A Graph Based Keyword Extraction Model using Collective Node Weight

Article *in* Expert Systems with Applications · December 2017

DOI: 10.1016/j.eswa.2017.12.025

CITATIONS
11

READS
1,385

3 authors, including:

Saroj Kr. Biswas
National Institute of Technology, Silchar
40 PUBLICATIONS **97** CITATIONS
SEE PROFILE

Monali Bordoloi
National Institute of Technology, Silchar
8 PUBLICATIONS **16** CITATIONS
SEE PROFILE

Expert Systems With Applications 97 (2018) 51–59

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A graph based keyword extraction model using collective node weight

Saroj Kr. Biswas*, Monali Bordoloi, Jacob Shreya

Computer Science and Engineering Department, National Institute of Technology Silchar, 788010 Assam, India

ARTICLE INFO

Article history:
Received 22 August 2017
Revised 12 December 2017
Accepted 13 December 2017
Available online 13 December 2017

Keywords:
Sentiment analysis
Keyword extraction
Graph based model
Centrality measure
Text mining

ABSTRACT

In the recent times, a huge amount of text is being generated for social purposes on twitter social networking site. Summarizing and analysing of twitter content is an important task as it benefits many applications such as information retrieval, automatic indexing, automatic classification, automatic clustering, automatic filtering etc. One of the most important tasks in analyzing tweets is automatic keyword extraction. There are some graph based approaches for keyword extraction which determine keywords only based on centrality measure. However, the importance of a keyword in twitter depends on various parameters such as frequency, centrality, position and strength of neighbors of the keyword. Therefore, this paper proposes a novel unsupervised graph based keyword extraction method called Keyword Extraction using Collective Node Weight (KECNW) which determines the importance of a keyword by collectively taking various influencing parameters. The KECNW is based on Node Edge rank centrality with node weight depending on various parameters. The model is validated with five datasets: Uri Attack, American Election, Harry Potter, IPL and Donald Trump. The result of KECMW is compared with three existing models. It is observed from the experimental results that the proposed method is far better than the others. The performances are shown in terms of precision, recall and F-measure.

1. Introduction

Keywords are defined as a series of one or more words which provide a compact representation of a document's content (Berry & Kogan, 2010; Boudin, 2013; Grineva, Grinev, & Lizorkin, 2009; Lahiri, Choudhury, & Caragea, 2014). Keywords are widely used to define queries within information retrieval (IR) systems as they are easy to define, revise, remember, and share. Other applications using keywords include automatic indexing, automatic summarization, automatic classification, automatic clustering, automatic topic detection and tracking, and automatic filtering (Palshikar, 2007). The task of mining these keywords from a document is called as keyword extraction. The manual assignment of keywords is a very time consuming and tedious task so it is important to have a proficient automated keyword extraction approach.

Micro-blogs have been recently attracting people to express their opinion and socialize with others. Micro blogging is a combination of blogging and instant messaging that allows users to create short messages to be posted and shared with an audience online. Social platforms like twitter have become extremely popular forms of this new type of blogging, especially on the mobile web – making it much more convenient to communicate with people compared to the days when desktop web browsing and interaction was the norm. Users share thoughts, links and pictures on Twitter, journalists comment on live events, and companies promote products and engage with customers. The list of different ways to use twitter could be really long, and with 500 millions of tweets per day, there is a lot of data to analyze and explore. One of the most important tasks in analyzing twitter data is keyword extraction. If keywords of a text are extracted properly, subject of the text can be studied and analyzed comprehensively and good decision can be made on the text.

Texts are commonly represented using the well-known Vector Space Model (VSM) (Salton, Yang, & Yu, 1975), however it results in sparse matrices to be dealt with computationally and while target application involves twitter contents, compared with traditional text collections, this problem becomes even worse. Due to the short texts (140 characters), diversity in twitter contents, informality, grammatical errors, buzzwords, slangs, and the speed with which real-time content is generated, an effective technique is required (Ediger et al., 2010) to extract useful keywords. Graph based technique to extract keywords is appropriate in such situation and has gained popularity in the recent times.

Bellaachia and Al-Dhelaan (2012) proposed a graph based method to extract keywords from twitter data, which uses node weight with TextRank and results in a node-edge weighting approach called NE-Rank (Node and Edge Rank). Term Frequency–Inverse Document Frequency (TF–IDF) is used as the node weight. But, keywords in twitter data do not only depend on TF–IDF.

* Corresponding author.
E-mail address: bissarojkum@yahoo.com (S.Kr. Biswas).

https://doi.org/10.1016/j.eswa.2017.12.025
0957-4174/

Abilhoa and Castro (2014) proposed a graph based technique to extract keywords from twitter data, which uses closeness and eccentricity centralities to determine node weight and, degree centrality as the tie breaker. Closeness and eccentricity centralities do not work well for disconnected graphs. However in most of the cases, the graph made from tweets becomes a disconnected graph due to the diversity of the tweet contents. Therefore, an effective graph based keyword extraction method is required which can overcome most of the drawbacks of graph based model including the ones cited above. This paper proposes such a graph based keyword extraction method called Keyword Extraction using Collective Node Weight (KECNW) which depends on many parameters of a node like frequency, centrality, position and strength of neighbors.

The remaining part of the research article is organized as follows. Section 2 presents literature survey which describes the related previous works. Section 3 discusses the proposed model in great detail. An illustrative example is presented in Section 4 to understand the proposed model clearly. Results with discussion are presented in Sections 5 and 6 draws some conclusions about the research work.

2. Literature survey

The keyword extraction techniques can be divided into four categories namely, linguistic approach, machine learning approach, statistical approach and other approaches (Zahang et al., 2008). Linguistic approach uses the linguistic properties of the words, sentences and documents and the most commonly examined linguistic properties are lexical, syntactic, semantic and discourse analysis (Cohen-Kerner, 2003; Hulth, 2003; Nguyen & Kan, 2007). Machine learning approach considers supervised or unsupervised learning for keyword extraction. Supervised machine learning approach induces a model which is trained on a set of known keywords and then is used to find the keywords for unknown documents (Medelyan & Witten, 2006; Witten, Paynter, Frank, Gutwin, & Nevill-Manning, 1999; Zhang, Xu, Tang, & Li, 2006). Statistical approach comprises simple methods which do not require the training data and are language and domain independent. The statistics of the words from document can be used to identify keywords such as n-gram statistics, word frequency, TF–IDF, word co-occurrences, PAT Tree etc. (Chen & Lin, 2010). Other approaches for keyword extraction in general combine all approaches mentioned above.

Graph based approach is a statistical approach. Recently some graph based methods for keyword extraction have been proposed. Litvak, Last, Aizenman, Gobits, and Kandel (2011) proposed an unsupervised, graph-based and cross-lingual key phrase extractor, known as DegExt which uses simple graph-based syntactic representation of text and web documents to enhance the traditional vector-space model by taking into account some structural document features. Bellaachia and Al-Dhelaan (2012) proposed a novel unsupervised graph based keyword ranking method, called NE-Rank which considers word weights in addition to edge weights when calculating the ranking. Bougouin, Boudin, and Daille (2013) proposed an unsupervised method that aims to extract key phrases from the most important topics of a document, called as TopicRank. Topics are defined as clusters of similar key phrase candidates. Beliga, Mestrovic, and Martincic-Ipsic (2015) proposed a node selectivity model for the task of keyword extraction. The node selectivity is defined as the average strength of the node. Abilhoa and Castro (2014) proposed a keyword extraction method from tweet collections that represents texts as graphs and applies centrality measures- degree, closeness and eccentricity, for finding the relevant vertices (keywords). Lahiri, Choudhury and Caragea (2014) experimented different centrality measures such as degree, strength, neighborhood size – order 1, coreness, pagerank etc. on word and noun phrase collocation networks for keyword extraction and analyzed their performance on four benchmark datasets. Kwon, Choi, and Lee (2015) proposed a model for term weighting and representative keyword extraction. Wang, Feng, and Li (2016) introduced Average Term Frequency (ATF) and Document Frequency (DF) to calculate the node weight. Martinez-Romo, Araujo, and Fernandez (2016) introduced an unsupervised algorithm for extracting key phrases from a collection of texts based on a semantic relationship graph, called SemGraph. Tixier, Malliaros, and Vazirgiannis (2016) introduced a new unsupervised keyword extraction technique that capitalizes on graph degeneracy. This technique applies the *K*-truss algorithm to the task of keyword extraction for the first time. Khan, Yukun, and Kim (2016) proposed a graph based re-ranking approach, called Term Ranker which extracts single-word and multi-word terms by using a statistical approach, identifies groups of semantically similar terms, estimates term similarity based on term embeddings and uses graph refinement and node centrality ranking. Xue, Qin, and Liu (2016) proposed a model to identify topics from folksonomy by using topic models. Ravinuthala and Reddy (2016) proposed a directed graph representation technique in which weighted edges are drawn between the words based on the theme of the document. Nagarajan, Nair, Aruna, and Puviarasan (2016) presented a keyword extraction algorithm where documents are represented as graphs, words of the documents are represented as nodes and the relation between the words of the documents is represented as edges. Then degree and closeness centrality measures are used for keyword extraction. Song, Go, Park, Park, and Kim (2017) proposed a method which considers three major factors that make it different from other keyword extraction methods. The three major factors are temporal history of the preceding utterances, topic relevance and the participants. The utterances spoken by the current speaker should be considered as more important than those spoken by other participants.

3. Proposed KECNW model

The KECNW model considers frequency, centrality, position and strength of neighbors of a node to calculate importance of the node. The implementation of the model is segregated in 4 phases: preprocessing, textual graph representation, node weight assignment and keyword extraction. The details of all the phases are given below.

3.1. Phase 1: pre-processing

Twitter is a micro-blog where people generally write in a conversational style. Tweets are known to be very noisy for any text mining task as they contain a number of symbols that do not have any useful information and make further processing ineffective. Therefore this model includes effective pre-processing phase which removes meaningless symbols from tweets and hence, effective keywords can be extracted. The steps for pre-processing are as follows:

i. *Remove username and retweet symbol:* Tweets often contain usernames beginning with the symbol '@'. Sometimes a tweet is also re-tweeted, which means a tweet by any user is shared again by other users and it contains the symbol RT. These usernames and retweet symbol do not contribute any significance to keyword extraction and act as noise. So, usernames and retweet symbols are removed.
ii. *Remove URLs:* Any URL links appearing in the tweets are removed as the model focuses only on the textual part of the tweet and URLs act as unnecessary noise while keywords are extracted.

iii. *Remove hash tags:* The Hash tag i.e. # before a word such as *#KarnatakaWithCongress* is removed to get '*KarnatakaWithCongress*'.
iv. *Tokenization:* Each term in a tweet is treated as a token. Tokens are the basic constituents of a tweet/text. Let T be the set of tweets which is represented as $T = \{T_1, T_2, T_3,\ldots,T_i|\ i$ is the number of tweets$\}$. Then each tweet in T is pre-processed and its terms are treated as tokens. Let t be the set of tokens represented as $t = \{t_1, t_2, t_3,\ldots,t_k\}$. t includes tokens from all the tweets of T where the number of tokens in the set T is k.
v. *Stop word removal:* A standard list of stop words is created and these stop words are then removed from the set.
vi. *Removal of unimportant tokens:* There are many tokens in the set which are comparatively less important and cannot be keywords. A mechanism is established to identify and remove these tokens so that they do not compete in the keyword extraction phase, making it more efficient. The tokens which occur less than the Average Occurrence Frequency (AOF) are removed. AOF is determined by Eq. (1):

$$AOF = \frac{\sum \text{Frequency of each node}}{\text{Number of nodes}} \tag{1}$$

For a given node i, if frequency $(i) <$ AOF, delete node i
vii. Additional white spaces are removed after removing the stop words and unimportant nodes.

3.2. Phase 2: textual graph representation

Let $G = (V, E)$ is a graph where V is the set of vertices and E is the set of edges. Then the textual graph is represented as follows:

i *Vertex assignment:* One vertex is created for one token. The set of vertices (V) is created from the set of tokens in the vertex assignment.
ii *Edging:* The edges are established by pairs of tokens read in the same sequence in which they appear in the original tweets/texts. So an edge $E_{i,j}$ is generated for each token "*i*" and its immediate successor "*j*". An adjacency matrix for the textual graph is created which represents weights of edges. The weight of each edge is based on frequency of the nodes and their co-occurrence frequency. The weight of an edge between vertices/nodes/terms t_i and t_j is determined by Eq. (2) (Sonawane & Kulkarni, 2014).

$$W_c(i, j) = \frac{\text{freq}(i, j)}{\text{freq}(i) + \text{freq}(j) - \text{freq}(i, j)} \tag{2}$$

Where, freq (i, j) is the number of times node i and j co-occur, and freq (i) and freq (j) are the occurrence frequencies of nodes i and j, respectively.

3.3. Phase 3: node weight assignment

The weight of a node plays important role in keyword extraction using graph based model. If the node weight is evaluated properly, effective and representative keywords of a text/tweet can be determined. The KECNW model considers different important parameters to calculate node weight as the importance of a node depends on many factors. The parameters used in the calculation of node weight are as follows:

i. *Distance from central node:* The closer a node is to the most central node the more probability it has to be an important keyword. Therefore, importance of a node is calculated as the inverse of its distance from the central node. The central node is found using degree of the nodes. The node with highest degree is considered as the most central. In case of a tie in the degree of nodes, frequency of the nodes is used to break the tie. If central node is c then the importance for a node i is determined by Eq. (3).

$$D_{C(i)} = \frac{1}{d(c, i)} \tag{3}$$

Where $d(c, i)$ is distance of node i from the central node c. This value is normalized to be within the range of 0 and 1. For the central node c, the $D_{C(c)}$value is set to 1.
ii. *Selectivity Centrality:* Centrality measure is an indication of the importance of a node within a graph. Selectivity is defined as the average weight distribution on the links of a single node. Selectivity centrality works well for even disconnected graphs. The strength of a vertex (v), $s\ (v)$, is a sum of the weights of all edges incident with the vertex v. The selectivity of vertex v is calculated as a fraction of the vertex strength and vertex degree $d(v)$ and is given by Eq. (4).

$$SC(v) = \frac{s(v)}{d(v)} \tag{4}$$

$$s(v) = \sum_u W_{vu} \tag{5}$$

iii. *Importance of neighboring nodes:* A node is considered more important if its neighbors are also important. The importance of the neighboring nodes can be calculated as the average strength of all the neighbors, which is calculated by Eq. (6).

$$Neigh_{Imp}(i) = \frac{\sum_j \text{Strength}(j)}{N} \tag{6}$$

Where j is any neighbor of i, N is the number of neighbors of i and $Neigh_{Imp}(i)$ is the importance of the neighboring nodes of i.
iv. *Position of a node:* Hotho, Nürnberger, and Paab (2005) suggested that the position of a term is an important criterion while extracting keywords. Since twitter datasets contain short texts, the probability of the first or last word to become keyword is higher. So added weight is given to them, which is calculated as follows.

a. If token i is the first word then, $F[i] = n_f/\text{freq}(i)$, where, n_f is the number of times i is the first word, freq(i) is the frequency of the term i and $F[i]$ is the weight of i.
Else $F[i] = 0$
b. Similarly, If token i is the last word then, $L[i] = n_l/\text{freq}(i)$, where, n_l is the number of times i is the last word and $F[i]$ is the weight of i.
Else $L[i] = 0$

v. *Term frequency:* It is defined as the number of times a given term occurs in a document and is an essential parameter because important keywords tend to occur more frequently in a document.

Then, the weight of any node i is calculated by Eq. (7) and is normalized to get value between 0 and 1 by Eq. (8). The normalized value of i is the final weight of i.

$$Node\_weight(i) = D_{C(i)} + SC(i) + Neigh_{Imp}(i) + F(i) + L(i) + TF(i) \tag{7}$$

Where, $SC(i)$ and $TF(i)$ are the selectivity centrality and term frequency of i, respectively.

$$Final_{weight(i)} = \frac{Node\_weight(i) - \min\_weight}{\max\_weight - \min\_weight} \tag{8}$$

Where $Node\_weight(i)$ is the weight of node i, min_*weight* is the minimum weight among all the nodes and max_*weight* is the maximum weight among all the nodes.

3.4. Phase 4: keyword extraction

Keyword extraction is the process of identifying keywords from a tweet/document that can appropriately represent the subject of the tweet/document. The proposed model uses NE rank and degree to extract keywords.

i. *Calculate the NE rank:* NE rank method uses the weight of nodes with TextRank method which results in a node-edge weighting approach called NE-Rank (Node and Edge Rank) (Bellaachia & Al-Dhelaan, 2012): The rank/relevance of node v_i using NE-Rank is calculated by Eq. (9) given below:

$$R(v_i) = (1-d)\cdot W(v_i) + d\cdot W(v_i)\cdot \sum_{j:v_j\to v_i} \frac{w_{ji}}{\sum_{k:v_j} w_{jk}} R(v_j) \quad (9)$$

Here, d is the damping factor which denotes the probability of jumping from a node to the next node and is usually set to 0.85. Similarly, $(1-d)$ denotes the probability of jumping to a new node. $W(v_i)$ is the weight of the current node v_i. w_{ji} is the weight of the edge from the previous vertex v_j to the current vertex v_i and $\sum_{k:v_j} w_{jk}$ is the summation of all edge weights in the previous node v_j. $R(v_j)$ denotes the rank/relevance of node v_j.

ii. *Calculate the degree centrality:* Degree centrality of a node is the number of edges incident on the node.

iii. *Sort the keywords:* Do the following for all the terms/nodes.

Let i and j be two terms/nodes such that j occurs immediately after i in the list/text.
Then:
a. If NE(i) = NE(j)
   If degree(i) < degree(j)
   Swap (i, j)
b. Else if NE(i) < NE(j)
   Swap (i, j)
c. Otherwise
   No action

Finally, n best ranked terms/nodes are selected as keywords.

4. An illustrative example

Five tweets from the IPL dataset are taken to illustrate the whole model in detail. Tweets are given as follows:

(i) 'Very excited for todays IPL contest RPS vs KKR, @msdhoni vs @GautamGambhir fight! #IPL'
(ii) '#poll who score 50+ score today #smithy #dhoni #stokes #Rahane #KKRvRPS #rpsvskkr #cricketlovers #ipl #IPL2017'
(iii) 'RPS should be happy team today because KKR have decided to rest NCN. He has been in prime form. #KKRvRPS #IPL @RPSupergiants @KKRiders'
(iv) 'KKR seek to extend unbeaten run against Pune https://t.co/NdEuZIdxL5 via @cricbuzz @RPSupergiants @KKRiders #IPL'
(v) '#RPSvKKR Predict What will be the outcome? #ipl #KKRvRPS #ipl #Smithy #Gambhir 21

After removing noise such as RT symbol, @ symbol and URLs, the tweets are as follows:

(i) 'Very excited for todays IPL contest RPS vs KKR, vs fight! IPL'
(ii) 'poll who score 50+ score today smithy dhoni stokes Rahane KKRvRPS rpsvskkr cricketlovers ipl IPL2017'
(iii) 'RPS should be happy team today because KKR have decided to rest NCN. He has been in prime form. KKRvRPS IPL
(iv) 'KKR seek to extend unbeaten run against Pune via IPL'
(v) 'RPSvKKR Predict What will be the outcome? ipl KKRvRPS ipl Smithy Gambhir 21

Table 1
Weights of nodes of the example.

Keywords	Node weight
ipl	1
rps	0.185009684653528
vs	0.129336748044948
kkr	0.367501703383498
score	0.207936870261025
today	0
smithy	0.178192465012761
kkrvrps	0.203549229564497

Table 2
NE centrality and degree centrality for the example.

Nodes	NE rank centrality	Degree centrality
ipl	0.160774440966870	0.714
rps	0.0287550866827946	0.428
vs	0.0212645819311876	0.428
kkr	0.0584493747455156	0.571
score	0.0342120630760550	0.142
today	0	0.5714
smithy	0.0277812860734499	0.428
kkrvrps	0.0351474648352414	0.428

Table 3
Top keywords.

Keywords	Rank
ipl	1
kkr	2
kkrvrps	3
score	4
rps	5
smithy	6
vs	7
today	8

After tokenization and removing stop words the set (t) of tokens is represented as t = {excited, todays, ipl, contest, rps, vs, kkr, vs, fight, ipl, poll, score, 50, score, today, smithy, dhoni, stokes, rahane, kkrvrps, rpsvskkr, cricketlovers, ipl, ipl2017, rps, happy, team, today, kkr, decided, rest, ncn, prime, form, kkrvrps, ipl, kkr, seek, extend, unbeaten, run, pune, ipl, rpsvkkr, predict, outcome, ipl, kkrvrps, ipl, smithy, gambhir, 21}.

Average Occurrence Frequency (AOF) is calculated using frequency of all the nodes which is 1.3659. The nodes with frequency less than 1.3659 are removed. So the set (t) is represented as: t = {ipl, rps, vs, kkr, score, today, smithy, kkrvrps}.

The textual graph is then represented by Fig. 1 and weights of the nodes calculated by Eq. (7) are shown in normalized form in Table 1.

The NE rank centrality and degree centrality of each node are given in Table 2.

Top keywords using NE rank centrality with degree as tie breaker are shown in Table 3.

5. Results and discussion

The experiments are conducted with five datasets: Donald Trump, Harry Potter, IPL, Uri Attack and American Election. All datasets are collected from twitter. Donald Trump, Harry Potter and IPL datasets contain 2000 tweets each while Uri Attack and American Elections contain 500 tweets each.

Precision (Pr), Recall (Re), and F-measure performance measures are used as evaluation metrics for keyword extraction and are given in Eqs. (10)–(12).

$$Pr = \frac{|\{Relevant\}\cap\{Retrieved\}|}{|\{Retrieved\}|} \quad (10)$$

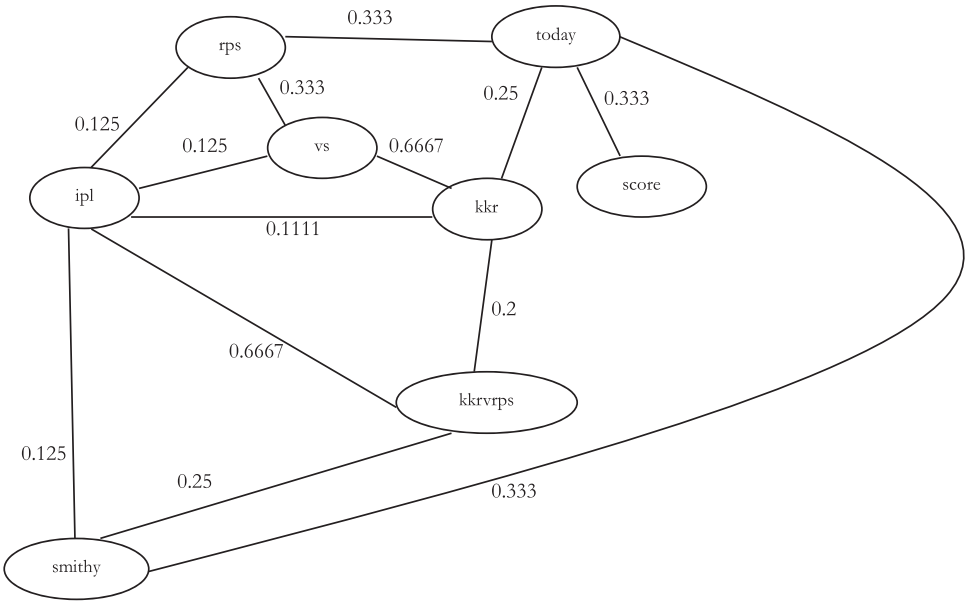

Fig. 1. Textual graph of the illustrative example.

$$Re = \frac{|\{Inter\_Relevant\} \cap \{Retrieved\}|}{|\{Inter\_Relevant\}|} \tag{11}$$

$$F - measure = 2 \times \frac{Pr \times Re}{(\mathrm{Pr} + Re)} \tag{12}$$

To compute Pr, *Relevant* denotes the number of retrieved keywords which appear in at least one of the human lists/sets. To compute Re, *Inter_Relevant* denotes the number of retrieved keywords which appear in the intersection set of the three human lists.

The node weight assignment is a sum of five different measures over the nodes, therefore the experiment takes different combinations to show how different measures are sensitive or important to find overall accuracy of the propose model. The description of the five different combinations considered along with the proposed KECNW is given below:

(i) (KECNW – Distance from central node) is the KECNW model without distance from central node and its accuracy shows the importance of distance from central node to find keywords.
(ii) (KECNW – Selectivity centrality) is the proposed model without selectivity centrality and its accuracy shows the importance of selectivity centrality to find keywords.
(iii) (KECNW – Importance of neighboring nodes) is the KECNW model without importance of neighboring nodes and its accuracy shows the importance of importance of neighboring nodes to find keywords.
(iv) (KECNW – Position of a node) is the proposed model without position of a node and its accuracy shows the importance of position of a node to find keywords.
(v) (KECNW – Term frequency) is the proposed model without term frequency and its accuracy shows the importance of term frequency to find keywords.

Performances of five different combinations mentioned above along with KECNW are shown in Table 4 in percentage when the number of retrieved keywords is 10.

Following observations are studied from the experimental results shown in Table 4:

(i) KECNW and (KECNW – Term frequency) produce 100% accuracy and all other combinations produce 94.74% accuracy in Uri Attack dataset.
(ii) KECNW and all the combinations produce the same results in American Election dataset.
(iii) KECNW and (KECNW – Position of a node) produce 94.74% accuracy and all other combinations produce 88.89% accuracy in IPL dataset.
(iv) KECNW and (KECNW – Term frequency) produce 88.89% accuracy and all other combinations produce 82.35% accuracy in Harry Potter dataset.
(v) KECNW produces 88.89% and all the combinations produce 75% accuracy in Donald Trump dataset.

It can be concluded from the results that except (term frequency) and (position of a node) all other measures have equal importance in the model. (term frequency) and (position of a node) are less important than other measures. (term frequency) is less important than (position of a node). Finally, it can be concluded that all the measure are significant to find the node weight, however weighted measures cannot be considered as the proposed KECNW is an unsupervised model.

In addition an incremental approach is used to make five different combinations to show how different parameters are sensitive or important to find the overall accuracy of the proposed model. The different parameters are added one by one in KECNW model for experimentation. Performances of five different combinations obtained by adding the parameters one by one are shown in Table 5 in percentage when the number of retrieved keywords is 10. Following observations are studied from the experimental results shown in Table 5:

(i) Using only Distance from central node, $D_{C(i)}$ for the node weight assignment, KECNW produces the same accuracy for four datasets namely Uri Attack, American Election, Harry Potter and IPL and slightly lower accuracy for the Donald Trump dataset as compared to that when all parameters are used.
(ii) For Harry Potter dataset, KECNW produces slightly higher accuracy when summation of $D_{C(i)}$ and SC(i) is used for assigning node weights than KECNW model with all the parameters. For two datasets, KECNW produces same results

Table 4
Performance of the different combinations of measures in KECNW model.

Dataset	Uri Attack			American Election			IPL			Harry Potter			Donald Trump		
Model	Pr	*Re*	F	Pr	*Re*	F	Pr	*Re*	F	Pr	*Re*	F	Pr	*Re*	F
KECNW	100	100	100	90	100	94.74	90	100	94.74	80	100	88.89	80	100	88.89
KECNW- Distance from central node	90	100	94.74	90	100	94.74	80	100	88.89	70	100	82.35	60	100	75
KECNW- Selectivity Centrality	90	100	94.74	90	100	94.74	80	100	88.89	70	100	82.35	60	100	75
KECNW- Importance of neighboring nodes	90	100	94.74	90	100	94.74	80	100	88.89	70	100	82.35	60	100	75
KECNW- Position of a node	90	100	94.74	90	100	94.74	90	100	94.74	70	100	82.35	60	100	75
KECNW- Term frequency	100	100	100	90	100	94.74	80	100	88.89	80	100	88.89	60	100	75

Table 5
Performance of the incremental combinations of parameters in KECNW model.

Dataset	Uri Attack			American Election			IPL			Harry Potter			Donald Trump		
Model	Pr	*Re*	F	Pr	*Re*	F	Pr	*Re*	F	Pr	*Re*	F	Pr	*Re*	F
KECNW using $D_{C(i)}$	100	100	100	90	100	94.74	90	100	94.74	80	100	88.89	70	100	82.35
KECNW using $D_{C(i)} + SC(i)$	90	100	94.74	90	100	94.74	90	100	94.74	90	100	94.74	70	100	82.35
KECNW using $D_{C(i)} + SC(i) + Neigh_{Imp}(i)$	100	100	100	90	100	94.74	90	100	94.74	80	100	88.89	70	100	82.35
KECNW using $D_{C(i)} + SC(i) + Neigh_{Imp}(i) + F(i) + L(i)$	100	100	100	90	100	94.74	90	100	94.74	80	100	88.89	70	100	82.35
KECNW (using $D_{C(i)} + SC(i) + Neigh_{Imp}(i) + F(i) + L(i) + TF(i)$)	100	100	100	90	100	94.74	90	100	94.74	80	100	88.89	80	100	88.89

when all the parameters are used and when summation of $D_{C(i)}$ and SC(i) is considered. However, a slight decrease in accuracy is encountered for Uri Attack and Donald Trump datasets when only $D_{C(i)}$ and SC(i) is used for node weight assignment.

(iii) For four datasets, the results using $D_{C(i)}$, SC(i) and $Neigh_{Imp}(i)$, and using all the parameters in KECNW model are same. However KECNW with all the parameters performs better than KECNW with $D_{C(i)}$, SC(i) and $Neigh_{Imp}(i)$, in Donald Trump dataset.

(iv) For four datasets, the results using $D_{C(i)}$, SC(i), $Neigh_{Imp}(i)$ and; F(i) and L(i) in KECNW model and using all five parameters in KECNW model are same. However KECNW model with all the parameters performs better than KECNW model with $D_{C(i)}$, SC(i), $Neigh_{Imp}(i)$ and; F(i) and L(i), in Donald Trump dataset.

From the observations, it can be concluded that distance from the central node ($D_{C(i)}$) is the most important parameter in the node weight assignment. Secondly, selectivity centrality (SC(i)) is found to be the second most significant parameter while calculating the weights of the nodes. Term frequency seems to be important than neighboring nodes ($Neigh_{Imp}(i)$), and position of nodes (F(i), L(i)). However neighboring nodes ($Neigh_{Imp}(i)$) and position of nodes (F(i), L(i)) seem to be equally significant in the weight assignment.

From the observations of Tables 4 and 5, it can be concluded that though all the parameters are significant for the node weight assignment, the distance from the central node ($D_{C(i)}$) and selectivity centrality (SC(i)) are the most important parameters in the proposed KECNW model.

Finally the proposed KECNW is compared with two graph based keyword extraction techniques: TKG (Abilhoa & Castro , 2014) and KEGBA (Nagarajan, Nair, Aruna, & Puviarasan, 2016) and, a non-graph based keyword extraction technique: TF–IDF. Variants of TKG (Abilhoa & Castro, 2014) are also considered for comparison.

Three edge-weighting possibilities are considered for existing model (Abilhoa & Castro, 2014): same weight assignment (W_1), weight as co-occurrence frequency (W_f) and weight as inverse co-occurrence frequency ($W_{1/f}$). Degree centrality (C_d), Closeness centrality (C_c) and Eccentricity centrality (C_e) are used to measure the relevance of nodes/terms. However, the proposed model uses Eq. (2) to find edge weight and Eq. (7) to find node weight.

Keywords are defined manually because there is no correct set of keywords for a given dataset, not even humans may agree on the keywords they extract from a given dataset. Therefore, three persons are invited to suggest an unspecified number of keywords from the datasets. The intersection and union of the sets identified by three people for each dataset are determined. Table 6 contains keywords extracted by three persons for five datasets and the intersection sets are represented by bold face.

Tables 7–11 show performances of TKG, KEGBA, TF–IDF and proposed KECNW for Uri Attack, American Election, IPL, Harry Potter and Donald Trump datasets, respectively. The following observations are made from the experimental results when the number of retrieved keywords is 10. The proposed model provides highest Precision, Recall and F-measure in Uri Attack dataset. However, the Precision is equal to the Precision produced by KEGBA with degree centrality. The KECNW model provides highest Precision, Recall and F-measure in American Election dataset. However the Recall is equal to the Recall produced by TF–IDF model. The model also provides highest Precision, Recall and F-measure in IPL dataset. The proposed model provides highest Precision, Recall and F-measure in Harry Potter dataset however the Precision is equal to the Precision produced by TKG with TextRank centrality, and the Recall is equal to the Recall produced by TF–IDF model. For Donald Trump dataset the proposed KECNW model provides better Precision and F-measure than TKG and KEGBA model; however, the Recall is equal to the Recall produced by TKG with TextRank and closeness centralities, and KEGBA with degree centrality. Further KECNW and TF–IDF produce the same results for this dataset.

The significant improvement is observed in all the datasets because the proposed KECNW uses effective pre-processing step which also remove unimportant tokens/terms by using AOF. Weight of the edges is determined based on frequency of the nodes and their co-occurrence frequency, thus KECNW takes care of frequency and relation between the nodes in an effective and balanced manner to find weight of the edges. Weight of the nodes is determined by combining five measures, which overcomes disadvantages of each other's. Therefore weight of the nodes determined by KECNW becomes more effective. Finally rank of keywords is determined by NE centrality which works well with disconnected graph as term frequency and degree of node are used as parameters to compute this centrality. Whereas TKG uses simple pre-processing step by doing only tokenization and stop word removal and finds weight of the edges by same weight edges/ co-

Table 6
Keywords extracted by three persons for five datasets.

	Dataset	Extracted keywords
Reader 1	Uri Attack	**Martyred, uri, attack, terror,** pm**, army,** soldiers, condemns, **surgical,strike**,terrorist, **india,Pakistan,,**surgicalstrike, pak, Kashmir, **uriattack**
	American Election	America, campaign, supporters, Russia, riot, land, media **American,** Donald**, presedential,**People, **election, trump, Hilary,**2016
	IPL	**'ipl'**, **'rpsvkkr'**, **'dhoni'**, 'rps', **'ipl2017'**, 'kkrvrps', pune, score, smith, msdhoni, hit, Kolkata, Tripathi, team, playing
	Harry Potter	**'harry', 'potter'**, 'time', **'oscar'**, 'wins', marcus, love, books, movies, universal, watch, read, phoenix
	Donald Trump	**'donald', trump,** 'russia', 'administration', **'president', 'obama'**, donaltrump, people, hate, Hillary, senator, soldiers, Oscars, republicans
Reader 2	Uri Attack	**uri, attack, army, surgical, strike, uriattack, india, terror, Martyred,**people,**Pakistan**,fawad,ban,artist,jawans,pm
	American Election	**American,** 2016,**presedential, election, trump, Hilary,** media, Clinton, Obama, shock, refugee, flag, win, result, supporters
	IPL	**'ipl', 'rpsvkkr'**, 'kkr', tripathi, **'dhoni'**, 'rps', **'ipl2017'**, 'match', rahane, uthappa, kohli, cricket, win, scores, commentary
	Harry Potter	**'harry', 'potter'**, 'finally', **'oscar'**, 'harrypotter', series, time, books, read, winning, half-blood, prince, Radcliff
	Donald Trump	**'donald', trump, 'president'**, 'clinton', **'obama'**, 'house', travel, ban, hate, muslims, cost, trumprussia, tax, returns
Reader 3	Uri Attack	**uri, attack, army, surgical, strike, uriattack, india, terror**, surgicalstrike, **Martyred,** Kejri, Aap, Ban, Modi, terrorist, issue, Rahul, blame, **Pakistan**
	American Election	**American, presedential, election, trump, Hilary,** people, muslims, Donald, president, US, democracy, like,lost, canada, women
	IPL	**'ipl', 'rpsvkkr'**, 'kkr', **'dhoni'**, **'ipl2017'**, 'gambhir', 'kkrvrps', top, hit, rcb, virat, Stokes, msdhoni, raina, Bravo
	Harry Potter	**'harry', 'potter', 'oscar'**, 'back', 'wins', first, franchise,wands, ron, weasley, Daniel,scar, Gryffindor, favourite
	Donald Trump	**'donald', trump, 'president', 'obama'**, 'clinton', tax, Russian, criminals, republicans, bush, resistance, interview, cost, refugees

Table 7
Performances in Uri Attack dataset.

Model	Edge weighting Mechanism	Pr in %	Re in %	F in %
TKG with closeness centrality (Abilhoa & Castro, 2014)	W_1	60	60	60
	W_f	50	30	37.5
	$W_{1/f}$	70	50	58.33
TKG with Eccentricity Centrality (Abilhoa & Castro, 2014)	W_1	90	80	84.7
	W_f	30	10	15
	$W_{1/f}$	50	30	37.5
TKG with Eigen centrality (Abilhoa & Castro, 2014)	W_1	30	30	30
	W_f	30	30	30
	$W_{1/f}$	60	30	40
TKG with TextRank centrality (Abilhoa & Castro, 2014)	W_1	90	80	84.7
	W_f	90	80	84.7
	$W_{1/f}$	90	70	78.75
TKG with Selectivity Centrality (Abilhoa & Castro, 2014)	W_c	70	50	58.33
KEGBA with degree (Nagarajan et al., 2016)	W_f	100	80	88.89
KEGBA with closeness (Nagarajan et al., 2016)	W_f	70	50	58.33
TF–IDF (Zhang et al., 2006)	...	90	60	72
Proposed KECNW	W_c	100	100	100

Table 8
Performances in American Election dataset.

Model	Edge weighting Mechanism	Prin %	Re in %	F in %
TKG with closeness centrality (Abilhoa & Castro, 2014)	W_1	50	80	61.53
	W_f	40	40	40
	$W_{1/f}$	60	80	68.57
TKG with Eccentricity Centrality (Abilhoa & Castro, 2014)	W_1	10	0	0
	W_f	10	10	10
	$W_{1/f}$	60	80	68.57
TKG with Eigen centrality (Abilhoa & Castro, 2014)	W_1	50	60	54.54
	W_f	60	80	68.57
	$W_{1/f}$	60	60	60
TKG with TextRank centrality (Abilhoa & Castro, 2014)	W_1	60	80	68.57
	W_f	60	80	68.57
	$W_{1/f}$	60	80	68.57
TKG with Selectivity Centrality (Abilhoa & Castro, 2014)	W_c	20	20	20
KEGBA with degree (Nagarajan et al., 2016)	W_f	60	80	68.57
KEGBA with closeness (Nagarajan et al., 2016)	W_f	40	40	40
TF–IDF (Zhang et al., 2006)	....	80	100	88.89
KECNW	W_c	90	100	94.74

occurrence frequency/ inverse co-occurrence frequency. TKG uses closeness and eccentricity centralities to determine node weight and, degree centrality as the tie breaker; however closeness and eccentricity centralities do not work well for disconnected graphs. Even TKG is implemented with different centrality measures but does not perform better than KECNW because TKG uses simple pre-processing and edge weighting techniques, and does not combine different measures to find weight of the nodes. KEGBA represents graph without doing pre-processing and uses simple co-occurrence relations between words to find weight of the edges. Finally degree and closeness centralities are separately used to find weight of the nodes. Thus noises are not removed and edge weights are not properly captured. Centrality measures and others are not combined to overcome each other's disadvantages to find

Table 9
Performances in IPL dataset.

Model	Edge weighting mechanism	Prin %	Re in %	*F* in %
TKG with closeness centrality (Abilhoa & Castro, 2014)	W_1	60	75	66.67
	W_f	70	50	58.33
	$W_{1/f}$	70	50	58.33
TKG with Eccentricity Centrality (Abilhoa & Castro, 2014)	W_1	10	25	30.77
	W_f	40	25	30.77
	$W_{1/f}$	40	25	30.77
TKG with Eigen centrality (& Castro, 2014)	W_1	60	75	66.67
	W_f	60	50	54.54
	$W_{1/f}$	60	50	54.54
TKG with TextRank centrality (Abilhoa & Castro, 2014)	W_1	80	75	77.42
	W_f	80	75	77.42
	$W_{1/f}$	80	75	77.42
TKG with Selectivity Centrality (Abilhoa & Castro, 2014)	W_c	70	75	67.59
KEGBA with degree (Nagarajan et al., 2016)	W_f	80	75	77.42
KEGBA with closeness (Nagarajan et al., 2016)	W_f	70	50	58.33
TF–IDF (Zhang et al., 2006)	....	70	75	72.41
KECNW	W_c	90	100	94.74

Table 10
Performances in Harry Potter dataset.

Model	Edge weighting mechanism	Pr in %	Re in %	*F* in %
TKG with Closeness centrality (Abilhoa & Castro, 2014)	W_1	70	66.67	68.29
	W_f	70	66.67	68.29
	$W_{1/f}$	70	66.67	68.29
TKG with Eccentricity Centrality (Abilhoa & Castro, 2014)	W_1	40	33.33	36.36
	W_f	50	33.33	39.99
	$W_{1/f}$	50	33.33	39.99
TKG with Eigen centrality (Abilhoa & Castro, 2014)	W_1	20	66.67	30.77
	W_f	20	66.67	30.77
	$W_{1/f}$	20	66.67	30.77
TKG with TextRank centrality (Abilhoa & Castro, 2014)	W_1	80	66.67	72.73
	W_f	80	66.67	72.73
	$W_{1/f}$	80	66.67	72.73
TKG with Selectivity Centrality (Abilhoa & Castro, 2014)	W_c	60	66.67	63.16
KEGBA with degree (Nagarajan et al., 2016)	W_f	70	66.67	68.29
KEGBA with closeness (Nagarajan et al., 2016)	W_f	20	66.67	30.77
TF–IDF (Zhang et al., 2006)	....	60	100	75
KECNW	W_c	80	100	88.89

Table 11
Performances in Donald Trump dataset.

Model	Edge weighting mechanism	Pr in %	Re in %	*F* in %
TKG with Closeness centrality (Abilhoa & Castro, 2014)	W_1	60	100	75
	W_f	60	100	75
	$W_{1/f}$	60	100	75
TKG with Eccentricity Centrality (Abilhoa & Castro, 2014)	W_1	20	25	22.22
	W_f	20	25	22.22
	$W_{1/f}$	20	25	22.22
TKG with Eigen centrality (Abilhoa & Castro, 2014)	W_1	20	25	22.22
	W_f	20	25	22.22
	$W_{1/f}$	20	25	22.22
TKG with TextRank centrality (Abilhoa & Castro, 2014)	W_1	60	100	75
	W_f	60	100	75
	$W_{1/f}$	60	100	75
TKG with Selectivity Centrality (Abilhoa & Castro, 2014)	W_c	30	75	42.85
KEGBA with Degree (Nagarajan et al., 2016)	W_f	60	100	75
KEGBA with Closeness (Nagarajan et al., 2016)	W_f	50	50	50
TF–IDF (Zhang et al., 2006)	....	80	100	88.89
KECNW	W_c	80	100	88.89

weight of the nodes. TF–IDF solely depends on frequency of words and disregards the relationship between words. Sometimes an important word appears less number of times which is not taken care by TF–IDF. TF–IDF does not perform better keyword extraction in twitter data due to the diversity and informality in twitter contents.

Asymptotic time complexity of the proposed KECNW is $O(N^2)$ where N is the total number of terms/words in a dataset because the time complexity of the pre-processing phase is $O(N)$, the textual graph representation phase is $O(N)$, the node weight assignment is $O(N)$ and the keyword extraction phase is $O(N^2)$. The time complexity of TKG is $O(N)$ and the time complexity of KEGBA is also $O(N)$. The time complexity of the TF-IDF is $O(N^2)$ where N is the number of tweets. Though the time complexity of the proposed model is more, however the accuracy is far better than TKG, KEGBA and TF–IDF.

6. Conclusions

Keyword extraction is one of the most important tasks in analyzing twitter data. This paper proposes a keyword extraction model called KECNW which consists of four phases: pre-processing, textual graph representation, node weight assignment and keyword extraction. Pre-processing involves removing unwanted noise, tokenization and stop word removal. Textual graph representation involves vertex assignment and establishment of edges between vertices. Node weight assignment phase determines node weight based on its frequency, centrality, position and strength of neighbors. Finally, keywords are extracted using NE rank centrality with degree as tie breaker. The proposed, KECNW model is compared with variants of two existing graph based models and one non graph based model. It is observed from the experimental results that the performance of KECNW model is far better than others. The proposed KECNW model can be used in sentiment analysis for keyword extraction.

In future, new edge weighing and node weighing methods can be proposed and they can be used in NE rank centrality to find rank of keywords. Other centrality measures can be used individually or in combination in future to get better results. Semantic methods for keyword extraction can be explored with proposed model to get better results.

Funding

This study is not funded by any research grant.

Conflict of Interest

There is no conflict of interest.

Human and animal rights

This article does not contain any studies with human or animal participant.

References

Abilhoa, W. D., & Castro, L. N. de. (2014). A keyword extraction method from twitter messages represented as graphs. *Applied Mathematics and Computation, 240*, 308–325.

Beliga, S., Mestrovic, A., & Martincic-Ipsic, S. (2015). An overview of graph-based keyword extraction methods and approaches. *J. Inf. Org. Soc., 39*(1), 1–20 2015.

Bellaachia, A., & Al-Dhelaan, M. (2012). NE-Rank: a novel graph-based key phrase extraction in twitter. In *Proceedings of the international joint conferences on web intelligence and intelligent agent technology: 1* (pp. 372–379). IEEE, WIC, ACM.

Berry, M. W., & Kogan, T. (2010). *Text mining: applications and theory*. UK: Wiley.

Boudin, F. (2013). A comparison of centrality measures for graph-based keyphrase extraction. In Proceedings of the international joint conferences on Natural Language Processing *(IJCNLP)* (pp. 834–838).

Bougouin, A., Boudin, F., & Daille, B. (2013). TopicRank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the international joint conference on natural language processing (IJCNLP)* (pp. 543–551).

Chen, P., & Lin, S. (2010). Automatic keyword prediction using Google similarity distance. *Expert System with Application, 37*(3), 1928–1938.

Cohen-Kerner, H. (2003). Automatic extraction of keyword from abstracts, automatic extraction of keyword from abstracts. *Lecture Notes in Computer Science, 2773*, 843–849.

Ediger, D., Jiang, K., Riedy, J., Bader, D. A., Corley, C., Farber, R., et al. (2010). Massive social network analysis: Mining twitter for social good. In *Proceedings of the thirty-ninth international conference on parallel processing* (pp. 583–593). IEEE.

Grineva, M., Grinev, M., & Lizorkin, D. (2009). Extracting key terms from noisy and multi-theme documents. In *Proceedings of the eighteenth international conference on world wide web* (pp. 661–670).

Hotho, A., Nürnberger, A., & Paab, G. (2005). A brief survey of text mining. *LDV Forum GLDV Journal for Computational Linguistics and Language Technology, 20*(1), 19–62.

Hulth, A. (2003). Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 216–223).

Khan, T. Md., Yukun, M., & Kim, J. (2016). Term ranker: A graph based re-ranking approach. In *Proceedings of the FLAIRS conference (AAAI)* (pp. 310–315).

Kwon, K., Choi, C. H., & Lee, J. (2015). A graph based representative keywords extraction model from news articles. In *Proceedings of the International conference on big data applications and services* (pp. 30–36). ACM.

Lahiri, S., Choudhury, S. R., & Caragea, C. (2014). Keyword and keyphrase extraction using centrality measures on collocation networks. arXiv:1401.6571 [cs.CL].

Litvak, M., Last, M., Aizenman, H., Gobits, H., & Kandel, A. (2011). DegExt – A language-independent graph-based keyphrase extractor. In E. Mugellini, P. S. Szczepaniak, M. C. Pettenati, & M. Sokhn (Eds.). In *Advances in intelligent web mastering – 3. Advances in intelligent and soft computing: 86* (pp. 121–130). Springer.

Martinez-Romo, J., Araujo, L., & Fernandez, A. D. (2016). SemGraph: Extracting keyphrases following a novel semantic graph-based approach. *Journal of the Association for Information Science and Technology, 67*(1), 71–82.

Medelyan, O., & Witten, I. H. (2006). Thesaurus based automatic keyphrase indexing. In *Proceedings of the sixth ACM/IEEE-CS joint conference on digital libraries* (pp. 296–297).

Nagarajan, R., Nair, S. A. H., Aruna, P., & Puviarasan, N. (2016). Keyword extraction using graph based approach. *International Journal of Advanced Research in Computer Science and Software Engineering, 6*(10), 25–29.

Nguyen, T. D., & Kan, M. Y. (2007). Keyphrase extraction in scientific publications. In *Proceedings of the tenth international conference on Asian digital libraries: Looking back 10 years and forging new frontiers* (pp. 317–326).

Palshikar, G. K. (2007). Keyword extraction from a single document using centrality measures. *Pattern Recognition and Machine Intelligence (LNCS), 4851*, 503–510.

Ravinuthala, M. K. V. V., & Reddy, S. (2016). Thematic text graph: A text representation technique for keyword weighting in extractive summarization system. *International Journal of Information Engineering and Electronic Business, MECS, 8*(4), 18–25.

Salton, G., Yang, C. S., & Yu, C. T. (1975). A theory of term importance in automatic text analysis. *Journal of the American Society and Information Science and Technology (eCommons), 26*(1), 33–44.

Sonawane, S. S., & Kulkarni, P. A. (2014). Graph based representation and analysis of text document: A survey of techniques. *International Journal of Computer Applications, 96*(19), 1–8.

Song, H. J., Go, J., Park, S. B., Park, S. Y., & Kim, K. Y. (2017). A just-in-time keyword extraction from meeting transcripts using temporal and participant information. *Journal of Intelligent Information System, 48*(1), 117–140.

Tixier, A. J. P., Malliaros, F. D., & Vazirgiannis, M. (2016). A graph degeneracy-based approach to keyword extraction. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)* (pp. 1860–1870).

Wang, Z., Feng, Y., & Li, F. (2016). The improvements of text rank for domain-specific key phrase extraction.. *International Journal of Simulation Systems, Science & Technology, 17*(20), 11.1–11.5.

Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., & Nevill-Manning, C. G. (1999). KEA: practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM conference on Digital libraries* (pp. 254–255).

Xue, H., Qin, B., & Liu, T. (2016). Topical key concept extraction from folksonomy through graph-based ranking. *Journal of Multimedia Tools and Application, 75*(15), 8875–8893.

Zahang, C., Wang, H., Liu, Y., Wu, D., Liao, Y., & Wang, B. (2008). Automatic keyword extraction from documents using conditional random fields. *Journal of CIS, 4*(3), 1169–1180.

Zhang, K., Xu, H., Tang, J., & Li, J. (2006). Keyword extraction using support vector machine. In *Proceedings of the seventh international conference on advances in web-age information management* (pp. 85–96).