MIKE: Keyphrase Extraction by Integrating Multidimensional Information

Yuxiang Zhang
Civil Aviation University of China
Tianjin, China
yxzhang@cauc.edu.cn

Yaocheng Chang
Civil Aviation University of China
Tianjin, China
ycchang_yjs15@cauc.edu.cn

Xiaoqing Liu
Civil Aviation University of China
Tianjin, China
xqliu_yjs15@cauc.edu.cn

Sujatha Das Gollapalli
Institute for Infocomm Research, A*STAR
Singapore, Singapore
gollapallis@i2r.a-star.edu.sg

Xiaoli Li
Institute for Infocomm Research, A*STAR
Singapore, Singapore
xlli@i2r.a-star.edu.sg

Chunjing Xiao
Civil Aviation University of China
Tianjin, China
cjxiao@cauc.edu.cn

ABSTRACT

Traditional supervised keyphrase extraction models depend on the features of labelled keyphrases while prevailing unsupervised models mainly rely on structure of the word graph, with candidate words as nodes and edges capturing the co-occurrence information between words. However, systematically integrating all these *multidimensional heterogeneous information* into a *unified model* is relatively unexplored. In this paper, we focus on how to effectively exploit *m*ultidimensional *i*nformation to improve the *k*eyphrase *e*xtraction performance (MIKE). Specifically, we propose a random-walk parametric model, MIKE, that learns the latent representation for a candidate keyphrase that captures the mutual influences among all information, and simultaneously optimizes the parameters and ranking scores of candidates in the word graph. We use the gradient-descent algorithm to optimize our model and show the comprehensive experiments with two publicly-available WWW and KDD datasets in Computer Science. Experimental results demonstrate that our approach significantly outperforms the state-of-the-art graph-based keyphrase extraction approaches.

KEYWORDS

Keyphrase extraction; multidimensional information; graph-based keyphrase extraction approach; parametric model

1 INTRODUCTION

Automatic keyphrase extraction extracts a set of *representative phrases* that are related to the main *topics* discussed in a document [27]. Since keyphrases can provide a high-level topic description of a document, they are useful for a wide range of natural language processing tasks such as text summarization [22], information retrieval [25] and question answering [44]. However, the performance of existing methods is still far from being satisfactory [17, 26]. The main reason is it is very challenging to determine if a phrase or set of phrases accurately capture main topics that are presented in a document.

Existing methods for keyphrase extraction can be broadly divided into *supervised or unsupervised approaches*. The supervised approaches generally treat the keyphrase extraction as a binary classification task, in which a learning model is trained on the *features of labelled keyphrases* to determine whether a candidate phrase is a keyphrase. Current state-of-the-art supervised models include Naïve Bayes [8, 9, 13], Decision Trees [34, 37], Maximum Entropy [45], Random Forest [1], Multi-layer Perceptron [28], Support Vector Machines (SVM) [28], Conditional Random Fields [15], Integer Linear Program [7, 10], Deep Recurrent Neural Networks [46] etc. Although these classification approaches can achieve good results given sufficient training data for model building, they do not incorporate information about which candidates are better than the others. Hence, ranking-based supervised approaches, such as ranking SVM [21] and CoRankBayes [39], were shown to outperform the conventional binary classification approaches on the keyphrase extraction task [21].

In contrast, unsupervised approaches directly treat keyphrase extraction as a ranking problem, scoring each candidate using different kinds of techniques such as language modelling [36], clustering [27] or graph-based ranking [12, 14, 26, 31, 35, 38]. In particular, the graph-based algorithms are widely used in the unsupervised scenario. These approaches first build *a word graph* (as illustrated in Fig. 1(2)) in which each node denotes a candidate word and each edge/link prepresents a relevant relation (e.g., co-occurrence relation) between candidates within a document. Subsequently, various centrality measures [6] or random walk techniques (e.g., PageRank [32]) are used on the word graph to rank candidate words. Note that PageRank-based models (i.e., random-walk models) were shown to be state-of-the-art in previous research involving word graphs for keyphrase extraction [14]. The main reason is that the PageRank score for a node provides a measure of importance for the node in the graph by taking into account global information computed recursively from the entire word graph [32]. Furthermore, in order to extract good keyphrases relevant to the major *topics* of the document and cover the document's major topics, Liu [2010] proposed the Topical PageRank (TPR, as shown in Fig.1(3)), which

CIKM'17, November 6–10, 2017, Singapore.

ISBN 978-1-4503-4918-5/17/11…$15.00
DOI: https://doi.org/10.1145/3132847.3132956

decomposes traditional Page-Rank into multiple PageRanks specific to various topics and obtains the importance scores of words under different topics.

As summarized above, current supervised approaches only rely on *candidate features* without considering the structure of word graph, whereas graph-based approaches mainly rely on the *structure of the word graph* and *edge features* to rank candidates. In particular, as illustrated in Fig.1(1-2), four categories of *candidate features* (node features) are commonly used in supervised approaches (shown in Fig. 1(1)), and *structural information* and *edge features* are used in graph-based unsupervised approaches (such as TextRank [31]) to rank candidate words (shown in Fig. 1(2)). It is obvious to see that one major limitation for both supervised approaches and graph-based approaches is they utilize candidate features and structural information separately. That is to say, there is a lack of a general effective way to integrate these different types of information into a unified model for keyphrase extraction.

Furthermore, in contrast to TextRank (as shown in Fig. 1(2)), the TPR [26] (as shown in Fig. 1(3)) integrates *topical information* of candidate words and documents into the TextRank. Fig. 1 from (2) to (3) illustrates an intuitive research line that different types of information are progressively fused to extract good keyphrases in the graph-based approaches (The details of this research line will be described in the next section.). Besides, multidimensional information has been effectively used to benefit various data mining tasks in previous works, such as air quality forecasting [49], traffic congestion estimation [42].

Motivated by these two observations, we propose MIKE, a graph-based approach to leverage multidimensional heterogeneous information to gain better performance in keyphrase extraction. Specifically, MIKE (as shown in Fig. 1(4)) integrates five different types of information, i.e., *features of candidate words*, *topical information of the document*, *structure of the word graph*, *features of co-occurrence between candidates* and *prior knowledge*, and captures the latent influences among all these information on a candidate word.

Three features that distinguish MIKE from many other graph-based keyphrase extraction approaches are as follows:

(1) MIKE integrates five different types of information mentioned above. To our best knowledge, our model includes almost all of the types of information which have been used for extracting keyphrases in previous works, and none of the existing approaches incorporates these different types of information into a unified model.
(2) MIKE is a random-walk optimization model based on the word graph. It can automatically learn the weights of node features and edge features. In pervious graph-based keyphrase extraction approaches, the weights of edges are usually given in advance.
(3) MIKE is a scalable method. Any new node features and edge features can be easily added to our model to rank candidate words with other information.

The remaining of this paper is organized as follows. In the following section, we summarize related, state-of-the-art approaches to keyphrase extraction, especially, the graph-based approaches. Preliminaries are given in Section 3. MIKE is described in detail

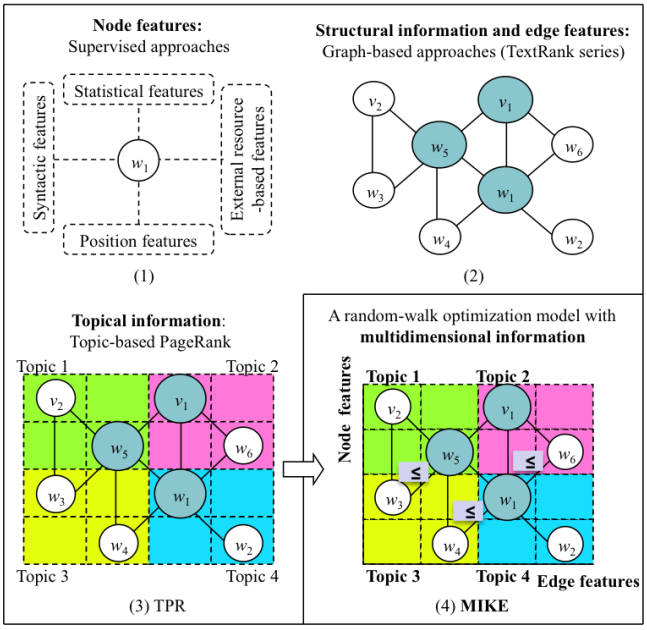

Figure 1: Different approaches for keyphrase extraction with different information

in the "Proposed Model" section. Finally, we present our datasets, experiments and results before concluding the paper.

2 RELATED WORKS

In addition to the keyphrase extraction methods discussed in Introduction Section, a recent trend toward further improvement is to incorporate external knowledge to enhance the effectiveness of existing models. More specifically, in the supervised approaches, many external knowledge-based features, such as the number of times that candidate is retrieved in search engine [11], Wikipedia-based keyphrasenes [30], Wikipedia-based category hierarchy and a list of multiword expressions [3], Wikipedia-based semantic correlations among candidate phrases [40], citation context-based features [9] and expert knowledge [15], are proposed and used in conjunction with traditional candidate features.

In the existing graph-based approaches, TextRank [31] is the first to rank keyphrases based on the co-occurrence links between candidate words. Following this approach, numerious studies use different background knowledge to enhance the accuracy of the word graph. ExpandRank [38] leverages a small number of nearest neighbor documents to compute more accurate co-occurrences in word graph. TPR [26] first incorporates topical information into TextRank model which increases the weight of important topics generated by a topical model. Zhao [2011] also proposed a context-sensitive topical TextRank method for extracting keyphrases in microblogs. CiteText-Rank [14] incorporates evidence from a citation context to enhance co-occurrences in word graph. Zhang [2013] proposed a supervised random walk model which combines the semantic relations between words and words' intrinsic attributes. Bellaachia [2014] replaced conventional word graph with a hypergraph, which is capable of integrating temporal and social features

into the ranking for short document (e.g. messages and tweets). Wang [2015b] integrated information provided by word embedding vectors into a graph-based ranking approach. Additionally, two algorithms were recently proposed to rank web pages by taking into account metadata associated with nodes and edges within PageRank [43, 50]. However, these methods didn't consider the topical information of web pages. Our proposed model MIKE is a unified random-walk model that not only incorporates candidate features, link features and structure of the word graph but also considers the influence of different topics of nodes and prior knowledge on our ranking scores.

3 FRAMEWORK AND PRELIMINARIES

3.1 Framework

Algorithms for graph-based keyphrase extraction commonly involve four steps: candidate word selection, word graph construction, candidate word ranking and keyphrase extraction [14, 26, 38, 47]. In this work, our framework illustrated in Fig.1(4) consists of the following five steps:

(1) **Candidate word selection.** In this step, candidate words or lexical units are extracted from the textual content of the target document by applying stopword and parts-of-speech (POS) filters. Only nouns and adjectives are selected as candidate words because most of keyphrases are noun phrases and they commonly consist of nouns and adjectives [14, 20, 27].
(2) **Word graph construction.** Similar to the TextRank family of algorithms for keyphrase extraction, a word graph is built according to the co-occurrence relation between candidate words within the target document. Note that some graph-based methods construct the word graph on a set of documents that are related to the the target, such as neighbor documents [38] and citation contexts [14].
(3) **Multidimensional information integration.** Features of candidate words and their co-occurrences (i.e., node features and edge features), topic distributions of candidates and relative importance relation between candidates (i.e., prior knowledge) are collected and then integrated into a unified model. Compared to previous works in which node features and edge features are exploited in supervised methods and unsupervised methods separately, our work integrates these different types of information into a random-walk parametric model.
(4) **Candidate word ranking using a learning model.** In this step, our model defines a loss function that is used to learn the parameters and simultaneously compute the ranking score of each candidate word, and optimizes it using gradient descent. Different from previous graph-based models in which the weights of edges are usually assigned in advance, our model automatically learns the weights of node features and edge features.
(5) **Keyphrase extraction.** Finally, consecutive words, phrases or n-grams are scored by using the sum of scores of individual words that comprise the phrase [14, 38]. The formula used for computing the sum of scores is given as follows:

$$R(p) = \psi_p \sum_{w \in p} R(w), \tag{1}$$

where $R(w)$ represents the ranking score of candidate word w, and ψ_p is a weight of p according to the length of phrase p. The top-scoring phrases are output as the final results (the keyphrases for the document).

Our main contributions lie in the Step 3 to Step 4. We now briefly summarize Topic Discovery and Topical PageRank which provide the essential background for our own model.

3.2 Topic Discovery

We first describe how we discover a set of topics. Topics can be automatically extracted from a set of documents using different statistical methods. Latent Dirichlet Allocation (LDA) [5], a Bayesian generative probabilistic model for latent topic layer is widely used to model scientific text corpora. Compared to Latent Semantic Analysis (LSA) [23] and probabilistic LSA (pLSA) [19], LDA has more feasibility for inference and can reduce the risk of over-fitting [26]. In LDA, for each document d, a multinomial distribution θ_d over K topics is sampled from a Dirichlet with parameter α. For each word w, a topic z is chosen from the topic distribution. The word w is generated from a topic-specific multinomial distribution ϕ_z, which is drawn from a Dirichlet with parameter β. The generating probability of the word w from a document d is denoted as follows:

$$Pr(w|d, \alpha, \beta) = \sum_{z=1}^{K} Pr(w|z, \beta) Pr(z|d, \alpha). \tag{2}$$

Using LDA, we can obtain the probability that the word w occurs in a given topic z (i.e., $Pr(w|z)$) and the topic distribution of a document d (i.e., $Pr(z|d)$). These values will be used for ranking keyphrases for different topics.

3.3 Topical PageRank (TPR) Models

TPR [26] first embeds topics into the TextRank and performs the biased TextRank for each topic separately to ensure that the extracted keyphrases cover the main topics of the document. The final score of a word is computed as the sum of its scores for each of the topics, weighted by the probability of that topic in the given document. More specifically, for each word w_i in the word graph, its TPR score in the topic z can be computed as follows:

$$R_z(w_i) = \lambda \sum_{j: w_j \to w_i} \left(\frac{e(w_j, w_i)}{O(w_j)} R_z(w_j) \right) + (1 - \lambda) Pr_z(w_i). \tag{3}$$

where $e(w_j, w_i)$ is the weight of edge $w_j \to w_i$, $O(w_j) = \sum_{w'} e(w_j, w')$ is the number of outbound edges, λ is a damping factor indicating the probability of a random jump to another node, and $Pr_z(w_i)$ is a topic specific preference for a topic z and can be further calculated by $Pr(w_i|z)$ (which is calculated by formula (2)) with the constraint $\sum_w Pr_z(w) = 1$. The final ranking score $R(w_i)$ of word w_i in document d is computed as the expected PageRank score over that topic distribution, $R(w_i) = \sum_{z=1}^{K} R_z(w_i) Pr(z|d)$ ($Pr(z|d)$ is calculated by formula (2)).

Because TPR requires running a TextRank for each topic, the total amount of TextRank is $K \times c$, where c is the size of the set of text documents. In order to avoid this large computational cost, Sterckx [2015] proposed a single global weight value (Single-TPR),

which represents the full topical importance of each word in the TextRank, to replace K topic-specific values and sum all results. Specifically, let $W_z(w_i)$ be the full topical importance of each word w_i over K topics in the TextRank, $\overrightarrow{Pr}(w_i|Z) = (Pr(w_i|z_1), Pr(w_i|z_2), ..., Pr(w_i|z_K))$ be the vector of word-topic probabilities for the word w_i, and $\overrightarrow{Pr}(Z|d) = (Pr(z_1|d), Pr(z_2|d), ..., Pr(z_K|d))$ be the vector of document-topic probabilities of the document d. Then, the $W_z(w_i)$ per word w_i and document d can be calculated by the cosine similarity between $\overrightarrow{Pr}(w_i|Z)$ and $\overrightarrow{Pr}(Z|d)$,

$$W_z(w_i) = \frac{\overrightarrow{Pr}(w_i|Z) \cdot \overrightarrow{Pr}(Z|d)}{\|\overrightarrow{Pr}(w_i|Z)\| \cdot \|\overrightarrow{Pr}(Z|d)\|}. \tag{4}$$

As a result, the TPR score $R(w_i)$ becomes

$$R(w_i) = \lambda \sum_{j:w_j \to w_i} \left(\frac{e(w_j, w_i)}{O(w_j)} R(w_j) \right) + (1-\lambda) \frac{W_z(w_i)}{\sum_w W_z(w)}. \tag{5}$$

Different from these topic-dedicated reset probability shown in the TPR [26] and Single-TPR [35], we first integrate the full topical importance mentioned in the formula (4) and many features of candidate words into the reset probability.

4 OUR PROPOSED MODEL: MIKE

4.1 Problem Formulation

Definition 4.1. **(Keyphrase Extraction)** Let $D = \{d_1, d_2, ..., d_m\}$ be a set of m text documents. Each document $d_i \in D$ includes a set of candidate words $W_i = \{w_{i1}, w_{i2}, ..., w_{ii_n}\}$. The goal of a keyphrase extraction model is to find a function to map each $w_{ij} \in W_i$ into a score, and then extract a ranked list of phrases (which consist of consecutive words) that best represents d_i.

4.2 Random Walk for Keyphrase Extraction

A word graph is defined as a six-tuple $G(V, E, A^V, A^E, Z, B)$, where V is a set of n nodes, E is a set of m edges, $A^V = (\overrightarrow{a}_i^V, i \in V)$ is a matrix composed of feature vectors of nodes ($\overrightarrow{a}_i^V = (a_{ij}^V, i \in V, 1 \le j \le l)$ is an l-dimensional feature vector of node i), $A^E = (\overrightarrow{a}_{ij}^E, i, j \in V)$ is a matrix composed of feature vectors of edges ($\overrightarrow{a}_{ij}^E = (a_{ijk}^E, i, j \in V, 1 \le k \le h)$ is an h-dimensional feature vector of edge $w_i \to w_j$), $Z = (\overrightarrow{z}_i, i \in V)$ is a matrix composed of topic vectors of nodes ($\overrightarrow{z}_i = (z_{ij}, i \in V, 1 \le j \le K)$ is an K-dimensional topic vector of node i), B is a q-by-n supervision matrix encoded by the prior knowledge (where q is the number of samples of prior knowledge).

In contrast with existing graph-based keyphrase extraction algorithms, MIKE leverages the *features of co-occurrence between candidates*, *features of candidate* and *topics of candidate* by substantially modifying the ***transition probability matrix*** and ***reset probability vector*** corresponding to the underlying word graph.

Features of edges are integrated into the transition probability. The function $f(\overrightarrow{\omega}, w_i \to w_j) = <\overrightarrow{\omega}, \overrightarrow{a}_{ij}^E>$ parameterized by vector $\overrightarrow{\omega} = (\omega_1, \omega_2, ..., \omega_h)$ is defined to assign a weight to each edge in E. Therefore, each entry $p_{ij}(\overrightarrow{\omega})$ of the transition probability matrix $P(\overrightarrow{\omega})$ in MIKE is calculated as follows:

$$p_{ij}(\overrightarrow{\omega}) = \frac{(\overrightarrow{\omega})^{\mathrm{T}} \overrightarrow{a}_{ij}^E}{\sum_j (\overrightarrow{\omega})^{\mathrm{T}} \overrightarrow{a}_{ij}^E} = \frac{\sum_k \omega_k a_{ijk}^E}{\sum_j \sum_k \omega_k a_{ijk}^E}. \tag{6}$$

Similarly, *features of nodes* are combined into the reset probability. The function $g(\overrightarrow{\varphi}, w_i) = <\overrightarrow{\varphi}, \overrightarrow{a}_i^V>$ parameterized by vector $\overrightarrow{\varphi} = (\varphi_1, \varphi_2, ..., \varphi_l)$ is defined to assign a weight to each node in V. Then, each element $r_i(\overrightarrow{\varphi})$ of the reset probability vector $\overrightarrow{r}(\overrightarrow{\varphi})$ in MIKE is calculated as follows:

$$r_i(\overrightarrow{\varphi}) = \frac{(\overrightarrow{\varphi})^{\mathrm{T}} \overrightarrow{a}_i^V}{\sum_i (\overrightarrow{\varphi})^{\mathrm{T}} \overrightarrow{a}_i^V} = \frac{\sum_k \varphi_k a_{ik}^V}{\sum_i \sum_k \varphi_k a_{ik}^V}. \tag{7}$$

In order to further consider the effect of all topics of a node on the reset probability, the *topics* of a node are merged into the reset probability. Accordingly, the element $r_i(\overrightarrow{\varphi})$ of the reset probability vector $\overrightarrow{r}(\overrightarrow{\varphi})$ is replaced by the $r_{z_i}(\overrightarrow{\varphi})$ of the new reset probability vector $\overrightarrow{r_z}(\overrightarrow{\varphi})$ in MIKE. $r_{z_i}(\overrightarrow{\varphi})$ can be calculated as follows:

$$r_{z_i}(\overrightarrow{\varphi}) = \frac{W_{z_i} \sum_k \varphi_k a_{ik}^V}{\sum_i \left(W_{z_i} \sum_k \varphi_k a_{ik}^V \right)}, \tag{8}$$

where W_{z_i} is the same as $W_z(w_i)$ given in formula (4) and is used to indicate the full topical importance of the word w_i over all K topics.

Then, we employ the loss function $\|\overrightarrow{\pi}^{(s+1)} - \overrightarrow{\pi}^{(s)}\|^2$ as the objective function of MIKE, where $\overrightarrow{\pi} = (\pi_1, \pi_2, ..., \pi_n)^{\mathrm{T}}$ is a ranking score vector of all nodes and is iteratively calculated according to the principle of PageRank during the Markov random process,

$$\overrightarrow{\pi}^{(s+1)} = \lambda P(\overrightarrow{\omega}^{(s)}) \overrightarrow{\pi}^{(s)} + (1-\lambda) \overrightarrow{r_z}(\overrightarrow{\varphi}^{(s)}) \quad \|\overrightarrow{\pi}^{(s)}\|_1 = 1. \tag{9}$$

At the same time, we simultaneously optimize the parameter $\overrightarrow{\omega}$ and $\overrightarrow{\varphi}$ during this process. Consequently, *the objective function* of MIKE is defined as follows:

$$\begin{aligned} &\|\overrightarrow{\pi}^{(s+1)} - \overrightarrow{\pi}^{(s)}\|^2 = \\ &\quad \|\lambda P(\overrightarrow{\omega}^{(s)}) \overrightarrow{\pi}^{(s)} + (1-\lambda) \overrightarrow{r_z}(\overrightarrow{\varphi}^{(s)}) - \overrightarrow{\pi}^{(s)}\|^2. \end{aligned} \tag{10}$$

The *relative importance relation* between nodes is treated as prior knowledge in MIKE. Specifically, for two given nodes w_i and w_j, we assume w_i has higher relative importance than w_j, without loss of generality. In other words, there exists a partial ordering, such as $w_i \succcurlyeq w_j$, which is represented in the row x ($1 \le x \le q$) of B as follows:

$$b_{xi} = 1, \quad b_{xj} = -1, \quad b_{xy} = 0 (y \ne i, j). \tag{11}$$

In order to supervise the learning process, the *constraints* of MIKE are defined based on the relative importance relation between nodes. If the w_i has higher relative importance than w_j, we hope the score of nodes w_i and w_j in the final ranking list is $\pi_i - \pi_j \ge 0$. The constraint function is thus defined as

$$S(\overrightarrow{\pi}; B, \overrightarrow{\mu}) = -\overrightarrow{\mu}(\overrightarrow{1} - B\overrightarrow{\pi}) > 0. \tag{12}$$

where $\overrightarrow{\mu} = (\mu_1, \mu_2, ..., \mu_q)$ denotes weights on samples of supervision information, $\overrightarrow{1}$ is a column vector of 1. Finally, the objective function in (9) can be transformed as the following *optimization problem*:

$$\begin{aligned} \min_{\overrightarrow{\omega} \ge 0; \overrightarrow{\varphi} \ge 0; \overrightarrow{\pi} \ge 0} \mathcal{L}(\overrightarrow{\omega}^{(s)}, \overrightarrow{\varphi}^{(s)}, \overrightarrow{\pi}^{(s)}) &= \|\overrightarrow{\pi}^{(s+1)} - \overrightarrow{\pi}^{(s)}\|^2 \\ &= \delta \|\lambda P(\overrightarrow{\omega}^{(s)}) \overrightarrow{\pi}^{(s)} + (1-\lambda) \overrightarrow{r_z}(\overrightarrow{\varphi}^{(s)}) - \overrightarrow{\pi}^{(s)}\|^2 \\ &\quad + (1-\delta) \overrightarrow{\mu}(\overrightarrow{1} - B\overrightarrow{\pi}). \end{aligned} \tag{13}$$

4.3 Optimization Problem Solution

To solve the optimization problem stated in formula (13), we use the gradient descent algorithm to minimize function $\mathcal{L}$. Each element $\nabla\omega_k^{(s)}(1 \le k \le h)$ of the partial derivatives of $\mathcal{L}$ with respect to ω can be calculated as

$$\nabla\omega_k^{(s)} = \frac{\partial\mathcal{L}}{\partial\omega_k^{(s)}} = 2\delta\lambda\left(\overrightarrow{\pi}^{(s)}\right)^{\mathrm{T}}\left[\frac{\partial P(\overrightarrow{\omega}^{(s)})}{\partial\omega_k^{(s)}}\right]^{\mathrm{T}} \left[\left(\lambda P(\overrightarrow{\omega}^{(s)}) - \mathrm{I_n}\right)\overrightarrow{\pi}^{(s)} + (1-\lambda)\overrightarrow{r_z}(\overrightarrow{\varphi})\right]. \tag{14}$$

In formula (14), each entry of the partial derivative can be further calculated as

$$\frac{\partial p_{ij}(\overrightarrow{\omega}^{(s)})}{\partial\omega_k^{(s)}} = \frac{a_{ijk}^E}{\sum_j\sum_k\omega_k^{(s)}a_{ijk}^E} - \frac{(\sum_k\omega_k^{(s)}a_{ijk}^E)\sum_j a_{ijk}^E}{\left[\sum_j\sum_k\omega_k^{(s)}a_{ijk}^E\right]^2}. \tag{15}$$

Each element $\nabla\varphi_k^{(s)}(1 \le k \le l)$ of the partial derivatives of $\mathcal{L}$ with respect to φ can be calculated as

$$\nabla\varphi_k^{(s)} = \frac{\partial\mathcal{L}}{\partial\varphi_k^{(s)}} = 2\delta(1-\lambda)\left(\overrightarrow{\pi}^{(s)}\right)^{\mathrm{T}}\left[\frac{\partial\overrightarrow{r_z}(\overrightarrow{\varphi}^{(s)})}{\partial\omega_k^{(s)}}\right]^{\mathrm{T}} \left[\left(\lambda P(\overrightarrow{\omega}^{(s)}) - \mathrm{I_n}\right)\overrightarrow{\pi}^{(s)} + (1-\lambda)\overrightarrow{r_z}(\overrightarrow{\varphi})\right]. \tag{16}$$

In formula (16), each element of the partial derivative can be further calculated as

$$\frac{\partial r_{z_i}(\overrightarrow{\varphi}^{(s)})}{\partial\varphi_k^{(s)}} = \frac{W_{z_i}a_{ik}^V}{\sum_i(W_{z_i}\sum_k\varphi_k^{(s)}a_{ik}^V)} - \frac{W_{z_i}(\sum_k\varphi_k^{(s)}a_{ik}^V)\sum_i W_{z_i}a_{ik}^V}{\left[\sum_i(W_{z_i}\sum_k\varphi_k^{(s)}a_{ik}^V)\right]^2}. \tag{17}$$

Each element $\nabla\pi_k^{(s)}(1 \le k \le n)$ of the partial derivatives of $\mathcal{L}$ with respect to π can be calculated as

$$\nabla\pi_k^{(s)} = \frac{\partial\mathcal{L}}{\partial\pi_k^{(s)}} = 2\delta\left[\frac{\partial\left(\left(\lambda P(\overrightarrow{\omega}^{(s)}) - \mathrm{I_n}\right)\overrightarrow{\pi}^{(s)}\right)}{\partial\pi_k^{(s)}}\right]^{\mathrm{T}} \left[\left(\lambda P(\overrightarrow{\omega}^{(s)}) - \mathrm{I_n}\right)\overrightarrow{\pi}^{(s)} + (1-\lambda)\overrightarrow{r_z}(\overrightarrow{\varphi}^{(s)})\right] - (1-\delta)\sum_{i=1}^{q} b_{ik}\mu_i. \tag{18}$$

In formula (18), the partial derivative can be calculated as

$$\frac{\partial\left(\left(\lambda P(\overrightarrow{\omega}^{(s)}) - \mathrm{I_n}\right)\overrightarrow{\pi}^{(s)}\right)}{\partial\pi_k^{(s)}} = \left(\lambda P_{1k}(\overrightarrow{\omega}^{(s)}) - 1, ..., \lambda P_{nk}(\overrightarrow{\omega}^{(s)}) - 1\right)^{\mathrm{T}}. \tag{19}$$

4.4 Algorithm

This detailed algorithm is shown in Algorithm 1, where ρ is the learning rate and ϵ is an error estimator which indicates the stopping condition. From steps 2 to 4, we first initialize all static parameters (including δ, λ, ρ, ϵ), iteration variable s and optimizing parameters (including $\overrightarrow{\pi}^{(s)}, \overrightarrow{\omega}^{(s)}, \overrightarrow{\varphi}^{(s)}$). Steps 6 and 7 calculate respectively the transition probability matrix $P^{(s)}$ by formula (6), reset probability $\overrightarrow{r_z}^{(s)}$ by formula (8) and objective function $\mathcal{L}^{(s)}$ by formula (13). From steps 8 to 10, the partial derivatives $\nabla\overrightarrow{\pi}^{(s)}$, $\nabla\overrightarrow{\omega}^{(s)}$ and $\nabla\overrightarrow{\varphi}^{(s)}$ of $\mathcal{L}$ are calculated by formula (18), (14) and (16), respectively. Steps 12 and 13 present the update equations for $\overrightarrow{\omega}^{(s+1)}$, $\overrightarrow{\varphi}^{(s+1)}$, and $\overrightarrow{\pi}^{(s+1)}$. Steps 15 and 16 normalize each element of the vectors $\overrightarrow{\omega}^{(s+1)}$, $\overrightarrow{\varphi}^{(s+1)}$, and $\overrightarrow{\pi}^{(s+1)}$, respectively. In step 18, the objective function $\mathcal{L}^{(s+1)}$ is updated on the above mentioned parameters by formula (13). From steps 19 to 23, we first compare $|\mathcal{L}^{(s)} - \mathcal{L}^{(s+1)}|$ to ϵ, and then determine whether to continue this algorithm through the loop or stop. Finally, the results are returned in the step 24.

Algorithm 1 MIKE Algorithm

Input: *(1) Word Graph $G = (V, E)$; (2) Features of nodes A^V; (3) Parameter vector of node $\overrightarrow{\varphi}$; (4) Features of links A^E; (5) Parameter vector of link $\overrightarrow{\omega}$; (6) Topical importance vector of nodes $\overrightarrow{W}_z$; (7) Supervision matrix B; (8) Parameter vector of supervision information $\overrightarrow{\mu}$; (9) Parameters λ, δ, ρ, and ϵ.*

Output: *Ranking score vector of nodes $\overrightarrow{\pi}^*$.*

1: **Initialize:**
2: $\delta; \lambda; \rho; \epsilon$;
3: $s = 0$;
4: $\overrightarrow{\pi}^{(s)}; \overrightarrow{\omega}^{(s)}; \overrightarrow{\varphi}^{(s)}$;
5: **Calculate:**
6: $P^{(s)} = P(\overrightarrow{\omega}^{(s)}); \overrightarrow{r_z}^{(s)} = \overrightarrow{r_z}(\overrightarrow{\varphi}^{(s)})$;
7: $\mathcal{L}^{(s)} = \mathcal{L}(\overrightarrow{\omega}^{(s)}, \overrightarrow{\varphi}^{(s)}, \overrightarrow{\pi}^{(s)})$;
8: $\nabla\overrightarrow{\pi}^{(s)} = (\nabla\pi_1^{(s)}, \nabla\pi_2^{(s)}, ..., \nabla\pi_n^{(s)})$;
9: $\nabla\overrightarrow{\omega}^{(s)} = (\nabla\omega_1^{(s)}, \nabla\omega_2^{(s)}, ..., \nabla\omega_h^{(s)})$;
10: $\nabla\overrightarrow{\varphi}^{(s)} = (\nabla\varphi_1^{(s)}, \nabla\varphi_2^{(s)}, ..., \nabla\varphi_l^{(s)})$;
11: **Update:**
12: $\overrightarrow{\omega}^{(s+1)} = \overrightarrow{\omega}^{(s)} - \rho\nabla\omega^{(s)}; \overrightarrow{\varphi}^{(s+1)} = \overrightarrow{\varphi}^{(s)} - \rho\nabla\varphi^{(s)}$;
13: $\overrightarrow{\pi}^{(s+1)} = \overrightarrow{\pi}^{(s)} - \rho\nabla\pi^{(s)}$;
14: **Normalize:**
15: $\pi_i^{(s+1)} = \frac{\pi_i^{(s+1)}}{\sum_{j=1}^n\pi_j^{(s+1)}}; \omega_i^{(s+1)} = \frac{\omega_i^{(s+1)}}{\sum_{j=1}^l\omega_j^{(s+1)}}$;
16: $\varphi_i^{(s+1)} = \frac{\varphi_i^{(s+1)}}{\sum_{j=1}^h\varphi_j^{(s+1)}}$;
17: **Calculate:**
18: $\mathcal{L}^{(s+1)} = \mathcal{L}(\overrightarrow{\omega}^{(s+1)}, \overrightarrow{\varphi}^{(s+1)}, \overrightarrow{\pi}^{(s+1)})$;
19: **if** $|\mathcal{L}^{(s)} - \mathcal{L}^{(s+1)}| < \epsilon$ **then**
20: *stop* ; $\overrightarrow{\pi}^* = \overrightarrow{\pi}^{(s+1)}$;
21: **else**
22: $s = s + 1$; *go to step* 5;
23: **end if**
24: **return** $\overrightarrow{\pi}^*$

In general, the time complexity of MIKE is only of order $O(nl + mh)$, where m is the number of edges and n is the number of nodes in the graph. In the practical applications, l or h is far less than n.

5 EXPERIMENTAL EVALUATION

5.1 Experimental Setup

5.1.1 Datasets and Preprocessing. We evaluate our models using the research paper datasets collected by recent works on keyphrase extraction [8, 14, 15]. To the best of our knowledge, these datasets

Table 1: Summary of Datasets

Dataset	#Abs.	#ACdCtx.	#ACgCtx.	#AKPs	#uni-grams	#bi-grams	#tri-grams	#>tri-grams
WWW	425	15.45	18.78	4.87	680	1036	247	110
KDD	365	12.68	19.74	4.03	363	853	189	66

consist of the largest, publicly-available benchmark datasets of research paper abstracts containing both author-labeled keyphrases and citation network information (*cited* and *citing* contexts). The two research paper datasets are from two top-tier machine learning conferences: the ACM World Wide Web (WWW) and the ACM Knowledge Discovery and Data Mining (KDD). The cited and citing contexts for each paper were obtained from CiteSeer^x, the digital library portal for Computer Science related literature [24]. The datasets are summarized in Table 1 along with the number of papers (#Abs.), the average number of cited and citing contexts per paper (#ACdCtx. and #ACgCtx.), the average number of keyphrases per paper (#AKPs), and the number of unigrams, bigrams, trigrams and more than three grams (#>tri-grams), in each collection. Other details of these datasets are provided in the previous works[1]. For the evaluation phase, we used the author-labeled keyphrases obtained from the PDFs of the papers as our gold standard.

For data preprocessing, we use Python and the Natural Language Toolkit (NLTK) [4] package[2] to tokenize the raw text string, and then assign parts of speech (POS) to each word. Thirdly, we retain only nouns and adjectives based on POS filtering. Finally, we use Porter's stemmer [33][3] to stem all the words in order to remove words' suffix. Thus, the candidate words, which are used to construct the word graph, are obtained.

5.1.2 Evaluation Metrics. Almost all previous works on keyphrase extraction use precision (P), recall (R), F1-score (F1) and mean reciprocal rank (MRR) to evaluate the results [29]. Hence, we also keep our evaluation metric consistent. P, R and F1 are defined as follows:

$$P = \frac{\#_c}{\#_e}, \quad R = \frac{\#_c}{\#_s}, \quad F1 = \frac{2PR}{P+R}, \tag{20}$$

where $\#_c$ is the number of correctly extracted keyphrases, $\#_e$ is the total number of extracted keyphrases, and $\#_s$ is the total number of author-labeled standard keyphrases.

MRR is used to evaluate how the first correct keyphrase for each document is ranked. For a document d, MRR is defined as

$$MRR = \frac{1}{|D|} \sum_{d \in D} \frac{1}{rank_d}, \tag{21}$$

where D is the set of target documents and $rank_d$ refers to the rank of the first correct keyphrase with all extracted keyphrases. For the evaluation scores in our experiments, we first examine the average top-p predictions (*average p*), which refers to the average number of keyphrases for a particular above-mentioned dataset. For instance, average $p = 5$ for WWW and 4 for KDD. Besides, to make the comparison more convincing, we conduct experiments on top@5, top@10 and top@15 extracted keyphrases.

[1] http://www.cse.unt.edu/~ccaragea/keyphrases.html
[2] http://www.nltk.org/
[3] http://tartarus.org/martin/PorterStemmer/

5.1.3 Comparative Methods. To evaluate the performance of our method, we compare our method with following unsupervised methods (four of the five methods are graph-based methods):

(1) **TF-IDF.** TF-IDF method directly ranks each candidate word according to its tf-idf score (i.e., the term frequency-inverse document frequency). Although TF-IDF is the simplest of these comparative approaches, it can achieve good performance on some specific datasets, as introduced in the previous work [16].
(2) **TextRank [31].** TextRank model computes the score of each candidate word (i.e., the score of node, which reflects its importance) using structure information of word graph only from the *target document*. For each candidate word w_i, its score $R(w_i)$ is computed in an iterative manner until convergence using the following recursive formula:

$$R(w_i) = \lambda \sum_{j:w_j \to w_i} \frac{e(w_j, w_i)}{O(w_j)} R(w_j) + (1-\lambda), \tag{22}$$

where $e(w_j, w_i)$ is the weight of edge $w_j \to w_i$, and $O(w_j) = \sum_{w'} e(w_j, w')$ is the number of outbound edges. In our repeated experiments, $e(w_j, w_i)$ is the co-occurrence frequency of w_i and w_j in the target document.
(3) **ExpandRank [38].** ExpandRank is a TextRank extension that exploits a small *textual neighbourhood* in addition to the target document to enhance co-occurrence relations in the word graph. For a target document d, this approach first finds its nearest neighboring documents D_n using a similarity measure (e.g., cosine similarity). Then, the graph for d is built, where the candidate words are collected from the d and D_n. Finally, a modified TextRank model is used to compute the score of each candidate word, in which the weight of an edge $e(w_j, w_i)$ is recomputed as follows:

$$e(w_i, w_j) = \sum_{d_l \in D_n} cos(d, d_l) \cdot freq_{d_l}(w_i, w_j), \tag{23}$$

where $cos(d, d_l)$ is the cosine similarity between the *tf-idf* vectors of d and d_l [29], and $freq_{d_l}(w_i, w_j)$ is the co-occurrence frequency of w_i and w_j in document d_l.
(4) **CiteTextRank [14].** CiteTextRank is recently proposed by Gollapalli [2014] and uses the *citation context* in addition to the target document to enhance co-occurrence relations. This method is also a TextRank extension and similar to ExpandRank, in which the weight of an edge $e(w_j, w_i)$ is recomputed as follows:

$$e(w_i, w_j) = \sum_{k \in T} \sum_{c \in C_k} \gamma_k \cdot cos(d, c) \cdot freq_c(w_i, w_j), \tag{24}$$

where C_k is the set of contexts of type $k \in T$, and γ_k is the weight for contexts of type k. The types of contexts of d include the set of *cited, citing* contexts and itself.

(5) **Single-TPR [35].** Single-TPR, proposed by Sterckx [2015] and described in subsection 3.3 in detail, integrates the *full topical information* into the reset probability of PageRank. The recursive formula is represented in formula (5). Different from ExpandRank and CiteTextRank in which the transition probability of PageRank is redefined for exploiting different background knowledge to enhance the accuracy of the word graph, Single-TPR modifies the reset probability for using topical distribution of candidate words and documents.

Based on recent studies, these baselines comprise the state-of-the-art for keyphrase extraction [14]. Compared with these graph-based approaches, MIKE substantially modifies the both transition probability and reset probability. The TF-IDF, TextRank and ExpandRank are earlier representative methods. In order to clearly represent the experimental results, we select the best-performing method (BL*) from these three baselines with best-performing parameters for each dataset to compare with our method.

5.2 Used Multidimensional Information

In previous works on supervised methods, lots of traditional features of candidate words have been defined. These candidate features can be classified into four major categories, as shown in Fig. 1(1). Recently, a representative supervised method, proposed by Caragea [2014], designed some effective features based on citation context and used them in conjunction with traditional features for keyphrase extraction. These selected features include frequency features (such as tf-idf, tf-idf over a certain threshold), position features (such as first position, relative position), syntactic features (such as POS) and citation-context-based features (such as citation tf-idf, inCited, inCiting), around which many candidate features are checked by our experiments. Finally, the *tf-idf* (which is widely used, such as by Frank [1999], Caragea [2014]), *citation tf-idf* (used by Caragea [2014], Bulgarov [2015]) and *citation frequency* are chosen according to the variation coefficient of weights of a feature (i.e., values of $\varphi_i \in \overrightarrow{\varphi}$), which are obtained from the MIKE model. In particular, we choose the feature with a relatively small variation coefficient because it means that the influence of this feature is less variable and more stable in the MIKE. The experimental results also show that this feature selection strategy is effective. Note that our proposed *citation frequency* is defined as the number of times that this candidate occurs in citation contexts divided by the total in citation contexts and given research paper, and used to determine how important a given word is in the citation context.

In the experiments, we consider three edge features which are based on the co-occurrence frequency between candidate word pairs in the word graph. These three features have been used in the recent representative work [14], in which the weights of them are given in advance. In comparison, the weights are learned in our model. These three edge features are computed as follows:

$$a^E_{ijk} = \sum_{c \in C_k} cos(d, c) \cdot freq_c(w_i, w_j). \tag{25}$$

This formula is the part of formula (24). Note that when c is equal to d, we can get $a^E_{ijk} = freq_d(w_i, w_j)$, which is the co-occurrence frequency of w_i and w_j in document d.

This work gets the topic distribution of candidate word w in a given topic z (i.e., $Pr(w|z)$) and the topic distribution of a document d (i.e., $Pr(z|d)$) in the corpus through LDAModel [18][4].

The prior knowledge (i.e., supervised matrix B shown in the formula (11)) is represented with the pairwise preferences (i.e., relative importance relation between keyphrase and candidate). Specifically, we first randomly select a number of research papers with the author-annotated keyphrases for each paper. For a given paper, we then randomly choose some of words from it excluding words in keyphrases. Finally, a pairwise preference is composed of one word in keyphrases and the other word which is not in keyphrases, and corresponds to a row of the supervised matrix B, in which 1 represents the word in keyphrases and -1 represents the other word which is not in keyphrases.

5.3 Parameter Setting

Some parameters of MIKE are empirically set as follows:

(1) **Damping factor λ.** The factor λ is used to balance the probability of random jumping from the given node to a random node in the graph. The value of λ is empirically set to 0.85 in both Web surfing (PageRank [32]) and keyphrase extraction(graph-based methods, such as TextRank [31], TPR [26] and CiteTextRank [14]), and this is the value we are also using in our implementation.

(2) **Learning rate ρ.** The parameter ρ determines the convergence speed of parameters in learning process. A large ρ may speed up the learning process but it may also cause failures to converge to an optimal solution. In our experiments, we choose $\rho = 0.1$ as a tradeoff.

(3) **Tolerance ϵ.** The tolerance ϵ is used to determine whether to continue the Algorithm 1 through the loop or stop. We choose $\epsilon = 0.001$ in our experiments.

(4) **Weight ψ_p.** The weight ψ_p, in formula 1, used to adjust the final ranking score $R(p)$ of n-gram p. According to the ratio of the number of n-grams to total of keyphrases shown in Table 1, ψ_p is set as follows: $\psi_p = 1$, if $|p| = 1$; $\psi_p = 0.6$, if $|p| = 2$; $\psi_p = 0.3$, if $|p| \geq 3$.

Besides empirical parameters mentioned above, others will be introduced as follows.

In the word graph, each edges of graph is represented by the co-occurrence relation. That is to say, two candidate words (nodes) are connected if their corresponding lexical units co-occur within a window of maximum *window* words. Co-occurrence links express relations between syntactic elements, and represent cohesion indicators for a given text [31]. In our experiments, to illustrate the influence of the co-occurrence window *window*, we test values of this parameter *window* in the range of 2 to 10 and plot the results in Fig. 2. The best-performing setting is $window = 2$ (which is the same in TextRank) on two datasets, which is finally used in the comparison experiments. From the Fig. 2, compared with a larger window, a smaller window can help to improve performance of keyphrase extraction. This is probably explained by the fact that

[4] https://radimrehurek.com/gensim/

Table 2: The Comparison of MIKE with other Approaches at *average* $p=5$ on WWW and 4 on KDD Datasets.

Method	WWW				KDD			
	Precision	Recall	F1-score	MRR	Precision	Recall	F1-score	MRR
MIKE	**0.1480**	**0.1505**	**0.1492**	**0.3705**	**0.1607**	**0.1600**	**0.1603**	**0.3364**
CiteTextRank	0.1398	0.1433	0.1415	0.3314	0.1448	0.1434	0.1441	0.3155
Single-TPR	0.1310	0.1336	0.1323	0.2962	0.1289	0.1278	0.1283	0.2792
BL*	0.1243	0.1259	0.1251	0.2984	0.1256	0.1237	0.1247	0.2751

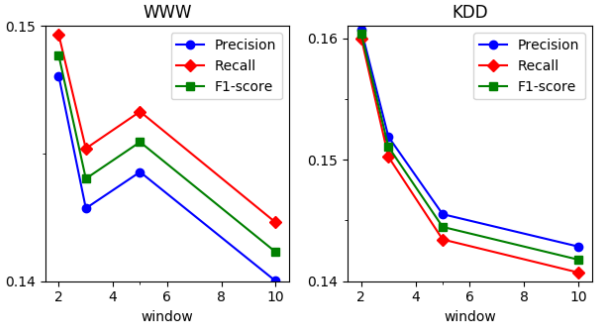

Figure 2: The influence of the co-occurrence window *window*

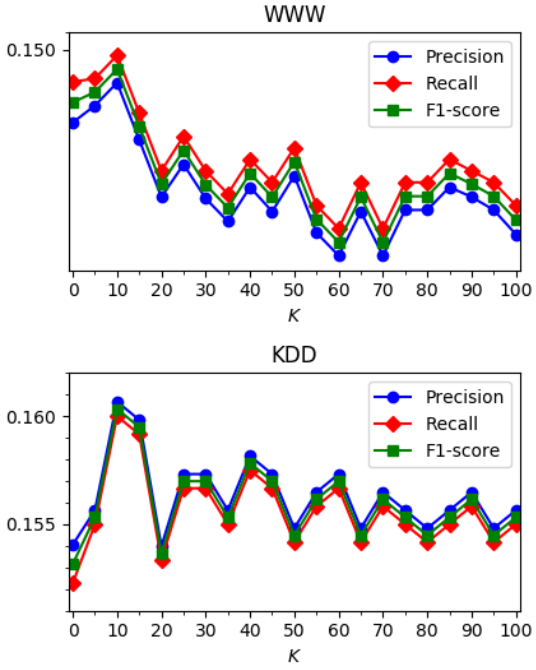

Figure 3: The influence of the number of topics K

a weak semantic relation between words is not strong enough to define a connection in the text graph [31].

To illustrate the influence of the number of topics in MIKE, we test values of this parameter K in the range of 0 to 100 and plot the results in Fig. 3. Note that the $K = 0$ means that the topical information has not been added to our model. We observe that the performance of MIKE is influenced by changes on the number of topics K. In general, the performance increases and then slowly decreases on both WWW and KDD datasets as K grows. The best-performing setting is $K = 10$ on both WWW dataset and KDD dataset, which is finally used in the comparison experiments.

In addition, the prior knowledge mentioned in formula (12) has been integrated into our model, which can be used to find the best-performing values of learning parameters $\overrightarrow{\omega}$ and $\overrightarrow{\varphi}$ in learning process and to help to choose the effective node features and edge features from lots of existing candidate features and edge features. In order to make the compared experimental results fairer, the prior knowledge is only used to help to choose the effective node features and edge features. In this choosing process, we set $\delta = 0.8$, which is used to balance the importance of modified PageRank (formula (9)) against of prior knowledge (formula (12)).

Finally, when the condition $|\mathcal{L}^{(s)} - \mathcal{L}^{(s+1)}| < \epsilon$ (i.e., Step 19 in Algorithm 1) is satisfied in this algorithm, the *maximum* and *average number of iterations* are 34 and 20.1 on WWW, and 33 and 19.8 on KDD, respectively.

5.4 Performance Comparison

We first compare the BL*, Single-TPR and CiteTextRank from previous researches with MIKE at average p ($p = 5$ on WWW and 4 on KDD datasets) and the results are shown in Table 2. Note that the average p are very close to the average numbers of keyphrases *AKPs* in given research papers (*AKPs* = 4.87 on WWW and 4.03 on KDD datasets), as shown in Table 1. The benefit is that the experiment can reflect real application environment. The results show that MIKE substantially outperforms other comparative approaches in terms of all performance measures on two datasets.

Secondly, we further conduct experiments to compare our model with two graph-based approaches, namely, Single-TPR and CiteTextRank, in terms of the accuracy of top@5, top@10 and top@15 extracted keyphrases. All the results are presented in Table 3. As the results show, our MIKE gets the best results in terms of all performance measures, indicating that our method indeed outperforms the other two approaches. In addition, the results show that the Precision and F1-score get smaller and the Recall and MRR become larger for each of these approaches as the number of extracted keyphrases increases. These changes are in accordance with the actual accident situation, and prove the correctness of our experiments.

Table 3: The Comparison of Approaches at @5, @10, @15 on both WWW and KDD Datasets

Method	WWW				KDD			
	Precision	Recall	F1-score	MRR	Precision	Recall	F1-score	MRR
MIKE@5	0.1480	0.1505	0.1492	0.3705	0.1401	0.1733	0.1549	0.3383
MIKE@10	0.0907	0.1814	0.1209	0.3785	0.0933	0.2242	0.1318	0.3476
MIKE@15	0.0681	0.2026	0.1019	0.3788	0.0742	0.2633	0.1158	0.3527
CiteTextRank@5	0.1398	0.1433	0.1415	0.3314	0.1288	0.1598	0.1426	0.3154
CiteTextRank@10	0.0871	0.1785	0.1170	0.3417	0.0879	0.2182	0.1254	0.3309
CiteTextRank@15	0.0649	0.1997	0.0980	0.3446	0.0656	0.2441	0.1034	0.3341
Single-TPR@5	0.1310	0.1336	0.1323	0.2962	0.1156	0.1428	0.1278	0.2879
Single-TPR@10	0.0874	0.1751	0.1166	0.3098	0.0849	0.2033	0.1197	0.3070
Single-TPR@15	0.0667	0.1987	0.0999	0.3130	0.0669	0.2366	0.1043	0.3113

Table 4: Sample Predictions for Different Datasets using Different Methods (Author-labeled keyphrases are shown in bold.)

1 Jena: implementing the semantic web recommendations (WWW, 2004; average $p = 5$)
Gold: jena, owl, rdf, rdql, semantic web, software architectures
MIKE: ***rdf***, ***jena***②, *rdf graph,* ***owl****④, ***semantic web***⑤
CiteTextRank: **rdf**, rdf graph, **owl***③, **jena**④, rdf recommend
Single-TPR: rdf graph, rdf recommend, web api, second-gener rdf, rdf toolkit

2 First-order focused crawling (WWW, 2007; average $p = 5$)
Gold: focused crawling, relational subgroup discovery
MIKE: subgroup discovery, relational subgroup, ***relational subgroup discovery***③, *focus,* ***focused crawling****⑤
CiteTextRank: focused web, subgroup discovery, relational subgroup, **focused crawling***④, web crawling
Single-TPR: first-order focus, **focused crawling***②, first-order focused crawler, focused crawler, focused web

3 Hierarchical model-based clustering of large datasets through fractionation and refractionation (KDD, 2002; average $p = 4$)
Gold: clustering, fractionation, model-based clustering, refractionation
MIKE: ***clustering***, ***model-based clustering***, ***fractionation***③, *hierarchical clustering*
CiteTextRank: **clustering**, **model-based clustering**, hierarchical clustering, non-parametric clustering
Single-TPR: **clustering**, **model-based clustering**, hierarchical clustering, non-parametric clustering

4 Finding a team of experts in social networks (KDD, 2009; average $p = 4$)
Gold: social networks, team formation
MIKE: team, ***team formation***②, *team formation problem,* ***social networks***④
CiteTextRank: problem, formation problem, team, **team formation**④
Single-TPR: problem, formation problem, combinatorial problems, team

In conjunction with the results shown in Table 2, we can clearly see the advantage of integrating all these different types of information into a unified model. Some anecdotal examples are shown in Table 4. The predictions obtained by different methods along with human-picked "gold" keyphrases are listed in this table. As can be seen, there is a high overlap between the "gold" and predicted keyword sets by MIKE. Besides, comparing with the Single-TPR and CiteTextTank, MIKE can improve the final ranking score of author-labeled keyphrases. For example, some extracted keyphrases labeled by the underline and circled position number achieve the high ranking using MIKE, but low ranking using the other two graph-based approaches. The number of counter-examples labeled by the asterisk and circled position number is few (Only two are "owl" in Sample 1 and "focused crawling" in Sample 2.). These examples can further indicates that the multidimensional information help to gain better performance in keyphrase extraction.

6 CONCLUSIONS

In this study, we presented a graph-based random-walk algorithm named MIKE for extracting keyphrases from scientific research papers. In particular, we integrated five different types of information, which have been widely used in the different approaches, into a unified model to capture the latent mutual influences of all these information on a candidate. These information includes the *features of candidate words*, *prior knowledge* (used in supervised approaches), *structure of word graph*, *features of links* (used in graph-based approaches), and *topical information* (used in supervised or graph-based approaches). Besides being capable of leveraging multidimensional information, MIKE learned the node and edge weights by our theoretically sound optimization model instead of assigning empirical weights, which distinguishes it from the other graph-based approaches (e.g., CiteTextRank). In addition, MIKE is a flexible parametric model in which we can easily add any

new features of nodes or links to it. Our experimental results have shown that the proposed algorithm can significantly outperform the state-of-the-art graph-based approaches on both WWW and KDD datasets.

In future, we plan to conduct more comprehensive experiments to investigate the effectiveness of other features of candidates and edges in the word graph, and evaluate MIKE for other types of documents, e.g., Biology and Chemistry.

ACKNOWLEDGMENTS

This work was partially supported by grants from the National Natural Science Foundation of China (Grant No. U1533104, U1633110).

REFERENCES

[1] Kolawole John Adebayo, Di Caro Luigi, and Boella Guido. 2016. A Supervised KeyPhrase Extraction System. In *Proceedings of SEMANTICS*. ACM, 57–62.
[2] Abdelghani Bellaachia and Mohammed Al-Dhelaan. 2014. HG-RANK: A Hypergraph-Based Keyphrase Extraction for Short Documents in Dynamic Genre. In *Proceedings of 4th Workshop on Making Sense of Microposts co-located with WWW*. ACM, 42–49.
[3] Gabor Berend. 2014. Exploiting Extra-textual and Linguistic Information in Keyphrase Extraction. *Natural Language Engineering* 22, 1 (2014), 73–95. https://doi.org/10.1017/S1351324914000126
[4] Steven Bird, Ewan Klein, and Edward Loper. 2009. . O'Reilly Media.
[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3 (2003), 993–1022.
[6] Florian Boudin. 2013. A Comparison of Centrality Measures for Graph-based Keyphrase Extraction. In *Proceedings of IJCNLP*. 834–838.
[7] Florian Boudin. 2015. Reducing Over-generation Errors for Automatic Keyphrase Extraction using Integer Linear Programming. In *Proceedings of Workshop on Novel Computational Approaches to Keyphrase Extraction*. ACL, 19–24.
[8] Florin Bulgarov and Cornelia Caragea. 2015. A Comparison of Supervised Keyphrase Extraction Models. In *Proceedings of WWW*. ACM, 13–14.
[9] Cornelia Caragea, Florin Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014. Citation-enhanced Keyphrase Extraction from Research Papers: A Supervised Approach.. In *Proceedings of EMNLP*. ACL, 1435–1446.
[10] Zhuoye Ding, Qi Zhang, and Xuanjing Huang. 2011. Keyphrase Extraction from Online News Using Binary Integer Programming. In *Proceedings of AFNLP*. 165–173.
[11] Kathrin Eichler and Günter Neumann. 2010. DFKI KeyWE: Ranking Keyphrases Etracted from Scientific Aarticles. In *Proceedings of International Workshop on Semantic Evaluation*. ACL, 150–153.
[12] Corina Florescu and Cornelia Caragea. 2017. A Position-Biased PageRank Algorithm for Keyphrase Extraction. In *Proceedings of AAAI*. AAAI, 4923–4924.
[13] Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-specific Keyphrase Extraction. In *Proceedings of IJCAI*. Morgan Kaufmann, 668–673.
[14] Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting Keyphrases from Research Papers using Citation Networks. In *Proceedings of AAAI*. AAAI, 1629–1635.
[15] Sujatha Das Gollapalli, Xiaoli Li, and Peng Yang. 2017. Incorporating Expert Knowledge into Keyphrase Extraction. In *Proceedings of AAAI*. AAAI, 3180–3187.
[16] Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in Unsupervised Keyphrase Extraction: Making Sense of the State-of-the-art. In *Proceedings of COLING*. ACM, 365–373.
[17] Kazi Saidul Hasan and Vincent Ng. 2014. Automatic Keyphrase Extraction: A Survey of the State of the Art. In *Proceedings of ACL*. ACL, 1262–1273.
[18] Matthew D. Hoffman, David M. Blei, and Francis Bach. 2010. Online Learning for Latent Dirichlet Allocation. In *Proceedings of Neural Information Processing Systems*. MIT Press, 856–864.
[19] Thomas Hofmann. 1999. Probabilistic Latent Semantic Indexing. In *Proceedings of SIGIR*. ACM, 50–57.
[20] Anette Hulth. 2003. Improved Automatic Keyword Extraction Given More Linguistic Knowledge. In *Proceedings of EMNLP*. ACL, 216–223.
[21] Xin Jiang, Yunhua Hu, and Hang Li. 2009. A Ranking Approach to Keyphrase Extraction. In *Proceedings of SIGIR*. ACM, 756–757.
[22] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2013. Automatic Keyphrase Extraction from Scientific Articles. *Language Resources and Evaluation* 43, 3 (2013), 723–742. https://doi.org/10.1007/s10579-012-9210-3
[23] Thomas K Landauer, Peter W. Foltz, and Darrell Laham. 1998. An Introduction to Latent Semantic Analysis. *Discourse Processes* 25, 2-3 (1998), 259–284. https://doi.org/10.1080/01638539809545028
[24] Huajing Li, Isaac G. Councill, Levent Bolelli, Ding Zhou, Yang Song, Wang-Chien Lee, Anand Sivasubramaniam, and C. Lee Giles. 2006. CiteSeer^x-A Scalable Autonomous Scientific Digital Library. In *Proceedings of International Conference on Scalable Information Systems*. ACM.
[25] Xin Li and Fei Song. 2015. Keyphrase Extraction and Grouping based on Association Rules. In *Proceedings of FLAIRS*. AAAI, 181–186.
[26] Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic Keyphrase Extraction via Topic Decomposition. In *Proceedings of EMNLP*. ACL, 366–376.
[27] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to Find Exemplar Terms for Keyphrase Extraction. In *Proceedings of EMNLP*. ACL, 257–266.
[28] Patrice Lopez and Laurent Romary. 2010. HUMB: Automatic Key term Extraction from Scientific Articles in GROBID. In *Proceedings of Workshop on Semantic Evaluation*. ACL, 248–251.
[29] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.
[30] Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009. Human-competitive Tagging using Automatic Keyphrase Extraction. In *Proceedings of EMNLP*. ACL, 1318–1327.
[31] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing Order into Text. In *Proceedings of EMNLP*. ACL, 404–411.
[32] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The Pagerank Citation Ranking: Bringing Order to the Web*. Technical Report. Stanford InfoLab. 1–14 pages.
[33] M.F. Porter. 2006. An Algorithm for Suffix Stripping. *Program: Electronic Library and Information Systems* 40, 3 (2006), 211–218. https://doi.org/10.1108/00330330610681286
[34] Lucas Sterckx, Cornelia Caragea, Thomas Demeester, and Chris Develder. 2016. Supervised Keyphrase Extraction as Positive Unlabeled Learning. In *Proceedings of EMNLP*. ACL, 1924–1929.
[35] Lucas Sterckx, Thomas Demeester, and Johannes Deleu. 2015. Topical Word Importance for Fast Keyphrase Extraction. In *Proceedings of WWW*. ACM, 121–122.
[36] Takashi Tomokiyo and Matthew Hurst. 2003. A Language Model Approach to Keyphrase Extraction. In *Proceedings of ACL Workshop on Multiword Expressions*. ACL, 33–40.
[37] Peter D. Turney. 2000. Learning Algorithms for Keyphrase Extraction. *Information Retrieval Journal* 2, 4 (2000), 303–336.
[38] Xiaojun Wan and Jianguo Xiao. 2008. Single Document Keyphrase Extraction using Neighborhood Knowledge. In *Proceedings of AAAI*. AAAI, 855–860.
[39] Chen Wang and Sujian Li. 2011. CoRankBayes: Bayesian Learning to Rank under the Co-training Framework and Its Application in Keyphrase Extraction. In *Proceedings of CIKM*. ACM, 2241–2244.
[40] Fang Wang, Zhongyuan Wang, Senzhang Wang, and Zhoujun Li. 2014. Exploiting Description Knowledge for Keyphrase Extraction. In *Proceedings of PRICAI*. 130–142.
[41] Rui Wang, Wei Liu, and Chris McDonald. 2015. Corpus-independent Generic Keyphrase Extraction using Word Embedding Vectors. In *Proceedings of DL-WSDM*. ACM, 39–46.
[42] Senzhang Wang, Lifang He, Leon Stenneth, Philip S. Yu, and Zhoujun Li. 2015. Citywide Traffic Congestion Estimation with Social Media. In *Proceedings of ACM SIGSPATIAL*. ACM, 1–10.
[43] Wei Wei, Bin Gao, Tieyan Liu, Taifeng Wang, Guohui Li, and Hang Li. 2016. A Ranking Approach on Large-scale Graph with Multidimensional Heterogeneous Information. *IEEE Transactions on Cybernetics* 46, 4 (2016), 930–944. https://doi.org/10.1109/TCYB.2015.2418233
[44] Zhang Weinan, Ming Zhaoyan, Zhang Yu, Liu Ting, and Chua Tatseng. 2015. Exploring Key Concept Paraphrasing based on Pivot Language Translation for Question Retrieval. In *Proceedings of AAAI*. AAAI, 410–416.
[45] Wen T. Yih, Joshua Goodman, and Vitor R. Carvalho. 2006. Finding Advertising Keywords on Web Pages. In *Proceedings of WWW*. ACM, 213–222.
[46] Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter. In *Proceedings of EMNLP*. ACL, 836–844.
[47] Wei Zhang, Wei Feng, and Jianyong Wang. 2013. Integrating Semantic Relatedness and Words' Intrinsic Features for Keyword Extraction. In *Proceedings of IJCAI*. Morgan Kaufmann, 139–160.
[48] Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming Li. 2011. Topical Keyphrase Extraction from Twitter. In *Proceedings of ACL*. ACL, 379–388.
[49] Yu Zheng. 2015. Methodologies for Cross-domain Data Fusion: An Overview. *IEEE Transactions on Big Data* 1, 1 (2015), 16–34. https://doi.org/10.1109/TBDATA.2015.2465959
[50] Maxim Zhukovskiy, Gleb Gusev, and Pavel Serdyukov. 2014. Supervised Nested PageRank. In *Proceedings of CIKM*. ACM, 1059–1068.