Keyphrase extraction using unsupervised PageRank model with position and theme bias

Tuhin Kundu*
University of Illinois at Chicago
Chicago, Illinois
tkundu2@uic.edu

Yaozong Shen
University of Illinois at Chicago
Chicago, Illinois
yshen55@uic.edu

ABSTRACT

With growing amounts of online scholarly data, the importance of keyphrase extraction is undeniable, which is used to represent high level contextual and descriptive information about research articles. Automated keyphrase extraction systems are the need of the hour for knowledge discovery, so that research articles can be more accurately described using keyphrases. In this article, we present an unsupervised graph model that uses a scoring criterion to rank keyphrases based on the position and frequency of keyphrases and the theme of the research document in question. We used FastText word embeddings to get thematic keyphrases, theme vector and thematic weights, all which are used to represent the theme of the document and are used in the scoring criterion. We use the PageRank algorithm to generate our unsupervised graph model used to score the keyphrases separately using position and theme bias of the keyphrases. Evaluations have been done over benchmark datasets, achieving significant performance improvements.

KEYWORDS

keyphrase extraction, PageRank, unsupervised learning, position, theme

ACM Reference Format:
Tuhin Kundu and Yaozong Shen. 2019. Keyphrase extraction using unsupervised PageRank model with position and theme bias. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

There are millions of scientific documents over the web and it is both a boon and a bane. The promise of knowledge discovery over the vast number of research documents seems alluring due to its benefits, but finding the knowledge or information that is useful in such a deluge of information poses a challenge in itself.

Keyphrases are composed of single or multiple words that are provide a high-level contextual and descriptive information of a document. They are used for efficient processing of information useful for the purpose of recommending articles to readers, identifying potential reviewers, highlighting research trends and mapping citations to articles. In addition, keyphrases are also used in other natural language processing and information retrieval tasks such as clustering, recommendation, summarization and search. Such a wide range of utilities make keyphrase extraction an important task for building automated systems for knowledge discovery.

In this article, we attempt to incorporate position and frequency of keyphrases and the theme of a scholarly documents, on the basis of two previous models, PositionRank [5] and Key2Vec [13], and use an unsupervised biased PageRank graph model to rank the keyphrases, on multiple benchmark datasets and successfully showcase significant performance gains over PositionRank [5].

The rest of the article is organized as follows: Section 2 summarizes related work, Section 3 describes the proposed construction of our model, datasets and experimental results are described in Section 4 and Section 5 concludes the article.

2 RELATED WORK

Keyphrase extraction are broadly classified into two types: supervised and unsupervised [8, 9]. Supervised method of keyphrase extraction includes approaching the problem as a binary classification. Binary classification task for keyphrase extraction includes classfying the keyphrases as candidate and non–candidate keyphrases (positive and negative class) respectively [6, 9, 10]. Citation networks can also be used in addition to existing feature based methods such as TF-IDF, POS, relativePOS [2]. Unsupervised line of keyphrase extraction includes using TF-IDF, graph–based methods and clustering. The graph based techniques include the construction of a directed or undirected graph with keyphrases considered as nodes and edges as association patterns. Biased–PageRank models are a common graph construction algorithm used for ranking the nodes (representing the keyphrases) [5, 12, 15]. Graph methods are generally considered to state-of-the-art [9].

With the increasing use of deep learning techniques in natural language processing (NLP), words (which are components of keyphrases), are more oftern represented as vectors of real-numbers, popularly known as word embeddings. Word embeddings are word vectors which preserve semantic and syntactic information about the word they are associated with are utilised in a wide range of NLP tasks such as part-of-speech tagging, name-entity recognition, chunking, syntactic parsing, semantic role enabling, speech processing and many others [4]. Widely known and used word embeddings include Word2Vec [16], GloVe [18] and FastText [1]. Word embeddings have previously been used in keyphrase extraction and have shown promising results [20, 21].

In this work, we take two previous models, PositionRank [5] and Key2Vec [13], and attempt to use position and frequency of keyphrases and the theme of the scholarly document to generate

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

a position and theme biased PageRank graph to assign scores to the nodes which are the candidate keyphrases and rank them in descending order to get the set of keyphrases with the highest scores. We use PositionRank as it uses the position and frequency of keyphrases in a position biased PageRank that takes into account all the positions of the keyphrase occurrences in an unsupervised approach. Higher probability is assigned to keyphrases occurring at the beginning of the document, rather than using a uniform probability distribution over the words present in the document. We then use a theme-weighted PageRank graph [11] to represent candidate keyphrases extracted from a scientific article using domain specific phrase embeddings so that the thematic weight of each keyphrase represents the similarity or closeness of the keyphrase with the thematic representation or the main theme of the article, also generated using the same word embeddings. The scores of the positon-weighted PageRank graph and the theme-weighted PageRank graph are scaled and added up, to rank the keyphrases in descending order to give out the set of keyphrases closesly associated with the position of the keyphrases and the main theme of scholarly document. To our knowledge, this may the first attempt of an unsupervised model incorporating both positonal and thematic information in an unsupervised approach for keyphrase extraction.

3 PROBLEM STATEMENT

The goal of this project is to build a keyphrase extraction model that uses candidate keyphrases extracted from scholarly articles and rank them using a modified novel PageRank algorithm in an unsupervised graph model.

Keyphrase extraction enables faster processing by mapping multiword phrases to a document, that describe it the best. The task is important for building automated systems that are able to provide high level contextual and descriptive information about research articles which may be used for recommending articles to readers, identifying potential reviewers, highlighting research trends and mapping citations to articles. This project aims to generate candidate keyphrases from an embedding model and rank them using a modified PageRank algorithm while capturing information that would accurately represent or describe the paper. Some previous graph based models such as Key2vec and PositionRank, amongst other supervised and unsupervised keyphrase extraction models, have been used for background and ideation of the project.

4 PROPOSED METHODOLOGY

In this section, we describe the proposed methodology of our model, based on PositionRank [5] and Key2Vec [13]. Both models use an unsupervised graph-based model that take into account either the position, frequency and theme-based information for the candidate scoring step after the construction a biased-PageRank graph, where each node is a candidate or a keyphrase. Graph based algorithms like PageRank [17] measure the importance of a node of a graph by recursively computing global information of the entire graph.

Candidate keyphrases are phrases with word units in contiguous positions in a document. We consider noun phrases comprising of unigrams, bigrams and trigrams, matching upto the regular expression (adjective)*(noun)+. The phrases matching the regular expression ar extracted from the document using the StanfordCoreNLP toolkit [14].

4.1 Position-biased PageRank scores

Generating scores for candidate keyphrases using position and frequency bias comprise of three major steps:

- Construction of the graph at word level
- Designing the position and frequency biased PageRank in lines of PositionRank [5]
- Computing scores for candidate keyphrases

The steps are are explained in detail below:

4.1.1 Graph construction. Let the target document be d for extracting keyphrases. We use StanfordCoreNLP toolkit [14] for Part-of-speech tagging and using candidate words comprising of nouns and adjectives. Using the candidate words, we form the candidate keyphrases using the regular expression (adjective)*(noun)+, and the list of all them is denoted by L. We build a word graph where every unique candidate word in a node V in graph $G = (V, E)$ for every document d. Two nodes in graph G are connected by nodes v_i and v_j are connected by edge $(v_i, v_j) \epsilon E$ if the words in the nodes corresponding to the nodes co-occurred within a window of w size in document d and where weight of the edge is denoted by the co-occurrence count of the respective words in nodes v_i and v_j. The construction of the graph can done using a directed or undirected graph, which has insignificant effect on the performance of the keyphrase extraction model [15]. Therefore, build an undirected graph for our PageRank model.

4.1.2 Biasing the PageRank. Let M be the adjacency matrix for the graph with every element $m_{ij} \epsilon M$ denotes the weight between nodes v_i and v_j if there exists an edge, 0 otherwise.

With $v_i \epsilon V$, let S be the vector with the PageRank scores with the initial values of S set at $\frac{1}{|V|}$ and the score at step $t + 1$ can be computed as:

$$S(t+1) = \widetilde{M} \cdot S(t) \tag{1}$$

where $\widetilde{M}$ is the normalized form of matrix M with $m_{ij} \epsilon \widetilde{M}$ defined as:

$$\widetilde{m_{ij}} = \begin{cases} \frac{m_{ij}}{\sum_{j=1}^{|V|} m_{ij}} & if \sum_{j=1}^{|V|} m_{ij} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The computation of the PageRank model can be viewed as a Markov Chain process, so by recursively using Equation (1), we get the principle eigenvector which denotes the stationary probability distribution of every node in our graph G [19].

To ensure the Pagerank model doesn't get stuck in cyclic subgraphs, we use α as a damping factor, to get out of cyclic walks in graph G during the recursive computations. Hence S is denoted by:

$$S = \alpha \cdot \widetilde{M} \cdot S + (1 - \alpha) \cdot \widetilde{p} \tag{3}$$

where $\widetilde{p}$ is the a vector of length $\mid V \mid$ with all the elements $\frac{1}{|V|}$, which denotes that from node v_i, the random walk can jump

to any other node with equal probability, and S is the principal eigenvector.

The main idea is to assign higher weights to the words occurring more frequently early in the document d. For example, a word found in the 3^{rd} position will have higher weight than the word in 40^{th} position, by assigning the weight as the inverse of the position it was found at. So if the same word appeared at multiple positions, for example, the 2^{th}, 4^{th} and 8^{th} position, it'll be the summation of the inverse of the positions, $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} = 0.875$. Then the vector $\widetilde{p}$ is normalized as:

$$\widetilde{p} = \left[\frac{p_1}{p_1 + p_2 + ... + p_{|V|}}, \frac{p_2}{p_1 + p_2 + ... + p_{|V|}}, ..., \frac{p_{|V|}}{p_1 + p_2 + ... + p_{|V|}} \right] \tag{4}$$

Hence the PageRank score $S(v_i)$ of a vertex v_i is obtained by recursively computing the following:

$$S(v_i) = (1 - \alpha) \cdot \widetilde{p_i} + \alpha \cdot \sum_{v_j \epsilon Adj(v_i)} \frac{w_{ji}}{O(v_j)} S(v_j) \tag{5}$$

where $O(v_j) = \sum_{v_j \epsilon Adj(v_i)} w_{jk}$ and $\widetilde{p_i}$ is the weight of node v_i obtained from p.

The stopping criterion of the biased-Pagerank model was either 100 iteration reached or difference between two consecutive iterations as 0.001 [5].

4.1.3 Computing scores for candidate keyphrases. Using the candidate keyphrases formed out of candidate words, we assign scores of the candidate keyphrases by summing up the scores of the individual candidate words forming the candidate keyphrases. We store all the candidate keyphrase scores in vector K where $k_i \epsilon K$ and k_i is the score of the i^{th} candidate keyphrase where $i \epsilon L$.

After getting scores of all the candidate keyphrases, we recompute elements k_i of vector K by:

$$k_i = \frac{k_i - min(K)}{max(K) - min(K)} \tag{6}$$

where $min(K)$ and $max(K)$ denote the minimum and maximum values present in vector K.

4.2 Theme-biased scores

The methodology of incorporating theme based scoring primarily includes three steps: candidate selection, candidate scoring and candidate ranking. We represented candidate keyphrases using FastText [1] word embeddings to compute the candidate scores. The main aim of the embeddings is to capture semantic and syntactic similarities between single or multi text units. We chose FastText embeddings over other embeddings such as Word2Vec or GloVe as it captures both semantic and morphological similarities between words. The FastText embeddings used are 1 million word vectors with 16 billion tokens trained using Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset with each word vector of 300 dimensions. The motivation for this model is from Key2Vec [13].

4.2.1 Candidate keyphrase selection. The process of selecting candidate keyphrases and the number of unique candidate keyphrases for the theme-weighted PageRank graph is same as the candidate keyphrase selectection process for the position-biased PageRank scores. So for document d let the set of unique candidate keyphrases be denoted by $C_{d_i} = \{c_1, c_2, ..., c_n\}_{d_i}$, which are to be used for scoring and ranking in the upcoming steps.

4.2.2 Candidate scoring. In this step of candidate scoring, we assign a theme vector τ_{d_i} to every unique document d_i, with the idea for representing each document's theme or main idea using real numbered vector. The first step in doing so is to extract a theme excerpt of the document d_i, which includes the title and the first sentence of the document, and further extract unique set of name entities, noun phrases and unigram words called thematic phrases $T_{d_i} = \{t_1, t_2, ..., t_m\}_{d_i}$ from the thematic excerpt. We form a vector representation $\hat{t_j}$ of each thematic phrase extracted from the thematic excerpt of document d_i using the FastText embeddings and get the final theme vector by performing vector addition $\hat{\tau_{d_i}} = \sum_{j=1}^{m} \hat{t_j}$ of document. The FastText embeddings are also used to get the vector representation of the individual thematic keyphrases C_{d_i} denoted by $\hat{c_k} \forall k \epsilon \{1...n\}$.

4.2.3 Candidate ranking. For the scoring of the candidate keyphrases, we use a weighted PageRank algorithm where the nodes are the candidate keyphrases and the thematic weights are to be used for scoring them. A graph G_{d_i} for created for document d_i with C_{d_i} as the nodes of the graph and E_{d_i} as the edges connecting the two candidate keyphrase if the co-occur within a given window size of w. The scores (weights) are denoted by $sr(c_j^{d_i}, c_k^{d_i})$ in Equation 9 are calculated by using the cosine similarity between the summed embedding vectors of the two candidate keyphrases (Equation 7) and the Pointwise Mutual Information (PMI) [3] (Equation 8) between the two candidate keyphrases. The main intuition behind using this method for calculating the score is to capture information regarding how well two candidate keyphrases are related to each other and capturing the relation of co-occurence between two candidate keyphrases encapsulating the local relationship between them in context to document d_i.

$$semantic(c_j^{d_i}, c_k^{d_i}) = cosine(c_j^{d_i}, c_k^{d_i}) \tag{7}$$

$$cooccur(c_j^{d_i}, c_k^{d_i}) = \frac{PMI(c_j^{d_i}, c_k^{d_i})}{number\ of\ words\ in\ d_i} \tag{8}$$

We divide $PMI(c_j^{d_i}, c_k^{d_i})$ by the number of words in document d_i to ensure that $coocur(c_j^{d_i}, c_k^{d_i})$ does not have a huge value and does not get an abnormally large value due to the fact that PMI includes $number\ of\ words\ in\ d_i$ in its numerator.

$$sr(c_j^{d_i}, c_k^{d_i}) = semantic(c_j^{d_i}, c_k^{d_i}) \times coocur(c_j^{d_i}, c_k^{d_i}) \tag{9}$$

Given graph G, if $S(c_j^{d_i})$ be the set of edges incident on node $c_j^{d_i}$ and $w_{c_j}^{d_i}$ be the thematic weight of $c_j^{d_i}$ are calculated in the candidate scoring step, then the final score of node $R(c_j^{d_i})$ in the theme-weighted PageRank graph G is calculated as in Equation 10 with a damping factor of $d = 0.85$ and $out(c_j^{d_i})$ being the out degree of node $c_j^{d_i}$.

$$R(c_j^{d_i}) = (1-d)w_{c_j}^{d_i} + \sum_{c_k^{d_i} \epsilon S(c_j^{d_i})} \frac{sr(c_j^{d_i}, c_k^{d_i})}{|out(c_j^{d_i})|} \quad (10)$$

Again, in the same way as computing the scaled scores of the Position-based PageRank model as in Equation 6, we calculate scores of all the candidate keyphrases with elements l_i of vector L by:

$$l_i = \frac{l_i - min(L)}{max(L) - min(L)} \quad (11)$$

where $min(L)$ and $max(L)$ denote the minimum and maximum values present in vector L.

4.3 Score scaling and combined method

After getting the scores of the set of candidate keyphrases from the Positon-based PageRank model by Equation 6 and the Theme based model by Equation 11, we simple sum both the scores for each individual unique candidate keyphrase and store them in vector M. We again recompute the scores of all the candidate keyphrases with elements m_i of vector M by:

$$m_i = \frac{m_i - min(M)}{max(M) - min(M)} \quad (12)$$

where $min(M)$ and $max(M)$ denote the minimum and maximum values present in vector M and rank the candidate keyphrases in descending values of vector M. We then use Porter Stemmer to find the word stems of the candidate keyphrases and the keyphrases present in the ground truth of document d_i of the respective datasets and compare them to compute our results.

5 EXPERIMENTS AND RESULTS

In this section, we describe the results we got from all the experiments and compare our method with results from the PositonRank paper.

5.1 Datasets

We used the data which includes research papers from ten years of the KDD and WWW conferences given by [7] with gold-standard annotations of the research papers, which includes the author input keywords from the **Keywords** field, for evaluation of the keyphrases returned by our models. We used files containing more than 250 characters for our evaluations, and there were 297 and 666 such files for the KDD and WWW datasets respectively.

5.2 Evaluation metrics

As in previous works, we use precision, recall and F1 score for evaluation of our top-k keyphrases with $k\epsilon\{2, 4, 6, 8\}$ returned as output extracted keyphrases by our theme based model and combined model with position and theme scores.

5.3 Experiments

We performed standalone experiments for both Position and Theme biased PageRank models along with the combined model on KDD and WWW datasets for top-k keyphrases with $k\epsilon\{2, 4, 6, 8\}$ calculating precision, recall and F1 scores for all the methods respectively. We did not tabulate results our Position biased model, but tabulated results from the actual PositionRank paper for comparison. The results for our PositionRank implementation using scaled scores and Porter Stemmer for comparison is shown while running our combined model in the ipython notebook.

5.4 Evaluation on datasets

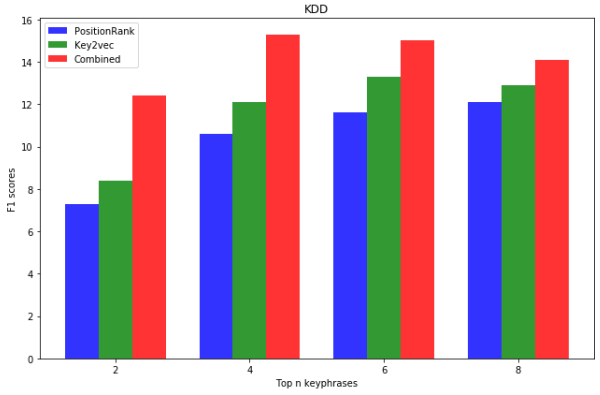
In table 1 and table 2, we tabulate the results run on the KDD and WWW datasets respectively. We ran the theme-weighted PageRank and our scoring-modified PositionRank model to obtain the results for our theme and position weighted scoring method for $N\epsilon\{2, 4, 6, 8\}$ with N being the number of top keyphrases returned by the respective models. We have also tabulated the results for the theme-weighted PageRank individually for comparison. We also tabulate the results from the original PositionRank paper [5] in the table to compare the results. P denotes precision, R denotes recall and $F1$ denotes the F1 score of the respective models in table 1 and table 2 for the two datasets.

6 DISCUSSION

As seen in tables 1 and 2, our method combining the position and theme weighted scores outperforms PositionRank [5] and also works significantly better than the theme weighted standalone model modelled on Key2Vec [13] too. Our model takes into consideration position, frequency and the theme into context for a particular given document hence outperforms PositionRank which takes into only the position and frequency. Key2Vec on the other hand does not take positional information into context, hence its crude implementation in our theme weighted model also performs worse than the combined model.

As the number of keyphrases returned as output by the models increases (top-k increases), there is an insigficant difference for the F1 scores of our combined model with $k\epsilon\{2, 4, 6, 8\}$. We also noticed, that there was a marginal improvement in performance using FastText embeddings over GloVe embeddings. Also, we check different sizes of theme excerpts (2 or 3 sentences instead of 1), and noticed that a very large theme excerpt led to degradation of performance metrics.

Figure 1: Comparison between the F1 scores for KDD dataset



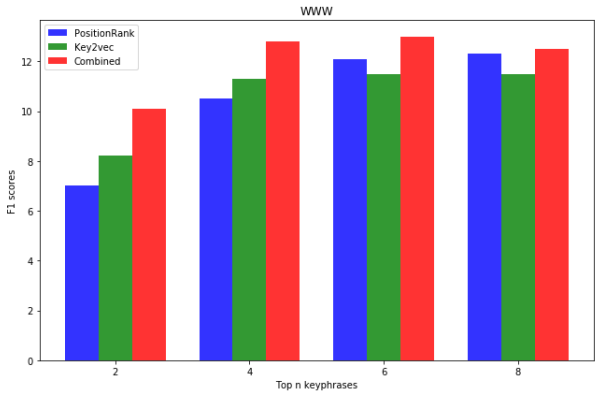
Methods	N=2 (%)			N=4 (%)			N=6 (%)			N=8 (%)		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
PositionRank (paper)	11.1	5.6	7.3	10.8	11.1	10.6	9.8	15.3	11.6	9.2	18.9	12.1
Theme based scores	12.6	6.3	8.4	12.1	12.2	12.1	11.1	16.8	13.3	9.6	19.3	12.8
Our method	18.5	9.3	12.4	15.2	15.3	15.3	12.4	18.8	15.0	10.4	21.1	14.0

Table 1: Results on KDD dataset

Methods	N=2 (%)			N=4 (%)			N=6 (%)			N=8 (%)		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Position Rank (paper)	11.3	5.3	7.0	11.3	10.5	10.5	10.8	14.9	12.1	9.9	18.1	12.3
Theme based scores	13.5	5.9	8.2	12.1	10.5	11.3	10.1	13.2	11.5	9.0	15.8	11.5
Our method	16.8	7.2	10.1	13.8	12.0	12.8	11.5	15.0	13.0	9.9	17.2	12.5

Table 2: Results on WWW dataset

Figure 2: Comparison between the F1 scores for WWW dataset



Comparing the F1 scores in figures 1 and 2, we see that our combined model consistently outperforms the other models.

7 CONCLUSION AND FUTURE WORK

In conclusion, we proposed a model that extracts keyphrases using both phrase positions and phrase embedding similarities. Our method outperforms both Position Rank model and the theme weighted PageRank model on the KDD and WWW datasets. The performance of the model in all experiments are as expected and the model is proven to be efficient at keyphrase extraction.

Even though our model outperforms the PositionRank model and the theme weighted model, there are still ways to improve it. In the position bias part, currently, we are assuming that the phrase appears earlier will be more important. However, it isn't always true. We may be able to add more restriction on positions of phrases to make position bias more accurate. For the embedding similarly part, we are currently using pre-trained FastText embeddings as the word embedding. Embeddings contain more information such as BERT embeddings can be tried in the future. Also, we can train the FastText embeddings on scholarly data to bias the embeddings towards capturing scientific terms in a better way in the word vector representations. The performance should improve again with more advanced embeddings.

Features other than positional and theme information can also be used on top of biased word vector representation to improve performances of unsupervised keyphrase extraction models in future.

8 OVERALL EXPERIENCE

The project opened up a new domains of information retrieval, natural language processing and using word vector representation for specific tasks for us in this project.

REFERENCES

[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[2] Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1435–1446.

[3] Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics* 16, 1 (1990), 22–29.

[4] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research* 12, Aug (2011), 2493–2537.

[5] Corina Florescu and Cornelia Caragea. 2017. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1105–1115.

[6] Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. 1999. Domain-Specific Keyphrase Extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 668–673.

[7] Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[8] Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, 365–373.

[9] Kazi Saidul Hasan and Vincent Ng. 2014. 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)*. 1262–1273.

[10] Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*. Association for Computational Linguistics, 216–223.

[11] Amy N Langville and Carl D Meyer. 2004. Deeper inside pagerank. *Internet Mathematics* 1, 3 (2004), 335–380.

[12] Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, 366–376.

[13] Debanjan Mahata, John Kuriakose, Rajiv Shah, and Roger Zimmermann. 2018. Key2vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 634–639.

[14] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 55–60.

[15] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*. 404–411.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[17] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[18] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[19] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. Introduction to information retrieval. In *Proceedings of the international communication of association for computing machinery conference*. 260.

[20] Rui Wang, Wei Liu, and Chris McDonald. 2014. Corpus-independent generic keyphrase extraction using word embedding vectors. In *Software Engineering Research Conference*, Vol. 39.

[21] Rui Wang, Wei Liu, and Chris McDonald. 2015. Using word embeddings to enhance keyword identification for scientific publications. In *Australasian Database Conference*. Springer, 257–268.