IJICTR International Journal of Information & Communication Technology Research

Volume 7- Number 4- Autumn 2015 (55-64)

Keyphrase Ranking Based on Second Order Co-Occurrence Analysis

Hosein Shahsavar Haghighi
Department of Computer Engineering
Malek Ashtar University
Tehran, Iran
Shahsavar@mut.ac.ir

Mojtaba Hoseini
Department of Computer Engineering
Assistant Professor Malek Ashtar University
Tehran, Iran
mojtabahoseini@mut.ac.ir

Jamshid Shanbehzadeh
Department of Computer Engineering
Associate Professor Kharazmi University
Tehran, Iran
jamshid@khu.ac.ir

Received: April 4, 2015- Accepted: August 18, 2015

Abstract— **State-of-the-art researches in unsupervised automatic keyphrase extraction focused on graph analysis. Keyphrase ranking is critical step in graph-based approaches. In this paper, we follow two main purposes including choice of good candidate phrases and computing importance of candidate phrase by considering the mutual information between words. Our documents representation improves the process of candidate phrases selection by constructing a single graph for all documents in the collection. We enjoy from parallel minimum spanning tree to prune irrelevant edge relations. We also consider second order co-occurrence of words by point-wise mutual information as a similarity measure and importance of terms to increase the performance of keyphrase ranking. We formed a single graph of co-occurrence network for all documents in the collection and analyze co-occurrence network with different settings. We compare our method with three baseline approaches of keyphrase extraction. Experimental results show that applying second order co-occurrence analysis improves keyphrases identification accuracy.**

Keywords-component; graph analysis, similarity measure, point-wise mutual information, co-occurrence networks, keyphrase ranking

I. INTRODUCTION

Keyphrase includes terms in a document that give a brief summary of its content and main concepts as the document is related to them. This task is used widely in many areas of information extraction such as a digital library[1],[2]. It's a critical task in natural language processing, document categorization and clustering [3],[4].

Although there are some structured texts, which are labeled with keyphrases by the authors, other resources such as web pages and social media content are still semi-structured text. They include different domains such as scientific, news, sports and blogs[5]. There are two overall categories for extracting keyphrase: supervised and unsupervised. The supervised approach [1] regards keyphrase extraction as a classification task, in which a model is trained to decide whether a candidate phrase is a keyphrase or not. Supervised methods require a document set with human-assigned keyphrases as training set [6]. The first task in supervised keyphrase extraction as a classification is carried out by [1]. The supervised methods need manually annotated training set which is time-consuming[7],[8]. In this paper, we focus on unsupervised method. As the manual tagging or providing a comprehensive list of human labeled keyphrases is time-consuming, we apply unsupervised learning in this study.

In the unsupervised approach [9], Offered a graph-based ranking method which builds a word graph according to word co-occurrences within the document. It uses PageRank as a random walk technique to measure word importance[6]. Existing graph-based methods compute an importance score for each word. Most of unsupervised method has faced two challenges.

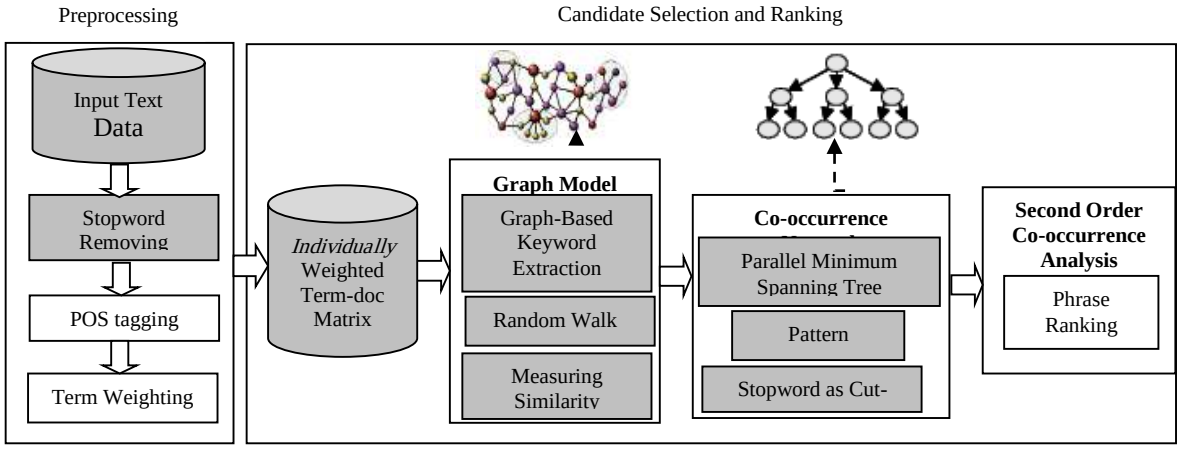

Fig 1. Keyphrase extraction system

First, selecting proper candidate phrase, especially candidate phrase with the length of 3-4 using background knowledge to understand deeper structure of document[10],[11]. This point of view can be facilitated by involving term weighting methods, constitution of parallel minimum spanning tree for eliminating original graph to get proper relevant edge and use page rank to propagate importance of words across the graph. The second challenge is computing importance of candidate phrase with a length of 3-4 words. We proposed a ranking method for candidate phrase. We compute words similarity based on second order co-occurrence analysis. This ranking methods assigned a weight to each candidate phrase. It helps us to find candidate keyphrases and non-keyphrases more precisely[12].

II. Related Works

TextRank [9], is a scoring algorithm of random walk modelling that represents text by a graph. Each vertex corresponds to a word type and its weight is the number of times the corresponding word types co-occur within a certain window. SingleRank [13] is similar to TextRank [9] with several differents. First, its edge weight is equal to the number of corresponding words co-occur. Second, TextRank filter the word type based their part-of-speech, whiles [13] does not consider such limitation. Finally, it uses a window size of 10. ExpandRank [13] is another extension of TextRank [9]. For a document d, it exploits K similar documents in corpus, by using a similarity measure (e.g., cosine similarity). Then, it builds a graph for document d by using the co-occurrence analysis of the words of these K neighbors. Once the graph is constructed then the rest of the procedure is same to SingleRank. Z. Liu et al. (Z. Liu, Li, Zheng, &Sun, 2009) Proposed a cluster-based approach called KeyCluster to cluster candidate words based on their semantic relationship.

Three clustering algorithms are used of which spectral clustering yields the best score. Once the clusters are formed, one representative word, called an exemplar term, is picked from each cluster. Finally, KeyCluster extracts from the document all the longest n-grams starting with zero or more adjectives and ending with one or more nouns, and if such an n-gram includes one or more exemplar words, it is selected as a keyphrase.

In the final step, candidate phrases are sorted by their scores. For example, If selected keyphrase includes one or more of the top-ranked keywords words [15],[16] or sum of the ranking scores of its words sequence which causes it have a top score[13].

III. Motivation

In this paper, we proposed an unsupervised method for automatic keyphrase extraction. We deal with two challenges. Most of unsupervised methods deal with two challenges. First, they do not understand deeper structure of document. As many recent work has focused on algorithmic development, we want to use background knowledge to understand deeper structure of document. This point of view can be facilitated by involving deeper knowledge of document Such as individual term weighting methods, constitution of minimum spanning tree for eliminating original graph to get proper relevant edge and use page rank to propagate importance of words across the graph.

most of keyphrases have length of 1–4 words [17]. Keyphrases are normally composed of nouns and adjectives, but may occasionally contain adverbs or containing Conjunction, prepositions, hyphens and apostrophes[18]. Those often used in the documents to be one of the following forms:

- Simple key words (e.g. “phrase”, “topic”)
- Noun phrases (e.g. “page rank”, “key word”, “topic modelling”).

Second challenge is ranking candidate phrase with a length of 3-4 words. It's possible to propose a method to handle the large number of documents, by using statistical methods, especially those that semantically improve vector space representation of term-document matrix. It will help system to distinguish candidate keyphrases and non-keyphrases. We proposed a new ranking method for candidate phrases with the length of 3-4 word to find prior keyphrases. We enjoy point-wise mutual information by considering second order co-occurrence of words.

IV. Methodology

Similar to most of unsupervised approaches, the proposed method comprises three main steps: Pre-processing, Candidate Selection and Candidate Ranking[19]. Fig.1 represents general framework of proposed method.

A. Proposed Algorithm

We describe an algorithm based on forming distributed minimum spanning tree of single corpus graph(See Fig.2). First, we do pre-processing steps on document collection. Then term weighting methods applied to extract individual score for each word (feature). After computing edge importance by point wise mutual information co-occurrence analysis of two word in whole corpus, co-occurrence network is created to show candidate phrases then random walk method such as Page Rank applies to propagate score of the word and edges across network. Nodes with high weights are keywords. In this step, we prune unnecessary relation between words using minimum spanning tree. we find candidate phrase considering three conditions in section (A.2). Finally, we use candidate phrase ranking method to find prior keyphrases.

Function KPE-PMST Returns A List Of Keyphrase

Inputs: $D = \{d_1, d_2, \dots, d_n\}$,
K // **number of extracted phrase**
TWM // **term weighting method**
WS // **windows size**
Output:
KPL // **list of key phrases**

result ← *remove-stopword (d, stopwords list)*
(result, pos) ← *stanford_pos_tager(result)*
eliminate words space with specific *pos* tag
result ← *reduce _space (result-{adj, nn, nns, nnp})*
return a weighted term vector by deploying different weighing methods
w_t ←*individual_weighting (result, twm)*
g(e,v) ← *conduct graph*(w_t, *result)*
for each vertex v_i, v_j *in v*

$$e_{ij} \leftarrow \frac{v_i \cap v_j}{v_i \cup v_j} = \frac{co-occurrence(v_i, v_j, ws)}{tf(v_i)+tf(v_j)}$$

keyword scoring: return an importance weight for each word in v
(keyword score, $R(t_j)$*)* ← page_rank (*e, v,* w_t)
candidate selection: return candidate phrases
*(*candidate phrases*)* ← *candidate selection (2-gram,3-gram,* D*)*
remove phrase from candidate list
for each phrase cp in candidate phrases
if (cp^ stopwords list) <>null
remove (cp)
(pmst_tree(e',v)) ← *parallel minimum spanning tree (g (e, v))*

candidate ranking: SOC_PMI(.) is our ranking method return ranked kephrase and KPL is a list of keyphrases
for each phrase ***cp*** *in candidate* phrases
if (cp ∩ pmst_tree(e',v)) ==null
remove (cp)
else
KPL = KPL ∪ $SOC\_PMI(cp)$

Fig2. Pseudo code of proposed algorithm

1) Preliminaries

Let $D = \{d_1, d_2, \dots, d_n\}$ be a set of document and $V = \{t_1, t_2, \dots, t_m\}$ is vocabulary which be set of all terms in corpus. For a document di, corresponding term weights vector is represented as $d_i = \{w_1, \dots, w_{|di|}\}$, where wi indicates how much w_i contributes to document d_i.

2) Pre-Processing

In Pre-Processing phase, we do three common tasks which include:

- We apply a stop word list to remove ineffective and common words[14].
- We consider words with certain part of-speech tags (e.g., nouns, adjectives) as candidate keywords[15]. In our experiments, we apply the following tags captured from Stanford POS Tagger [20] as candidates words: Noun, Proper Noun and Adjective.
- In the third step, we build a weighted term-matrix for all terms in the corpus based on their "individual importance" . We also consider three measures for this purpose namely, Entropy, Mutual information and variance approach[20].

3) Graph Model

The graph model is based on vector space model [21] by weighting each term according to its degree of "individual importance" regardless of term associations. Term-document weighting method such as, TF-Tdf weighting set the weight of each term individually without considering its correlation with other terms and their occurrences[22]. As a result, such methods omit latent and valuable information among the terms. Due to the above, we first set the weight of each term in collection as "individual importance" then compute "association's importance" by constructing co-occurrence network and measuring similarity as an edge weight between pair-terms by co-occurring analysis[19], [23]. We organize the single graph for all documents and their constructing units (words).

4) Keyword Ranking

After conducting graph and assigning an individual weight to each vertex, edge similarity between two vertexes is calculated by measuring number of co-occurrence between them within all documents[24]. When the network co-occurrence is formed, the edge weight is propagated across the network using random walk algorithm

5) Term Weighing Methods

In the first step, we set "individual importance" of each word. We use the following Tf-Tdf weighing method.

$$TF\_idf(t_i) = tf(t_i, d_i) \times idf(t_i, D) \quad (1)$$

Inverse document frequency weight is the most standard To separating terms among documents as follows[25]:

$$idf(t, D) = log \frac{|D|}{1 + |\{d \in D: t \in d\}|} \quad (2)$$

Some of frequent terms are less relevant to document concepts because they are irreverent to whole collection except a few documents[26], [27]. We exploit the three weighting methods, including mutual information, Entropy and term variance-based method and compared with TF_idf in table3 and table.4. Mutual information is use to compute the feature importance by measuring the statistical dependence between the feature and the document collection. It computes term weight ti as follows[25]:

$$I(t_i, D) = p(t_i) \sum_{d_j \in D} p(d_j|t_i) \left(\log \frac{p(d_j|t_i)}{p(d_j)} \right) \quad (3)$$

Where $P(d_i)$ is occurring probability of document dj in collection. $P(t_i)$ shows the occurring probability of t_i in the document collection, $P(d_i|t_i)$ is the probability that document d_j contains term t_i. Entropy is another measuring method which can compute the weights for features [28],[25]. It is based on uncertainty theory and illustrated in following equation:

$$EN(t_i) = 1 + \frac{1}{\log|D|} \sum_{j=1}^{|D|} p(d_j|t_i) \log p(d_j|t_i) \quad (4)$$

[29]proposed term variance approach which is computed as follows[25]:

$$Q(t_i) = \sum_{j=1}^{|D|} O_{d_j,t_i}^2 - \frac{1}{|D|} \left[\sum_{j=1}^{|D|} O_{d_j,t_i}^2 \right] \quad (5)$$

where $O_{di,ti}^2$ represents the frequency that the term ti occurs in the document d_j[26].

6) Measuring Similarity

Each word is a vertex of the graph. After computing the individual importance of the words by term weighting methods, a term-document matrix with initial weights is prepared[23]. Then, the relation between vertexes is captured by measuring the co-occurrence count of them within a sliding window N[30]. We extend this measure in section.8 (Ranking Method). In [9] shown that the edge direction of graph does not influence the accuracy of keyphrase extraction so much.

7) Candidate Selection

We apply three-stage filter for candidate selection. This filter is applied for each term and its neighbors in document collection.

Since keyphrases are usually noun phrases, we only add adjectives, nouns and proper noun in word graph. We apply the following pattern for candidate selection[19]:

(adjective)*(noun)+.

Using stop words as contour phrases. Fig.3 illustrates part of article in CNN news website with drawing stops words as phrases cut-off window. The green highlighted area are candidate keyphrase. It can be inferred that using stop words and conjunctions as cut-off widows improve detection of proper candidate keyphrases.

Yahoo wants to make its Web e-mail service a place you never want to -- or more importantly – have to leave to get your social fix. The company on Wednesday is releasing an overhauled version of its Yahoo Mail Beta client that it says is twice as fast as the previous version, while managing to tack on new features like an integrated Twitter client, rich media previews and a more full-featured instant messaging client. Yahoo says this speed boost should be especially noticeable to users outside the U.S. with latency issues, due mostly to the new version making use of the company's cloud computing technology. This means that if you're on a spotty connection, the app can adjust its behavior to keep pages from timing out, or becoming unresponsive. Besides the speed and performance increase, which Yahoo says were the top users requests, the company has added a very robust Twitter client, which joins the existing social-sharing tools for Facebook and Yahoo. You can post to just Twitter, or any combination of the other two services, as well as see Twitter status updates in the update stream below. Yahoo has long had a way to slurp in Twitter Feeds, but now you can do things like reply and retweet without leaving the page.

Fig 3. CNN news with drawing stops words as cut-off window

We draw minimum spanning tree from original co-occurrence network to find a tree whose sum of vertexes' weight is minimal and covered all vertices in the graph. We use parallel implementation of Boruvka's Minimum Spanning Tree Algorithm by S.Chung et al. [31].

8) Ranking Method

Given the three ranking functions for comparison: First technique is similar to [4], we can rank candidate keyphrases by $\sum_{w_j \in K} R(w_j)$, where $R(w_j)$ is the score assigned to word w by a keyword ranking method. We consider another ranking technique by $\sum_{w_j \in K} \log R(w_j)$. This technique is similar to former with the difference that calculates the logarithm of $R(w_j)$. In [1] Turney introduced point-wise mutual information an unsupervised learning methods for recognizing word similarity by using Point-wise Mutual Information. We proposed a similarity measure between words using second order co-occurrence point-wise mutual information**.** let $V = \{t_1, t_2, \dots, t_m\}$be the set of all unique words which occur in the documents collections *D*. $D = \{d_1, d_2, \dots, d_n\}$ denotes a large corpus of text containing *n* documents and vocabulary V contains *m* unique words which occur in the *D*. Let t_1 and t_2 be the two vertices of graph G=(V,E). We want to determine the semantic similarity between t_1 and t_2. as we know, the majority of keyphrases have length of 1 to 4 words [17]. Candidate phrases with 1 and 2 words are easily identified, But for the rest of candidate phrase with 3 or 4 words long, we obliged to offer a different ranking approach. After preprocessing steps, for recognizing triple candidate phrase, we will compute the similarity between two words t_1, t_2 which no direct connection established between them. We set a parameter α, which determines how many words can be included in the context window. The window also contains the target word t_1, t_2 themself. The steps in determining the semantic similarity consider the corpus and some functions related to frequency counts. We define *frequency* function for each words in V as $f(t_i)$ which says how many times t_i occurs in the entire corpus[32].

We also consider another frequency function named Co-occurrence function for two words in corpus if exists a connection between them (i.e. if two connected with edge in corpus graph) and shown with $C(t_i, t_j)$. It tells us how many times t_i, t_j Co-occurred together in a window size α. We proposed point-wise mutual information based Co-occurrence *function (SOC-PMI)* only for those words having $C(t_i, t_j) > 0$,

$$f^{pmi}(t_i, t_j) = \log_2 \frac{C(t_i, t_j) * m}{f(t_i) * f(t_j)} \quad (6)$$

Where $f(t_i) * f(t_j) > 0$ and m is total number of tokens in corpus D as mentioned earlier. For word v_1, we define a set of neighbor words as $i = 1, 2, \dots, \mu_1$, which $f^{pmi}(t_i, v_1) > 0$ and having μ_1 top-most value where:

$$\forall\, i = 1, \dots, \mu_1 \mid f^{pmi}(t_i, v_1) > f^{pmi}(t_{1+i}, v_1) \quad (7)$$

As a same way, for word v_2, we define a set of neighbor words as $j = 1, 2, \dots, \mu_2$, which $f^{pmi}(t_j, v_2) > 0$ and having μ_2 top-most value where

$$\forall\, j = 1, \dots, \mu_2 \mid f^{pmi}(t_j, v_2) > f^{pmi}(t_{j+1}, v_2) \quad (8)$$

Value of μ_1 *and* μ_2 depend on word *v*. We multiply the *SOC-PMI* function for all word as following:

$$f^{\alpha}(v_1) = \prod_{i=1}^{\mu_1} \left(\frac{f^{pmi}(t_i, v_2)}{\beta_{t_i} * \beta_{v_2}} \right) \quad (9)$$

Where $f^{pmi}(t_i, v_2) > 0$ and $f^{pmi}(t_i, v_1) > 0$ and β_{t_i}, β_{v_2} are branching coefficient (i.e. number of nodes with context windows size of 2 with t_i and v_2). It multiplies PMI values of all the semantically close words of v_2 (Note that we call it semantically-close because each t_i co-occurs with v_2 in context windows α, has high PMI value with v_2) but it doesn't guarantee t_i co-occurs with v_1 within the window size. in the same way, for word v_2, the *SOC-PMI* function is:

$$f^{\alpha}(v_2) = \prod_{j=1}^{\mu_2} \left(\frac{f^{pmi}(t_j, v_1)}{\beta_{t_j} * \beta_{v_1}} \right) \quad (10)$$

Where $f^{pmi}(t_j, v_1) > 0$ and $f^{pmi}(t_j, v_2) > 0$. It multiplies PMI values of all the semantically close words of v_1 (Note that we call it semantically-close because each word t_j co-occurs with v_1 in context windows α, has high PMI value with v_1) but it doesn't guarantee t_i co-occurs with v_2 within the window size. Finally, we define the *semantic PMI similarity* function between two words v_2 and v_1:

$$S(v_1, v_2) = \frac{f^{\alpha}(v_1)}{\mu_1} + \frac{f^{\alpha}(v_2)}{\mu_2} \quad (11)$$

We use from Md. Aminul Islam and Diana Inkpen work [33] for choosing value of μ_1, μ_2. It related to how many times the word, v_i, v_1 appears in the corpus. They define μ_i as:

$$\mu_i = \left(log\left(f^t(v_i)\right)\right)^2 \frac{\log_2 n}{\delta} \quad (12)$$

We also define a new method for determining top most neighbors of each node as following:

$$\mu_i = log\left(\frac{f^t(v_i)}{\beta_{v_i} * IDF_{v_i}}\right) * \frac{\beta_{v_i}}{\delta} \quad (13)$$

Where IDF_{v_i} is inverse document frequency of v_i and β_{v_i} is number of distinguished neighbors of v_i and δ is a constant for all experiments (we used δ= 5). The value of δ depends on size of the corpus. If smaller corpus is used, the value of δ should be smaller.

V. EXPERIMENTS

A. Database

We consider two corpora for constructing our model. First was built [13]. This dataset includes 308 news articles in DUC2001 [34]. Each article have 10 manually annotated keyphrases. The second corpus was built by [35] contains 2,000 abstracts of research articles and 19,254 manually annotated keyphrases. We remove the articles shorter than 100 words. After pre-processing steps, we build the vocabulary by selecting 20,000. We learn model by taking each article as a document.

B. Metrics

Despite the output of most keyphrase extraction systems are yet weak in comparison with other NLP-Branches, it doesn't indicate the performance is low. Even different manual annotators can assign different keyphrases to the same documents and rank extracted phrase arbitrarily. We choose traditionally NLP-Tasks metrics. It includes precision, recall, F-measure.

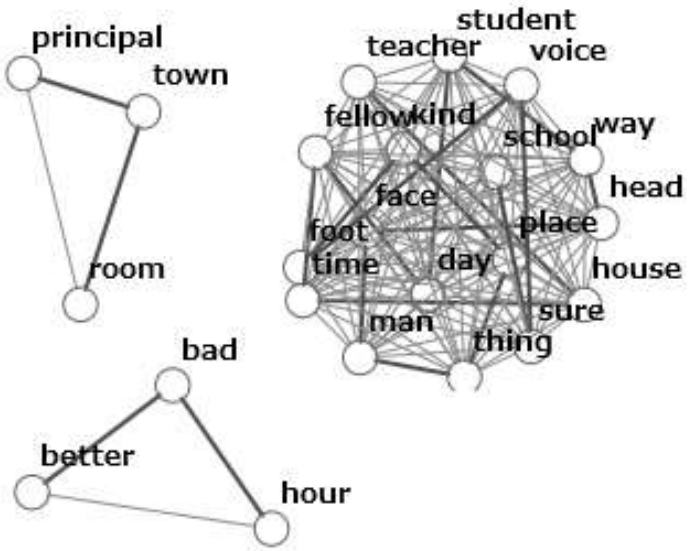

Fig. 4. Co-occurrence network with 35 node, 97 edge

C. Network Co-occurrence Analysis

In this step we show network co-occurrence analysis with two different settings. We conduct two network co-occurrence with 11 document, 695 paragraphs and 3064 sentences. The documents extracted from news articles in DUC2001 [34] with several limitations. First Ineffective and stop words are removed. Second, it has been allowed to words with certain part of-speech tags to be candidate keywords. These tags (Noun, Proper Name and Adjective) captured from Stanford POS tagger [12]. Third, term frequency rate for each word must be greater than 30 (TF^1>30) and document frequency of each word must be greater than 10 (DF^2>10). After removing stop words and tagging, the remaining words are weighted by one of the individually term weighing methods. Then we run page rank as a random walk algorithm to propagate weighted terms and importance of relations (edges) across the co-occurrence Network. Larger circles show higher weight in contrasting to smaller circle.

Fig.4 illustrates first co-occurrence network with 35 node, 97 edge and windows size is whole document. As the neighborhood window becomes larger, graph will be full and more complete. In this situation many irrelevant relation between words with high dispersion are considered. Fig.5 shows second network co-occurrence with 35 node, 61 edge and windows size 2. As you see, nodes with the same color are strong relevant to each other.

D. Drawing Minimum Spanning Tree

After analyzing two different network in earlier section. In this section, we captured minimum spanning tree from each co-occurrence network with two different settings. We conduct two networks co-occurrence with 11 documents, 695 paragraphs, 3064 sentences and 54867 tokens. After removing stop words and tagging, the remaining words are weighted by one of the term weighing methods. Then we run page rank as a random walk algorithm to propagate weighted terms and importance of relations (edges) larger circles shows higher weight in contrasting to smaller circle.

[1] Term frequency

[2] Document frequency

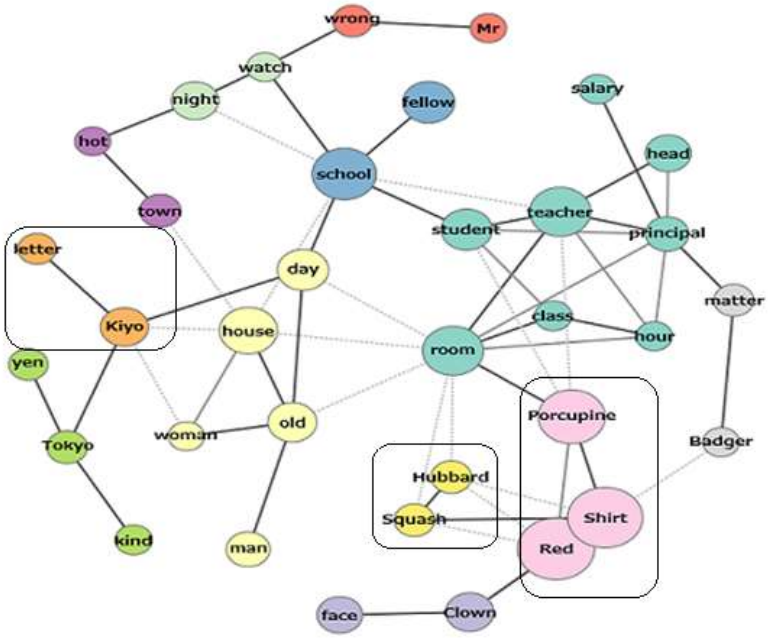

Fig 5. Co-occurrence network with 35 node, 61 edge

Fig.6 illustrates first network co-occurrence with 35 node, 34 edge and windows size 2. In this network semantically similar keywords have same color. After conducting minimum spanning tree, candidate phrases are highlighted and many of weak links removed.

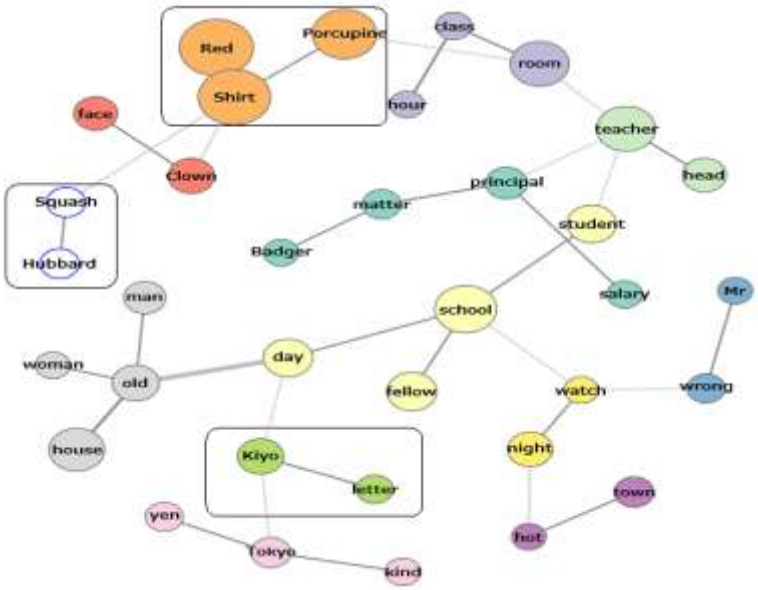

Fig 6. Co-occurrence network minimum spanning tree with 35 node, 34 edge

E. Comparing with Baseline Methods

We outperform three baselines (TF-IDF, Page Rank, SingleRank) on both datasets. The results show that the proposed method is more efficient than other methods in two datasets.This proves the effect of co-occurrence network optimization using Parallel minimum spanning tree so that reduces candidate number and increases accuracy of extracted keyphrases.

TABLE.1 COMPARING WITH BASELINE METHODS

Keyword Ranking Methods	**Comparison With Baseline Methods**		
	Pre.	***Rec.***	***F. measure***
TF-IDF	0.333	0.173	0.227
Page Rank	0.330	0.171	0.225
SingleRank	0.286	0.352	0.2

PMST[3]+PageRank+MI[4]	0.286	0.352	0.321

Comparing our result with different baseline methods when the number of extracted keyphrases from each document is 5 using dataset DUC[34].

The comparing result of our method with the other baseline methods under precision, recall and F-measure has been shown. (See Table.1 and Table.2).

TABLE.2 COMPARING WITH BASELINE METHODS

Keyphrase Ranking Method	Keyword Ranking Methods	comparison with Baseline Methods		
		Pre.	*Rec.*	*F. measure*
$\sum_{t_j \in K} \log R(w_j)$	TF-IDF	0.376	0.196	0.271
	Page Rank	0.330	0.171	0.283
	SingleRank	0.253	0.321	0.277
	PMST+ PageRank + MI[5]	0.359	0.386	0.376

Comparing our result with different baseline methods when the number of extracted keyphrases from each document is 10 using dataset[35].

Table.3 Keyphrase Extraction Results

Keyword Ranking Method	Term Weighting Method	Candidate Selection	Keyphrase Ranking Method	
			$\sum_{t_j \in K} R(t_j)$	$\sum_{t_j \in K} \log R$
			F. measure	*F. measure*
PageRank	tf*idf	PMST+ Stops words cut-off	0.250	0.248
PageRank	EN[6]		0.261	0.262
PageRank	MI[7]		0.265	0.266

Comparing results of different settings of proposed methods when the number of extracted keyphrases from each document is 5 using dataset [35].

We, also compare different version of our proposed methods with different settings (See Table.3 and Table.4) Our method exploits the advantages of both minimum spanning tree and PageRank, by eliminating irrelevant weak phrase from space of candidate keyphrases.

Table.4 Keyphrase Extraction Results

Keyword Ranking Method	Term Weighting Method	Candidate Selection	Keyphrase Ranking Method	
			$\sum_{t_j \in K} R(t_j)$	$\sum_{t_j \in K} \log R(w$
			F. measure	*F. measure*
PageRank	tf*idf	PMST+ Stops words cut-off	0.292	0.291
PageRank	EN[8]		0.315	0.315
PageRank	MI[9]		0.341	0.343

Comparing results of different settings of proposed methods when the of extracted keyphrases from each document is 10 using database DUC[34].

Moreover, we show the relation between number of extracted keyphrase per document and f-measure (f-score) for all documents in the corpus(See Fig.7, Fig.8). These curves are evaluated on different numbers of extracted keyphrases. Table.1 and Table.2, show that the proposed method has better overall performance by increasing the number of extracted keyphrases. Finally, we compare our proposed keyphrase ranking method with other baselines on [35] dataset. Table-5 shows that the proposed method is more efficient in identifying good keyphrase with the length of 3-4. This proves the effect of considering second order co-occurrence point-wise mutual information.

TABLE.5 DIFFERENT RANKING METHODS

Phrase Ranking	Keyword Ranking Method	Term Weighting Method	Candidate Selection	F. Measure
$\sum_{t_j \in K} R(t_j)$	PageRank	Variance Approach	PMST+ Stops words cut-off	**0.341**
$\sum_{t_j \in K} \log R($	PageRank	Mutual information		**0.343**
SOC-PMI[10]	PageRank	Mutual information		**0.374**

Comparing results of different Ranking methods when the number of phrases is 10 using database [35].

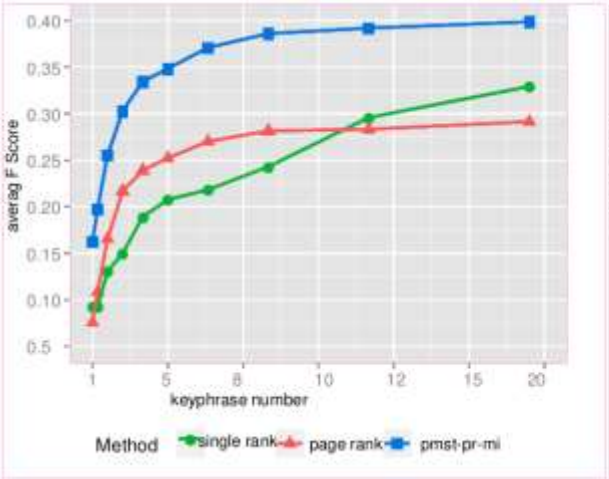

Fig 7. Comparing F-measure of proposed method with Baseline

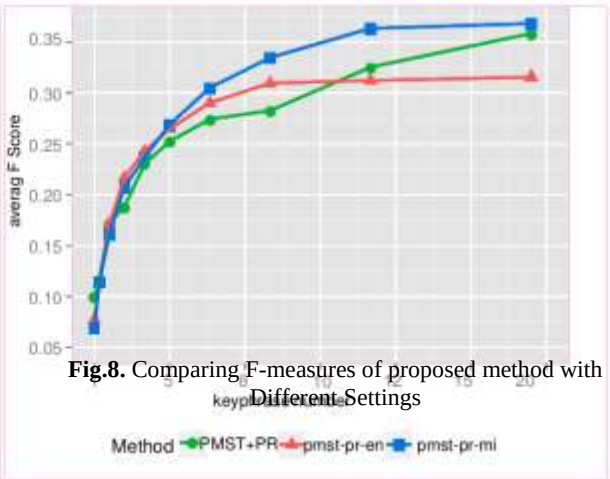

Fig.8. Comparing F-measures of proposed method with Different Settings

[6] Entropy

[7] Mutual Information

[10] Second order point-wise mutual information

VI. Conclusion

In this paper, we proposed an unsupervised method for automatic keyphrase extraction. Most of unsupervised methods deal with two challenges. First, they do not use background knowledge to understand deeper structure of document. This point of view can be facilitated by involving term weighting methods and constitution of parallel minimum spanning tree for eliminating original graph to get proper relevant edge and use page rank to propagate importance of words across the graph.

Second challenge is computing importance of candidate phrase with a length of 3-4 words. We proposed a new ranking method for candidate phrase with 3-4 long. We use candidate phrase ranking method to find prior keyphrases. We enjoy point-wise mutual information by considering second order co-occurrence of words. This ranking methods assigned a weight to each candidate phrase. It semantically improves vector space representation of term document matrix. It will help system to distinguish candidate keyphrases and non-keyphrases.

Refrences

[1] [1] P. D. Turney, "Learning Algorithms for Keyphrase Extraction," *Inf. Retr. Boston.*, vol. 2, no. 4, pp. 303–336, 2000.

[2] [2] T. D. Nguyen and M.-Y. Kan, "Keyphrase extraction in scientific publications," in *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers*, Springer, 2007, pp. 317–326.

[3] [3] A. SIDDHARTHAN, "Christopher D. Manning and Hinrich Schutze. Foundations of Statistical Natural Language Processing. {MIT} Press, 2000. {ISBN} 0-262-13360-1. 620 pp. {\textdollar}64.95/{\textsterling}44.95 (cloth).," *Nat. Lang. Eng.*, vol. 8, no. 01, 2002.

[4] [4] Z. Liu, W. Huang, Y. Zheng, and M. Sun, "Automatic Keyphrase Extraction via Topic Decomposition," *Comput. Linguist.*, no. October, pp. 366–376, 2010.

[5] [5] K. S. Hasan and V. Ng, "Automatic Keyphrase Extraction: A Survey of the State of the Art," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 1262–1273.

[6] [6] Z. Liu, W. Huang, Y. Zheng, and M. Sun, "Automatic keyphrase extraction via topic decomposition," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 2010, pp. 366–376.

[7] [7] F. Yu, H.-W. Xuan, and D.-Q. Zheng, "Key-Phrase Extraction Based on a Combination of CRF Model with Document Structure," in *Eighth International Conference on Computational Intelligence and Security*, 2012, pp. 406–410.

[8] [8] Y. Qi, M. Song, S.-C. Yoon, and L. deVersterre, "Combining Supervised Learning Techniques to Key-Phrase Extraction for Biomedical Full-Text," *Int. J. Intell. Inf. Technol.*, vol. 7, no. 1, pp. 33–44, 2011.

[9] [9] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," in *Proceedings of EMNLP*, 2004, vol. 4, no. 4, p. 275.

[10] [10] Y. F. Huang and C. S. Ciou, "Constructing personal knowledge base: Automatic key-phrase extraction from multiple-domain web pages," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7104 LNAI, Springer, 2012, pp. 65–76.

[11] [11] H. White, C. Willis, and J. Greensberg, "HIVEing: The Effect of a Semantic Web technology on Inter-Indexer Consistancy," *J. Doc.*, vol. 70, no. 3, pp. 1–43, 2014.

[12] [12] L. zi Liao and H. yan Huang, "Microblog Keyphrase Extraction Based on Similarity Features," in *Proceedings of the 2013 International Conference on Advanced Computer Science and Electronics Information*, 2013.

[13] [13] X. Wan and J. Xiao, "Single Document Keyphrase Extraction Using Neighborhood Knowledge.," in *AAAI*, 2008, vol. 8, pp. 855–860.

[14] [14] Z. Liu, P. Li, Y. Zheng, and M. Sun, "Clustering to Find Exemplar Terms for Keyphrase Extraction," in *Language*, 2009, vol. 1, pp. 257–266.

[15] [15] F. Liu, D. Pennell, and Y. Liu, "Unsupervised approaches for automatic keyword extraction using meeting transcripts," in *Naacl-Hlt*, 2009, no. June, pp. 620–628.

[16] [16] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," *Proc. EMNLP*, vol. 4, no. 4, pp. 404–411, 2004.

[17] [17] T. Tomokiyo and M. Hurst, "A language model approach to keyphrase extraction," in *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*, 2003, pp. 33–40.

[18] [18] N. Kumar and K. Srinathan, "Automatic keyphrase extraction from scientific documents using N-gram filtration technique," in *... of the eighth ACM symposium on Document ...*, 2008, vol. Sao Paulo, p. 199.

[19] [19] M. Litvak, M. Last, H. Aizenman, I. Gobits, and A. Kandel, "DegExt — A Language-Independent Graph-Based Keyphrase Extractor," in *Advances in Intelligent Web Mastering {\textendash} 3*, Springer Science \mathplus Business Media, 2011, pp. 121–130.

[20] [20] R. Wang, W. Liu, and C. McDonald, "How preprocessing affects unsupervised keyphrase extraction," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8403 LNCS, no. PART 1, Springer, 2014, pp. 163–176.

[21] [21] J. Repplinger, "G.G. Chowdhury. Introduction to Modern Information Retrieval. 3rd ed. London: Facet, 2010. 508p. alk. paper, $90 (ISBN 9781555707156). LC2010-013746.," *Coll. Res. Libr.*, vol. 72, no. 2, pp. 194–195, 2011.

[22] [22] W. Wang, D. B. Do, and X. Lin, "Term Graph Model for Text Classification," in *Advanced Data Mining and Applications*, Springer Science \mathplus Business Media, 2005, pp. 19–30.

[23] [23] A. Bellaachia and M. Al-Dhelaan, "NE-Rank: A novel graph-based keyphrase extraction in Twitter," in *Proceedings - 2012 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2012*, 2012, pp. 372–379.

[24] [24] M. Sasaki, "Latent Semantic Word Sense Disambiguation Using Global Co-Occurrence Information," in *Computer Science & Information Technology (CS & IT)*, 2014, pp. 463–468.

[25] [25] Y. Ye, X. Li, B. Wu, and Y. Li, "A comparative study of feature weighting methods for document co-clustering," *Int. J. Inf. Technol. Commun. Converg.*, vol. 1, no. 2, p. 206, 2011.

[26] [26] E. Tsui, W. M. Wang, L. Cai, C. F. Cheung, and W. B. Lee, "Knowledge-based extraction of intellectual capital-related information from unstructured data," *Expert Syst. Appl.*, vol. 41, no. 4 PART 1, pp. 1315–1325, 2014.

[27] [27] Y. Bin Kang, P. Delir Haghighi, and F. Burstein, "CFinder: An intelligent key concept finder from text for ontology development," *Expert Syst. Appl.*, vol. 41, no. 9, pp. 4494–4504, 2014.

[28] [28] K. E. Lochbaum and L. A. Streeter, "Comparing and combining the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval," *Inf. Process. Manag.*, vol. 25, no. 6, pp. 665–676, Jan. 1989.

[29] [29] I. Dhillon, J. Kogan, and C. Nicholas, "Feature Selection and Document Clustering," in *Survey of Text Mining*, Springer, 2003, pp. 73–100.

[30] [30] S. Rothe and H. Schütze, "CoSimRank: A Flexible and Efficient Graph-Theoretic Similarity Measure," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 1392–1402.

[31] [31] S. Chung and A. Condon, "Parallel implementation of Bouvka{\textquotesingle}s minimum spanning tree algorithm," in *Proceedings of International Conference on Parallel Processing*, 1996.

[32] [32] H. Ryang and U. Yun, "Unsupervised Keyphrase Extraction Based Ranking Algorithm for Opinion Articles," in *Multimedia and Ubiquitous Engineering*, Springer Science \mathplus Business Media, 2013, pp. 113–119.

[33] [33] A. Islam and D. Inkpen, “Second Order Co-occurrence PMI for Determining the Semantic Similarity of Words,” pp. 1033–1038.

[34] [34] H. T. Dang, “{DUC} 2005,” in *Proceedings of the Workshop on Task-Focused Summarization and Question Answering - {SumQA} {\textquotesingle}06*, 2006.

[35] [35] A. Hulth, “Improved Automatic Keyword Extraction Given More Linguistic Knowledge,” in *EMNLP’03: Proceedings of the 2003 conference on Empirical Methods in Natural Language Processing*, 2003, pp. 216–223.

Hosein Shahsavar received his B.S. degree in Software Engineering from Kharazmi University of Technology in 2012. He also received his M.Sc. degree in Computer Engineering from Amirkabir University of Technology in 2011.at present he is Ph.D. student in Malek Ashtar University of Technology from 2013. His research interests are text mining, natural language processing, and statistical linguistic.

Mojtaba Hosseini received his B.Sc. degree in Electronics Engineering from Malek Ashtar University of Technology in 1991. He also received his M.Sc. and Ph.D. degrees in Computer Engineering from Amirkabir University of Technology in 1995 and 2011 respectively. His research interests are wireless sensor networks, image processing, and evolutionary computing.

Jamshid Shanbe Zadeh is an Associate Professor at Kharazmi University. He received his M.Sc. degree in Electronics Engineering from Tehran University of Technology in 1986. He also received his Ph.D. degrees in electrical and computer engineering from Wollongong University of Technology in 1996. His research interests are Image Compression, OCR, Document Analysis and Image Retrieval.