{A}\boldsymbol{r}+(1-\lambda)\frac{1}{N}\mathbf{1}, \quad (3)$$

where $\boldsymbol{r}$ is the vector of PageRank scores of vertices.

In Ref. [7], Mihalcea and Tarau proposed to rank words of a word graph using PageRank for keyword extraction. The method was named as *TextRank*. As an unsupervised method, TextRank achieved great improvement on keyword extraction, which led to a F-measure higher than any of the previously proposed systems. Since PageRank scores are not related with initial scores of vertices, TextRank does not take prior knowledge of words into consideration. Empirically, some prior knowledge is important to determine whether a word is a keyword. A straightforward method to employ the prior knowledge in PageRank is *Biased-PageRank*.

2.2 Biased-PageRank

The key to creating Biased-PageRank is that we can bias the computation to increase effect of certain vertices by using a nonuniform $N\times 1$ vector $\boldsymbol{g}$. Biased-PageRank introduces additional rank to the appropriate vertices in each iteration of the computation.

$$\boldsymbol{r} = \lambda \boldsymbol{A}\boldsymbol{r} + (1-\lambda)\boldsymbol{g}. \tag{4}$$

We notice that in PageRank the $\boldsymbol{g}$ is uniform with equal probability to randomly surf to each vertex. For keyword extraction, we assign $\boldsymbol{g}$ using prior knowledge. In Biased-PageRank, the damping factor λ controls the weights of PageRank and prior knowledge in final ranking results.

Moreover, we may be more confident to the words with the highest prior knowledge values. For example, it is more confident for us to identify the words with the highest *tfidf* values as keywords, but we are not convinced to judge the words with their lower knowledge values. Thus, similar to combating web spam [13], instead of assigning the prior knowledge to all words, we select a subset of trusted words to assign the prior knowledge values. The vector $\boldsymbol{g}$ is set to be highly biased to the trusted words and other words are set to zero. In addition, the initial input of $\boldsymbol{r}$ is also set to be $\boldsymbol{g}$. In this paper, we use the trust ratio

$$\mu = \frac{\#\text{trusted words}}{\#\text{all words}}$$

to control the number of trusted words.

2.3 DiffusionRank

DiffusionRank was originally proposed for combating web spam [14], which has also been successfully used in social network analysis [15] and search query suggestion [16]. DiffusionRank is motivated by the heat diffusion process, which can be used for ranking, because the activities flow on the graph can be imagined as heat flow, the edge from a vertex to another can be treated as the pipe of an air-conditioner for heat flow.

Denoting $f_i(t)$ as the heat on vertex v_i at time t, we construct DiffusionRank as follows. Suppose at time t, each vertex v_i receives an amount of heat, $M(v_i, v_j, t, \Delta t)$, from its neighbor v_j during a period Δt. The received heat is proportional to the time period Δt and the heat difference between v_i and v_j, namely, $f_j(t)-f_i(t)$. Based on this, we denote $M(v_i, v_j, t, \Delta t)=\gamma(f_j(t)-f_i(t))\Delta t$, where γ is heat diffusion factor, i.e., the thermal conductivity. Therefore, the heat difference at node v_i between time $t+\Delta t$ and time t is equal to the sum of the heat that it receives from all its neighbors. This is formulated as

$$f_i(t+\Delta t)-f_i(t) = \sum_{v_j \in N(v_i)} \gamma\Big(f_j(t)-f_i(t)\Big)\Delta t. \tag{5}$$

The process can also be expressed in a matrix form:

$$\frac{\boldsymbol{f}(t+\Delta t)-\boldsymbol{f}(t)}{\Delta t} = \gamma \boldsymbol{H}\boldsymbol{f}(t), \tag{6}$$

where $\boldsymbol{f}$ is a vector of heat at vertices at time t, and $\boldsymbol{H}$ is

$$H(i,j) = \begin{cases} -1, & \text{if } i=j, \\ 0, & \text{if } (v_i,v_j)\notin E, \\ \dfrac{w(v_i,v_j)}{w(v_j)}, & \text{if } (v_i,v_j)\in E. \end{cases} \tag{7}$$

If the limit $\Delta t \to 0$, the process will become into

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{f}(t) = \gamma \boldsymbol{H}\boldsymbol{f}(t). \tag{8}$$

Solving this differential equation, we have $\boldsymbol{f}(t)=e^{\gamma t\boldsymbol{H}}\boldsymbol{f}(0)$. Here, we could extend the $e^{\gamma t\boldsymbol{H}}$ as

$$e^{rt\boldsymbol{H}} = \boldsymbol{I} + \gamma t\boldsymbol{H} + \frac{\gamma^2 t^2}{2!}\boldsymbol{H}^2 + \frac{\gamma^3 t^3}{3!}\boldsymbol{H}^3 + \cdots. \tag{9}$$

The matrix $e^{\gamma t\boldsymbol{H}}$ is named as the diffusion kernel in the sense that the heat diffusion process continues infinitely from the initial heat diffusion. γ is an important factor in the diffusion process. If γ is large, the heat will diffuse quickly. If γ is small, the heat will diffuse slowly. When $\gamma \to +\infty$, heat will diffuse immediately, and DiffusionRank becomes into PageRank. As in PageRank and Biased-PageRank, there are random relations among vertices. To capture these relations, we use a uniform random relation among different vertices as in PageRank. Let $1-\lambda$ denote the probability that random surfer happens and λ is the probability of following the edges. Based on the above discussion, we can modify DiffusionRank into

$$\boldsymbol{f}(t) = e^{\gamma t\boldsymbol{R}}\boldsymbol{f}(0),\ \boldsymbol{R} = \lambda\boldsymbol{H} + (1-\lambda)\frac{1}{N}\boldsymbol{1}. \tag{10}$$

In application, a computation of $e^{\gamma t\boldsymbol{R}}$ is time consuming. We usually to approximate it to a discrete form:

$$\boldsymbol{f}(t) = \left(\boldsymbol{I} + \frac{\gamma}{M}\boldsymbol{R}\right)^{Mt}\boldsymbol{f}(0). \tag{11}$$

Without loss of generality, we use one unit time for heat diffusion between vertices and their neighbors, and we have

$$\boldsymbol{f}(1) = \left(\boldsymbol{I} + \frac{\gamma}{M}\boldsymbol{R}\right)^{M}\boldsymbol{f}(0). \tag{12}$$

We could iteratively calculate $\left(\boldsymbol{I} + \frac{\gamma}{M}\boldsymbol{R}\right)^{M}\boldsymbol{f}(0)$ by applying the operator $\left(\boldsymbol{I} + \frac{\gamma}{M}\boldsymbol{R}\right)$ to $\boldsymbol{f}(0)$.

The detail of introduction to DiffusionRank can be found in Ref. [14]. In practice, for each iteraton, we could diffuse the heat values at each vertices using the following formulation:

$$\boldsymbol{r} = \left(1-\frac{\gamma}{M}\right)\boldsymbol{r} + \frac{\gamma}{M}\left[\lambda \boldsymbol{A}\boldsymbol{r} + (1-\lambda)\frac{1}{N}\boldsymbol{1}\right], \quad (13)$$

where M is the number of iterations. As analyzed in Ref. [14], for a given threshold ε, we can compute to get M such that

$$\left\| \left(\boldsymbol{I} + \frac{\gamma}{M}\boldsymbol{R}\right)^M - e^{\gamma \boldsymbol{R}}\boldsymbol{f}(0) \right\| < \varepsilon$$

for any $\boldsymbol{f}(0)$ whose sum is one. Similar to Ref. [14], in this paper, we set $M = 100$ for DiffusionRank.

As shown in the above analysis, prior knowledge is assigned to $\boldsymbol{g}$ in Biased-PageRank. The reason is that, in Biased-PageRank and PageRank, for any given nonzero initial input, the iteration will converge to the same stable distribution corresponding to the maximum eigenvalue 1 of the transition matrix. However, in DiffusionRank, different initial heat input will give rise to different heat distributions after a fixed time period of heat diffusion.

3 Prior knowledge for keyword extraction

3.1 Frequency-based knowledge

Frequency-based knowledge is widely used in both supervised and unsupervised methods for keyword extraction. The most widely used frequency-based feature is *tfidf*. *tfidf* is a standard metric in information retrieval for measuring how specific a word w is to a given document d:

$$prior_{tfidf}(w,d) = \frac{n(w,d)}{\sum_{w' \in d} n(w',d)} \log \frac{|D|}{|d' : w \in d'|},$$

where $n(w, d)$ indicates the number that w occurs in d, and $|D|$ is the number of documents in the document collection. In this equation, the first component is the term frequency (*tf*), namely, the normalized frequency of word w in document d, and the second is the logarithm form of the inverse document frequency (*idf*). Here, we use logarithm form of inverse document frequency because it is common practice in information retrieval [17–19].

3.2 Position-based knowledge

Position-based knowledge is another important feature for determining keywords in various extraction methods. The position of first occurrence of a word is calculated as its distance in words from the end of the document [20]. The distance is usually normalized by the word count of the document. As reported in previous work [21], the words that have extreme values, namely, high or low values, for this feature are more likely to be keywords. This is because either the beginning of the document, such as title, abstract, and introduction, or the final part, such as the conclusion, is usually the summary of the document, in which the words are more likely to be keywords. In this paper, we denote the position of a word w in document d as $prior_{\text{position}}(w, d)$:

$$prior_{\text{position}}(w,d) = \max\left(d(w), 1-d(w)\right), \quad (14)$$

where $d(w)$ indicates the normalized distance of first occurrence of w from the end of d. The feature is a number between 0 and 1 that represents how distant the first appearance of the word is from the beginning or the end of the document.

3.3 Semantic-based knowledge

Among various methods for measuring the relatedness between word and document, latent topic models are one of most widely used. This approach regards a document as a probabilistic mixture of latent topics, and each word of the document is generated from these topics one by one. Latent Dirichlet allocation (LDA) [22] is a representative latent topic model. Compared to latent semantic analysis (LSA) [23] and probabilistic LSA (pLSA) [24], LDA has more feasibility for inference and can avoid over-fitting. Using LDA, we can represent both document and word using a mixture of latent topics, which enables us to measure the relatedness between document and word.

As a generative model, in LDA, each word w of a document d is generated by first sampling a topic z from d's topic distribution $\theta^{(d)}$ and then sampling a word from the distribution over words $\phi^{(z)}$ that characterizes that topic z. In LDA, $\theta^{(d)}$ and $\phi^{(z)}$ are drawn from conjugate Dirichlet priors, α and β separately. Thus, θ and ϕ can be integrated out and the probability of w can be represented as follows:

$$p(w|d,\alpha,\beta) = \sum_{z=1}^{K} p(w|z,\beta)p(z|d,\alpha), \quad (15)$$

where K is the number of topics. Using LDA, we can get the topic distributions of documents and their words, namely, $p(z|d)$ and $p(z|w)$, for each topic $z \in \{1,2,...,K\}$. Given the topic distributions, we have various metrics for measuring the relatedness. In this paper, we select the following two measures.

Given a corpus, we can use some statistical inference techniques to interpret the generative process and infer the topic distributions of words and documents. Inference of LDA by directly maximizing the likelihood of the whole corpus is intractable. Various approximate methods can be used for inference of LDA, including expectation maximization (EM) [24], variational EM [22], expectation propagation [25], and Gibbs sampling [26]. As Gibbs sampling provides a simple and efficient method for obtaining parameter estimates under Dirichlet priors, most people use it to learn LDA by sampling z for each word w from the full conditional posterior distribution. One can refer to Ref. [26] for detailed description of the sampling technique.

Query likelihood, p(document|query), is originally used in IR for ranking documents [27]. Suppose each word w in the document d as a query, we have the metric

$$prior_{\mathrm{QL}}(w,d) = p(d|w) = \sum_{z=1}^{K} p(d|z)p(z|w), \qquad (16)$$

where $p(z)$ indicates the probability of topic z, and $p(d)$ is unchanged when calculating the query likelihood for each word in the document, which can be ignored.

KL-divergence measures the distance between two distributions. It thus can be used to measure the distance between word and document according to their topic distributions:

$$D_{\mathrm{KL}}(w\|d) = \sum_{z=1}^{K} P(z|w)\log\frac{P(z|w)}{P(z|d)}. \qquad (17)$$

In this paper, we want to obtain the semantic relatedness. We thus transform the KL-divergence into relatedness form using

$$prior_{\mathrm{KL}}(w,d) = e^{-1\times D_{\mathrm{KL}}(w\|d)}. \qquad (18)$$

4 Algorithm description

Given a document d for keyword extraction, the graph-based method using prior knowledge for keyword extraction is as follows:

1) **Candidate keywords selection**. We first use simple heuristic rules to find candidate keywords.

2) **Building word graph**. We build a word graph using the co-occurrence relationship of the words within the candidate keywords.

3) **Computing prior knowledge**. We have to compute the prior knowledge of each word in the candidate keywords, which indicates how likely a prior the word is a keyword.

4) **Running DiffusionRank**. Based on the word graph and prior knowledge, we run DiffusionRank and rank words.

5) **Extracting keywords**. Finally, we extract keywords according to ranking values of words.

We describe each step in detail as follows.

4.1 Candidate keywords selection

Not all words in a document are possible to be selected as keyword. To filter out the noisy words in advance, we select candidate keywords using some heuristic rules. This step proceeds as follows. First, the text is tokenized for English or segmented into words for Chinese and other languages without word separators. As reported in Ref. [28], most manually assigned key phrases turn out to be noun phrases. Therefore, we annotate the document with POS tags using Stanford Log-Linear Tagger. Then, we extract the noun phrases whose pattern is zero or more adjectives followed by one or more nouns. With regular expression, the pattern is represented as (ADJECTIVE)∗ (NOUN)+. These noun phrases are selected to be the candidate keywords of this document. The set of candidate keywords is denoted as C. For the adjectives and nouns that have occurred in candidate keywords, we denote as the candidate single-word set S.

To extract keywords, we have to rank candidate keywords according to their importance values and get the top-ranked ones as keywords. The importance value of each candidate keyword is computed using the value of each single word in the candidate keyword. In graph-based methods, the importance values of single words are computed by DiffusionRank. In the next subsection, we demonstrate how to build a word graph for the given document to execute DiffusionRank.

4.2 Building word graph

The word graph is built according to the word relatedness. An intuitive method for measuring word relatedness is based on term co-occurrence relationship within the given document. The co-occurrence relation expresses the cohesion relationships between words in the context of the document.

In this paper, co-occurrence-based relatedness is simply set to the count of co-occurrences within a window of maximum w words in the whole document. As shown in Ref. [8], the best results for keyword extraction are obtained when $w \geqslant 10$. Thus, in the experiments of this paper, the window size w is simply set as $w = 10$.

Each document can be regarded as a word sequence for computing co-occurrence-based relatedness. There are two types of word sequence for counting term co-occurrences. One is the original word sequence without filtering out any words, and the other is after filtering out the stop words or the words with specified POS tags. In this paper, we select the first type because each word in the sequence takes important role for measuring term co-occurrences regardless it is stop word or something else. If we filter out some words, the term relatedness will not be as precise as before.

4.3 Computing prior knowledge

As shown in Sect. 3, for each word in the word graph, we can compute the prior knowledge of its importance in this document, i.e., how likely a prior it will be selected as a part of keywords. The knowledge can be frequency-based, position-based, and semantic-based. In the experiments, we will investigate the efficiency of the three types of prior knowledge in graph-based methods for keyword extraction.

4.4 Running DiffusionRank

To obtain the importance values of each word, we run DiffusionRank on the word graph built in Step 2. Combined with DiffusionRank, the prior knowledge computed in Step 3 is used as a prior of how likely the words are selected as keywords. The prior knowledge can be frequency-based, position-based, or semantic-based.

4.5 Extracting keywords

After obtaining the importance values of each single word, we can compute the importance of each candidate keyword. Following the idea in Ref. [8], the importance value of a candidate keyword c, $score(c)$, is computed by summing the importance values of the single words contained in the candidate keyword:

$$score(c) = \sum_{v_i \in c} score(v_i). \quad (19)$$

All the candidate keywords in document d are ranked in decreasing order of the importance scores and the topic m multiple words are selected as the keywords of d. As in Ref. [8], the m ranges from 1 to 20 in experiments.

5 Experiments

5.1 Dataset

To evaluate the performance of our proposed methods for keyword extraction, we carry out experiments on the dataset constructed by Wan and Xiao[1)], which is first used in Refs. [8,9]. This dataset is based on the new articles in DUC-2001 [29], which is originally used for document summarization containing 308 news articles. All the documents have at least 10 sentences and the average length is 740 words. These news articles are selected from Wall Street Journal, AP Newswire, San Jose Mercury News, Financial Times, LA Times, and FBIS. The selection balance guarantees that there is no bias of article styles in keyword extraction.

The manually assigned keywords are all from corresponding documents, and there are at most 10 keywords for each document. For the dataset, kappa statistic for measuring inter-agreement among keyword annotators is reported to be 0.70 [8,9], which guarantees the quality of the annotation. In this dataset, there are 2488 keywords assigned for documents. The average keyword number is 8.08 and the average word number per keyword is 2.09.

For learning LDA models, we use Wikipedia, the largest encyclopedia in the world, as the corpus. Since the articles in Wikipedia are supposed to compile all human knowledge, we can learn high-quality topic models using Wikipedia. The Wikipedia dataset comes from the March 2008 snapshot[2)]. After removing those non-article pages and removing articles shorter than 100 words, we collected 2122618 articles. After tokenization, stop word removal and word stemming, we build the vocabulary by selecting 20000 words with top document frequency value.

LDA models are learned by taking each Wikipedia article as a document. Before learning LDA models, we have to specify topic number K. In the experiments, we learned several models with different topic numbers, 100, 500, 1000, and 1500, respectively. In evaluating the methods for keyword extraction, we found that the model with $K = 1000$ achieved the best result among different LDA models. Thus, in demonstrating experiment results, we only show the results with $K = 1000$. In practice, the topic number K of LDA can be specified using cross-validation techniques on experimental dataset.

5.2 Evaluation metric

For evaluation, the keywords extracted by different methods are compared with manually labeled keywords. The words in a keyword should be first reduced to their base forms for comparison. In this paper, we use the Porter Stemmer[3)] to complete the process. The precision, recall and F-measure are used as evaluation metrics:

$$p = \frac{\#\text{correct}}{\#\text{manual}},\ r = \frac{\#\text{correct}}{\#\text{extract}},\ F\text{-}measure = \frac{2pr}{p+r}, \quad (20)$$

where #correct, #manual, and #extract are the number of correct extracted keywords, manually assigned keywords, and total extracted keywords, respectively. The evaluation metrics are widely used in keyword extraction task [1,7–9,20,30].

As shown in Refs. [8,9], the performance of keyword extraction is not sensitive to the window size for building graphs. In the following experiments, we simply set the window size to 10.

5.3 Evaluation results — Biased-PageRank

In this section, we demonstrate the influence of prior knowledge under the framework of Biased-PageRank compared to PageRank. There are two parameters in Biased-PageRank, damping factor λ and trust ratio μ.

In Figs. 1(a)–1(c), we show the precision, recall, and F-measure of Biased-PageRank with different damping factor λ. Here, the trust ratio is simply set to $\mu = 1.0$. Since each document has at most 10 keywords, the number of keywords is set to $m = 10$. We can see that, except

1) The dataset could be accessed from the homepage of Xiaojun Wan. http://wanxiaojun1979.googlepages.com.
2) The snapshot of Wikipedia can be accessed from http://en.wikipedia.org/wiki/Wikipedia_database.
3) The package could be accessed from http://tartarus.org/~martin/PorterStemmer/.

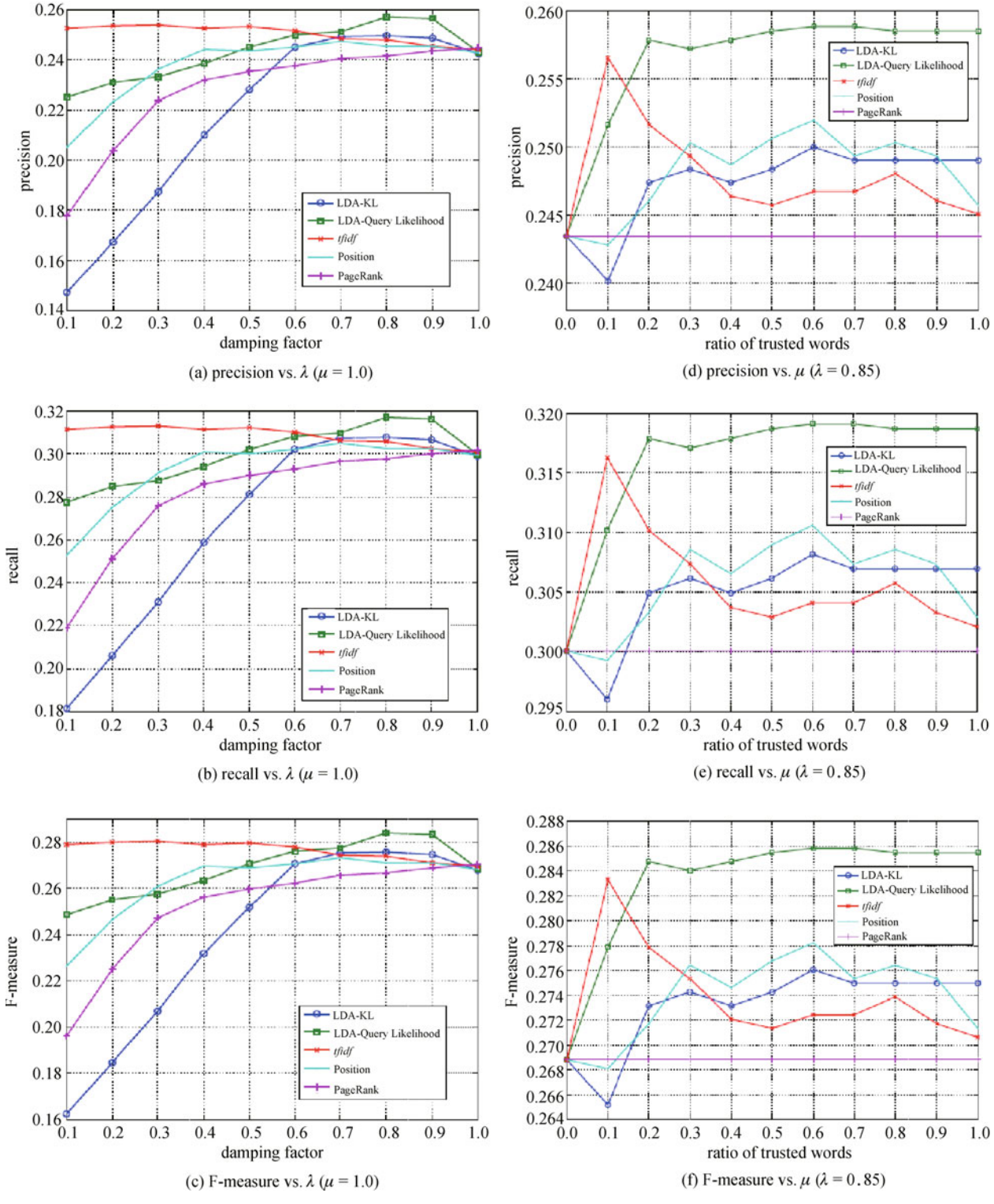

Fig. 1 Performance of Biased-PageRank on precision, recall, and F-measure with different damping factor λ (trust ratio $\mu = 1.0$) and different trust ratio μ (damping factor $\lambda = 0.85$). The number of extracted keywords is set to $m = 10$. (a) Precision versus λ ($\mu = 1.0$); (b) recall versus λ ($\mu = 1.0$); (c) F-measure versus λ ($\mu = 1.0$); (d) precision versus μ ($\lambda = 0.85$); (e) recall versus μ ($\lambda = 0.85$); (f) F-measure versus μ ($\lambda = 0.85$)

KL-divergence using LDA, other measures of prior knowledge outperform PageRank in precision, recall, and F-measure. Among these measures, the improvement of *tfidf* with lower λ and query likelihood using LDA with higher λ are more significant.

Figures 1(d)–1(f) demonstrates the precision, recall, and F-measure of Biased-PageRank with different trust ratio μ. Since most prior knowledge achieve good performance when damping factor λ is between 0.8 and 0.9, here the damping factor is set to $\lambda = 0.85$ and the number of keywords is $m = 10$. Since there is no trust ratio in PageRank, the results of PageRank are unchanged when μ is changing. From the figures, we find that both *tfidf* and query likelihood using LDA obtain more significant improvement. Moreover, *tfidf* is sensitive to different values of trust ratio, while query likelihood is more stable as trust ratio changes. By investigating the value distributions of prior knowledge within a document in descending order, we notice that the distribution of query likelihood is steeper, and there are many zero values for those out-vocabulary words. On the contrary, *tfidf* has a smoother distribution. Therefore, query likelihood is not as sensitive as *tfidf* when trust ratio is growing large.

Based on above analysis, we conclude that Biased-PageRank with three types of prior knowledge all outperform traditional PageRank for keyword extraction.

In DiffusionRank, there are also two parameters: the diffusion factor γ and trust ratio μ. In Fig. 2, we demonstrate the results with different values of trust ratios when $\gamma = 0.5$ and $\gamma = 2.0$.

As described in Sect. 2.3, the diffusion factor γ indicates the diffusion speed of heat. When $\gamma = 0.5$, heat diffuses slowly, and the ranking results are thus more relative to initial input values. When $\gamma = 2.0$, heat diffuses faster, and the results are thus more relevant to word graph topology. In extreme cases, when $\gamma \to +\infty$, heat will diffuse immediately, and DiffusionRank becomes into PageRank, the ranking results of which are independent of initial input values.

From the figures, we find that, when γ is small, i.e., $\gamma = 0.5$, most ranking results underperform traditional PageRank, except for *tfidf*. When γ is larger, i.e., $\gamma = 2.0$, as used in Ref. [14], the ranking results are more relevant to word graph topology and have similar performance to Biased-PageRank with $\lambda = 0.85$. From Figs. 2(d)–2(f), and Figs. 1(d)–1(f) we can find the similar performance between DiffusionRank with $\gamma = 2.0$ and Biased-PageRank with $\lambda = 0.85$.

In Fig. 3, we compare the performance of PageRank and DiffusionRank with *tfidf* knowledge when extracting different number of keywords, from $m = 1$ to $m = 20$. Here, the parameters of DiffusionRank are set to obtain the best results, that is, the trust ratio $\mu = 0.1$ and the diffusion factor $\gamma = 0.5$. From the figure, we can see that the performance of DiffusionRank for keyword extraction is always better than PageRank, especially when the number of keywords m ranges from 2 to 10. Moreover, the improvement of DiffusionRank compared to PageRank on precision is larger than recall.

In Table 1, we demonstrate the best results of PageRank and DiffusionRank when the number of extracted keywords is $m = 10$. For DiffusionRank, the best result is obtained when using *tfidf* measure by setting the trust ratio $\mu = 0.1$ and the diffusion factor $\gamma = 0.5$.

The best results of DiffusionRank outperform PageRank with about 2% improvement. As we have mentioned above, it is hard for keyword extraction to make a gold standard, and even different annotators may assign different keywords to the same document. Therefore, although the absolute value of improvement is small, it does not indicate that the improvement is not remarkable. Furthermore, DiffusionRank provides a flexible mechanism to incorporate prior knowledge by leveraging the diffusion factor γ.

5.4 Combine prior knowledge with neighborhood knowledge

In Refs. [8,9], Wan and Xiao proposed to use a small number of nearest neighbor documents to provide more knowledge to improve single document keyword extraction, which achieved great improvement. In this section, we investigate how to combine prior knowledge with neighborhood knowledge.

5.5 Algorithm description

We use the neighborhood knowledge for keyword extraction as described in Refs. [8,9]. Given a specified document d_0, its nearest neighbors are found by comparing the similarities of other documents with d_0. In Refs. [8,9], the similarity $sim_{\text{doc}}(d_i, d_j)$ between d_i and d_j is computed using the widely used cosine measure and the term weighting method is *tfidf*. Using the *tfidf* values of the words in the document, each document can be represented as a vector $\boldsymbol{w}_d$. Then, the similarity $sim_{\text{doc}}(d_i, d_j)$ can be defined as the normalized inner product of the two term vectors:

$$sim_{\text{doc}}(d_i,d_j) = \frac{\boldsymbol{w}_{d_i} \cdot \boldsymbol{w}_{d_j}}{\left\|\boldsymbol{w}_{d_i}\right\| \times \left\|\boldsymbol{w}_{d_j}\right\|}. \tag{21}$$

Using the cosine measure to compute the pairwise similarity between d_0 and the documents in the corpus, we can choose h documents with largest similarity values as the nearest neighbors of d_0. d_0 and the h documents form the expanded document set D_{d_0} of d_0.

Based on the expanded document set, the word graph is also built according to the co-occurrence relationship. However, the co-occurrence counts in different documents have different weights, i.e., the similarities with d_0 of that documents. Thus, the weight between v_i and v_j is computed

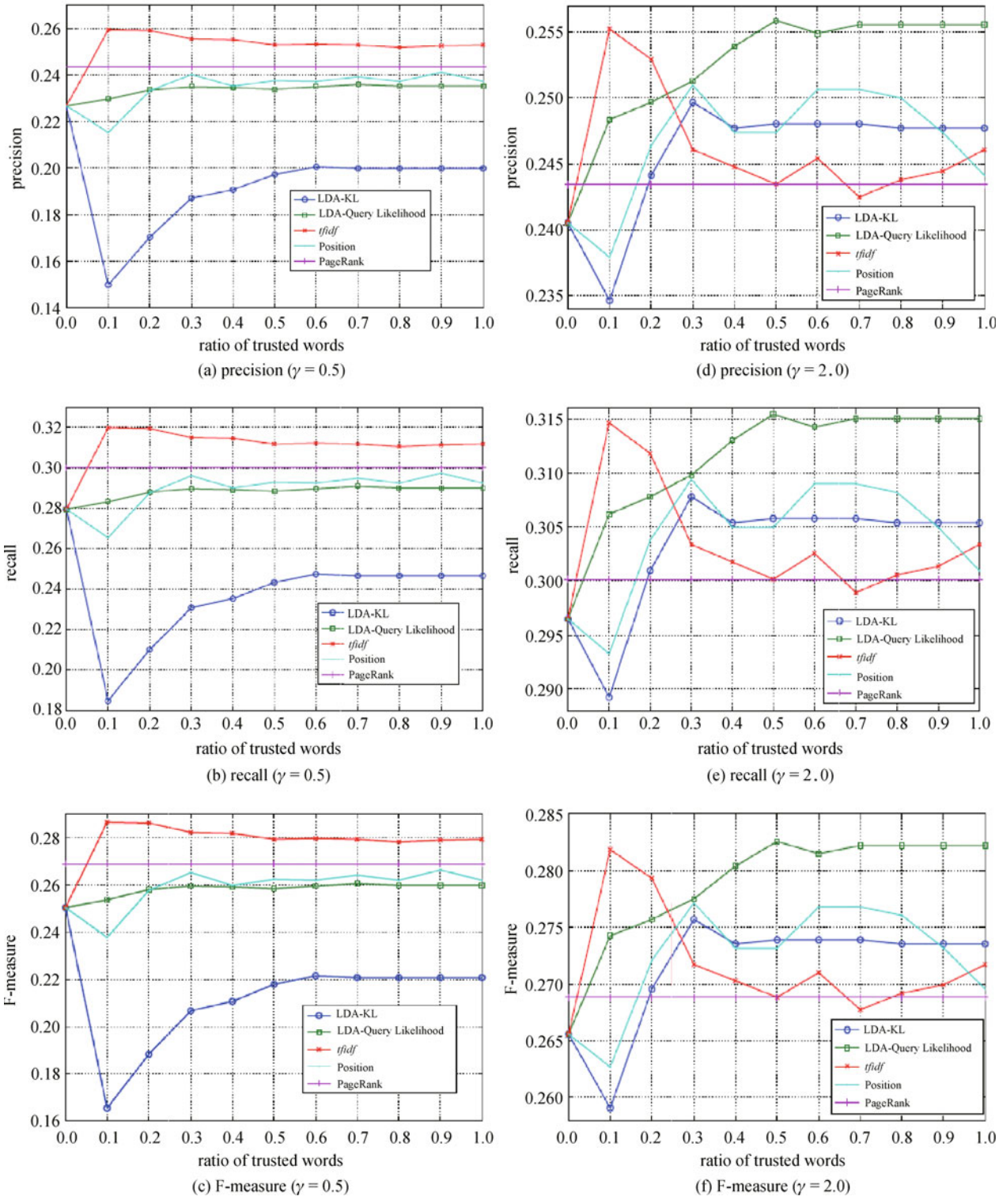

Fig. 2 Performance of DiffusionRank on precision, recall, and F-measure versus different trust ratio μ. The number of extracted keywords $m = 10$ and the diffusion factor $\gamma = 0.5$ and 2.0. (a) Precision ($\gamma = 0.5$); (b) recall ($\gamma = 0.5$); (c) F-measure ($\gamma = 0.5$); (d) precision ($\gamma = 2.0$); (e) recall ($\gamma = 2.0$); (f) F-measure ($\gamma = 2.0$)

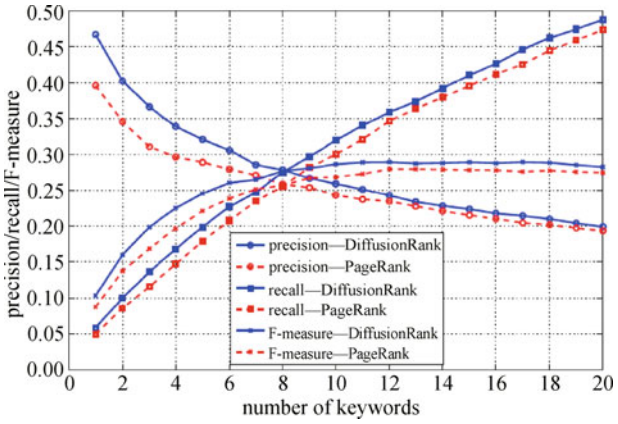

Fig. 3 Performance of DiffusionRank with *tfidf* measure on precision, recall, and F-measure with different numbers of extracted keywords. The trust ratio $\mu = 0.1$ and the diffusion factor $\gamma = 0.5$

Table 1 Results of PageRank and DiffusionRank on total correct extracted keywords, precision, recall, and F-measure when the number of extracted keywords is $m = 10$

method	correct	precision	recall	F-measure
PageRank	744	0.243	0.300	0.269
Biased-PageRank	791	0.259	0.319	**0.286**
DiffusionRank	793	0.259	0.320	**0.287**

as follows:

$$w(v_i,v_j) = \sum_{d_p \in D_{d_0}} sim_{\text{doc}}(d_0,d_p) \times count_{d_p}(v_i,v_j), \quad (22)$$

where $count_{d_p}(v_i,v_j)$ is the count of the co-occurrences between the words v_i and v_j in document d_p, and $sim_{\text{doc}}(d_0,d_p)$ is the similarity factor to reflect the confidence for using document d_p as the expanded document.

After building word graph using neighborhood knowledge, we can apply graph-based ranking algorithms to extract keywords as described in Sect. 4. As reported in Refs. [8,9], the best results were obtained when selecting about 5 neighbor documents for expanded document set. Therefore, we set the neighbor number to 5. However, with the same reason mentioned in Sect. 5.1, although we implement the algorithm described in Refs. [8,9], we did not obtain the same evaluation results reported in Refs. [8,9]. Therefore, we compare the performance based on our implementation instead of what is reported in Refs. [8,9].

Besides building word graphs based on neighborhood knowledge, we can also compute some prior knowledge of words based on neighborhood knowledge. For example, we compute the *tfidf* values by considering the expanded document set as a large document. However, for position-based and semantic-based knowledge, the neighborhood knowledge is not helpful. Therefore, when combined with neighborhood knowledge, the frequency-based prior knowledge perform the best compared to other prior knowledge. The experiment results are shown as follows.

5.6 Experiment results

In DUC-2001 dataset, the news articles are categorized into 30 document sets. Each set has between 3 and 20 documents, with an average of 10 documents, and is on the same news event or topic. Therefore, although the whole dataset is not large, there exist nearest neighbors of high quality for each document. In experiments, we select 5 nearest neighbors for each document as neighborhood knowledge. In Table 2, we demonstrate the results of PageRank and DiffusionRank with neighborhood knowledge when the number of extracted keywords is $m = 10$. The results of Biased-PageRank are obtained when prior knowledge is *tfidf*, damping factor is $\lambda = 0.2$, and trust ratio is $\mu = 0.2$. While the results of DiffusionRank are obtained when prior knowledge is *tfidf*, trust ratio is $\mu = 0.1$ and diffusion factor is $\gamma = 0.5$, which is similar to the settings when there is no neighborhood knowledge shown in Table 1.

Table 2 Results of PageRank and DiffusionRank with neighborhood knowledge on total number of correct extracted keywords, precision, recall, and F-measure when the number of extracted keywords is $m = 10$

method	correct	precision	recall	F-measure
PageRank	818	0.268	0.330	0.296
Biased-PageRank	881	0.288	0.355	**0.318**
DiffusionRank	885	0.290	0.357	**0.320**

With Fig. 4, we demonstrate the performance of PageRank and DiffusionRank with neighborhood knowledge on precision, recall, and F-measure of different number of extracted keywords. In this figure, we also show the results of PageRank without neighborhood knowledge for comparison. From the table, we can see prior knowledge play similar improvement as neighborhood knowledge. Compared to the results with only prior knowledge, the combination of prior knowledge and neighborhood knowledge gains more improvement, with more than 2% improvement compared to PageRank with only neighborhood knowledge and more than 5% improvement compared to traditional PageRank.

6 Conclusions and future work

In this paper, for the first time, we propose to use DiffusionRank to incorporate prior knowledge of words

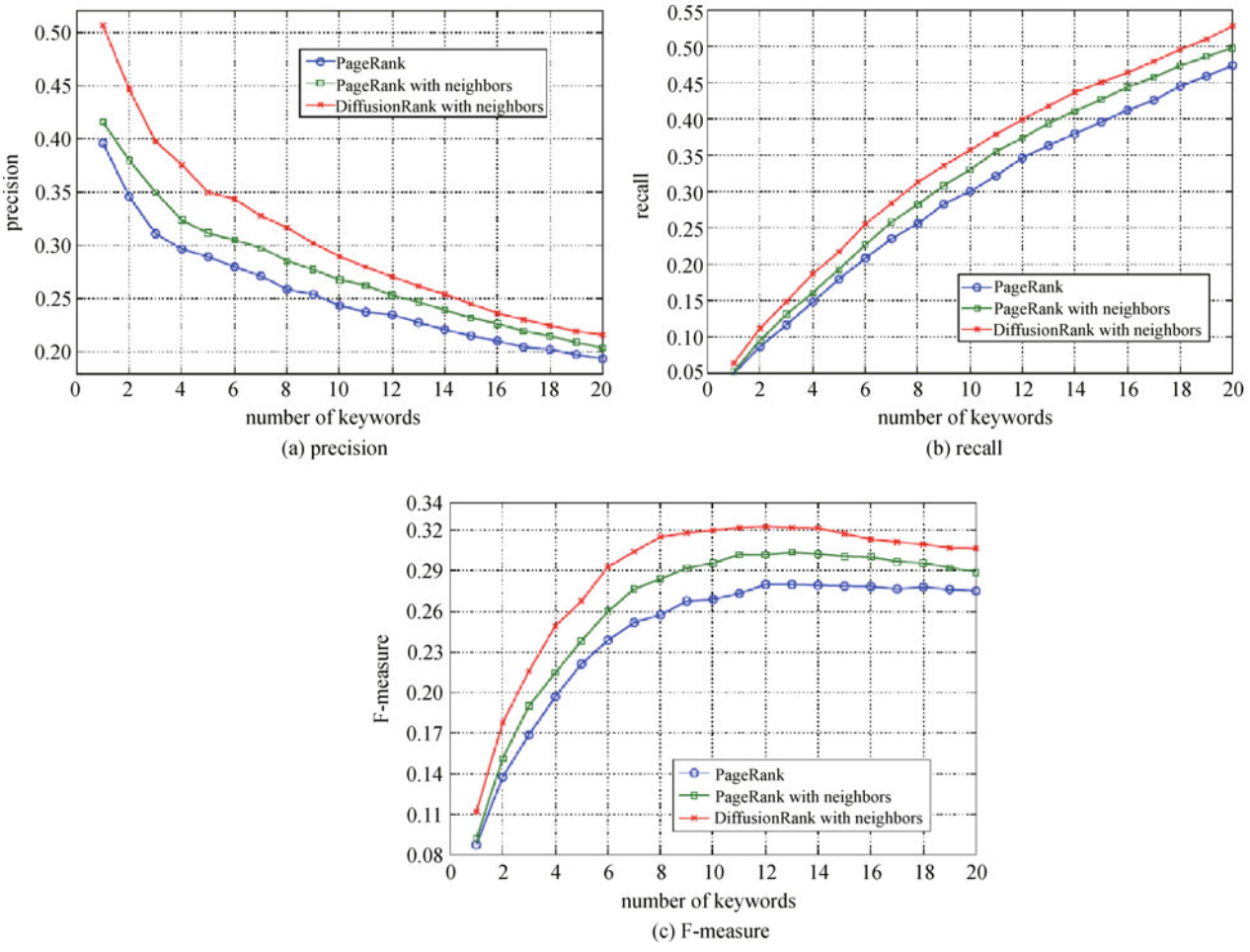

Fig. 4 Performance of PageRank and DiffusionRank using neighborhood knowledge on precision, recall, and F-measure versus different numbers of extracted keywords. Here, the number of nearest neighbors is set $h = 5$. For DiffusionRank, the prior knowledge is calculated using *tfidf* and the trust ratio $\mu = 0.1$ and diffusion factor $\gamma = 0.5$. (a) Precision; (b) recall; (c) F-measure

into graph-based methods for keyword extraction. Experiment results show that prior knowledge can help graph-based methods for keyword extraction. Moreover, by combining prior knowledge with neighborhood knowledge together for keyword extraction, we achieve the highest precision, recall, and F-measure across all compared methods. Similar to PageRank for keyword extraction, our proposed method is easy for implementation and also does not require any training.

In this paper, we only investigate the isolated contributions of several prior knowledge for graph-based keyword extraction. Initial experiments indicate that naive linear combination among various prior knowledge does not help for graph-based methods. We plan to find a good combination of several prior knowledge to improve graph-based ranking methods. Some supervised methods using training set may be adopted to obtain the combination of various knowledge.

References

1. Turney P D. Learning to extract keyphrases from text. Technical Report ERB-1057. Ottawa: National Research Council Canada, 1999
2. Liu Z, Li P, Zheng Y, Sun M. Clustering to find exemplar terms for keyphrase extraction. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing. 2009, 257–266
3. Liu Z, Sun M. Domain-specific term rankings using topic models. In: Proceedings of the 6th Asia Information Retrieval Societies Conference. Lecture Notes in Computer Science, 2010, 6458: 454–465
4. Liu Z, Shi C, Sun M. FolkDiffusion: A graph-based tag suggestion method for folksonomies. In: Proceedings of the 6th Asia Information Retrieval Societies Conference. Lecture Notes in Computer Science, 2010, 6458: 231–240
5. Liu Z, Huang W, Zheng Y, Sun M. Automatic keyphrase extraction via topic decomposition. In: Proceedings of the 2010 Conference on

Empirical Methods in Natural Language Processing. 2010, 366–376
6. Liu Z, Chen X, Zheng Y, Sun M. Automatic keyphrase extraction by bridging vocabulary gap. In: Proceedings of the Fifth Conference on Computational Natural Language Learning. 2011, 135–144
7. Mihalcea R, Tarau P. TextRank: Bringing order into texts. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2004, 404–411
8. Wan X, Xiao J. Single document keyphrase extraction using neighborhood knowledge. In: Proceedings of the 23rd National Conference on Artificial Intelligence. 2008, 855–860
9. Wan X, Xiao J. CollabRank: Towards a collaborative approach to single-document keyphrase extraction. In: Proceedings of the 22nd International Conference on Computational Linguistics. 2008, 969–976
10. Litvak M, Last M. Graph-based keyword extraction for single-document summarization. In: Proceedings of the Workshop Multi-Source Multilingual Information Extraction and Summarization. 2008, 17–24
11. Huang C, Tian Y, Zhou Z, Ling C X, Huang T. Keyphrase extraction using semantic networks structure analysis. In: Proceedings of the Sixth IEEE International Conference on Data Mining. 2006, 275–284
12. Page L, Brin S, Motwani R, Winograd T. The PageRank citation ranking: Bringing order to the web. Technical Report. Stanford Digital Library Technologies Project, 1998, 1–17
13. Gyongyi Z, Garcia-Molina H, Pedersen J. Combating web spam with trustrank. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases. 2004, 576–587
14. Yang H, King I, Lyu M R. DiffusionRank: A possible penicillin for web spamming. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 2007, 431–438
15. Ma H, Yang H, Lyu M R, King I. Mining social networks using heat diffusion processes for marketing candidates selection. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management. 2008, 233–242
16. Ma H, Yang H, King I, Lyu M R. Learning latent semantic relations from clickthrough data for query suggestion. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management. 2008, 709–718
17. Baeza-Yates R, Ribeiro-Neto B. Modern Information Retrieval. Upper Saddle River: Addison-Wesley, 1999
18. Manning C D, Raghavan P, Schütze H. Introduction to Information Retrieval. New York, NY: Cambridge University Press, 2008
19. Croft B, Metzler D, Strohman T. Search Engines: Information Retrieval in Practice. Upper Saddle River: Addison-Wesley, 2009
20. Frank E, Paynter G W, Witten I H, Gutwin C, Nevill-Manning C G. Domain-specific keyphrase extraction. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence. 1999, 668–673
21. Medelyan O, Witten I H. Domain-independent automatic keyphrase indexing with small training sets. Journal of the American Society for Information Science and Technology, 2008, 59(7): 1026–1040
22. Blei D M, Ng A Y, Jordan M I. Latent Dirichlet Allocation. Journal of Machine Learning Research, 2003, 3: 993–1022
23. Landauer T K, Foltz P W, Laham D. An introduction to latent semantic analysis. Discourse Processes, 1998, 25(2–3): 259–284
24. Hofmann T. Probabilistic latent semantic indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 1999, 1–8
25. Minka T, Lafferty J. Expectation-propagation for the generative aspect model. In: Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence. 2002, 352–359
26. Griffiths T L, Steyvers M. Finding scientific topics. Proceedings of the National Academy of Sciences of the United States of America, 2004, 101(Suppl 1): 5228–5235
27. Zhai C. Statistical language models for information retrieval. Synthesis Lectures on Human Language Technologies, 2008, 1(1): 1–141
28. Hulth A. Improved automatic keyword extraction given more linguistic knowledge. In: Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing. 2003, 216–223
29. Over P, Liggett W, Gilbert H, Sakharov A, Thatcher M. Introduction to DUC-2001: An intrinsic evaluation of generic news text summarization systems. In: Proceedings of 2001 Document Understanding Conference. 2001
30. Turney P D. Learning algorithms for keyphrase extraction. Information Retrieval, 2000, 2(4): 303–336