Addressing Overgeneration Error: An Effective and Efficient Approach to Keyphrase Extraction from Scientific Papers

Haofeng Jia and Erik Saule

Dept. of Computer Science, UNC Charlotte, USA
{hjia1,esaule}@uncc.edu

Abstract. Keyphrases provide a concise summary of a document and play an important role for many other tasks like searching and clustering. With the large and increasing amount of online documents, automatic keyphrase extraction has attracted much attention. Existing unsupervised methods suffer from overgeneration error, since they typically identify key "words" and then return phrases that contain keywords as keyphrases. To alleviate this problem, we propose an unsupervised ranking scheme directly on "phrases" by exploring essential properties of keyphrases such as informativeness and positional preference. Experiments on two datasets show our approach significantly alleviates the overgeneration error and obtains improvement in performance over state-of-the-art keyphrase extraction approaches.

1 Introduction

Keyphrases are the words and phrases that provide a brief and precise description for a document. Automatically extracting keyphrases from a text document is a fundamental but hard problem which can benefit many tasks, such as document summarization, categorization, searching, indexing, and clustering.

This problem was traditionally solved by supervised methods that convert the problem to a binary classification problem, where a classifier will be trained to identify whether a phrase is a keyphrase or not. For such supervised methods, a lot of high-quality training data are required in order to reach a good performance. Although different learning algorithms have been employed to train the classifier, such as Naive Bayes [1], decision tree [2][3], logistic regression [4][5] and SVM [6][7], most efforts of research on supervised keyphrase extraction are made on feature selection, which turns out to have more significant impact on performance.

In the line of unsupervised research, despite the robust performance of TF-IDF, graph-based methods attract more attention. These methods construct a word graph from each document, such that nodes correspond to words and edges correspond to semantic relationships between words. Then words are scored according to graph centrality measures like PageRank. Finally the phrases consisting of top ranked words are returned as keyphrases. Recent work has

incorporated the positions of a word's occurrence into graph-based model and propose a position biased unsupervised approach [8].

Even though there is a vast literature on the automatic keyphrase extraction problem, state-of-the-art methods, would they be supervised or not, do not achieve satisfying performance.

Recent work has shown that most errors made by state-of-the-art keyphrase extraction systems are due to overgeneration [9]. According to Hasan et al. [10], overgeneration errors contribute to 28%−37% of the overall error. Overgeneration errors occur when a system erroneously outputs other candidates as keyphrases because they contain the highly scored word. Current keyphrase extraction systems typically assign scores to words firstly, and rank candidate phrases according to teh sum of weights of their component words. Therefore, this kind of mechanism tends to suffer from overgeneration errors. Table 1 shows an example of top 8 predicted keyphrases generated by SingleRank [11], a typical unsupervised keyphrase extraction method. The golden keyphrases (manually assigned by the authors) are marked in bold. Since the words "graph", "k-partite" and "structure" receive high scores, thus every candidate phrase that contains these words tends to be ranked as a keyphrase. As we can see, there are many top ranked keyphrases actually have the same or very similar semantics. These overgeneration errors significantly decrease the precision.

Table 1: Overgeneration Errors

Top k	SingleRank
1	original k-partite graph
2	**k-partite graph**
3	hidden structures
4	various structures
5	local cluster structures
6	global cluster structures
7	**relation summary network**
8	general model

In order to alleviate this problem, we look for a way to allow us to directly operates on phrases instead of their component words. Before doing any operation, a system should firstly generate a list of quality candidate phrases from each document, where a quality phrase means a continuous sequence of words with coherent semantics.

Therefore, two questions come to us: What kinds of properties make a sequence of words into a quality phrase? Then what kinds of properties make a phrase into a keyphrase? In this work, we explore these properties and propose KeyPhraser, which generates candidate phrases and ranks them by taking each phrase as one semantic unit. Through experiments carried on two datasets, we show that our approach improves the performances significantly on various metrics.

The paper is organized as follows: In section 2, we summarize related work from supervised keyphrase extraction methods to unsupervised ones. In Section 3, we define the problem and propose KeyPhraser, which is an effective and efficient approach to keyphrase extraction. Then we present the experiments and results in section 4. Finally, we conclude the paper in section 5.

2 Related Work

In general, keyphrase extraction techniques can be classified into two groups: supervised learning approaches and unsupervised ranking approaches [10].

Traditionally, supervised approaches recast the keyphrase extraction task as a binary classification problem. Given a set of annotated documents, the goal is to train a classifier to determine whether a candidate phrase is a key phrase. Various features and classification algorithms give rise to different models.

Although different learning algorithms have been employed to train the classifier, such as Naive Bayes [1], decision tree [2][3], logistic regression [4][5] and SVM [6][7], most efforts of research on supervised keyphrase extraction are made on feature selection, which turns out to have more significant impact on performance.

Textual features like term frequency and inverse document frequency play an important role for supervised keyphrase extraction. Frank et al. [1] developed a system using Naive Bayes as the classifier, named KEA , which is based on text features. Later work explore other textual features to perform consistently well for web pages and scientific articles [4][5][12][13] [14][7][15].

Many studies [3][16] suggest linguistic knowledge is helpful. For example, Hulth et al. [17] claim that part-of-speech sequences of keyphrases are similar. *Acronym* [18][5] and *suffix sequence* [18][5] are also used to capture the propensity of English to use certain Latin derivational morphology for technical keyphrases.

External Knowledge Features are also explored by previous work. Medelyan et al. [14] extend KEA by features extracted from *Wikipedia*. Similarly, *GRISP* [19][7], a large scale terminological database for technical and scientific domains, and *query logs* [4][20] are also used to gain better performance on web pages.

In particular, some types of documents have explicit structures. For instance, a scientific paper has various *sections*. Given this fact, some work try to design features that encode the structural information and improvements have been shown on data set consisting of scientific articles [18][5][7] or web pages [4].

Recently, Caragea et al. [21][22] point out that, citation context structure information have the potential to improve keyphrase extraction. As we know, scientific papers are highly inter-connected in *citation networks*, where papers cite or are cited by other papers in appropriate context [23]. CeKE [21] combines textual features from the target paper, as well as features extracted from the citation networks to extraction keyphrases from scientific artilces.

Besides supervised approaches, there is also an unsupervised line of research on automatic keyphrase extraction. Intuitively, the keyphrase extraction task is looking for phrases that are important. Therefore, various methods are proposed

to score words, which are later aggregated to obtain scores for phrases. Statistical measures [24] have been shown to work well in practice.

Graph-based ranking is now becoming more and more popular for keyphrase extraction task. The idea behind graph-based methods is to construct a graph that represents the text and encodes the relationship between words in a meaningful way. Typically, words appearing in the text will be taken as nodes, and edges represent semantic relationships between words. Then, the keyphrase extraction task is transferred into a graph ranking problem based on the importance of nodes. The importance of a word is determined by its relatedness to others. In other words, a word is important if it is related to a lot of words or some words that are important. Each edge can be deemed as a vote from one node to another. After convergence, graph-based methods select top ranked nodes as keywords.

The basic graph-based method is TextRank[25]. An unweighted text graph is constructed where nodes represent words and edges indicate two words co-occur within a certain window size in the text. Now the goal is to extract the keywords on this undirected word graph. So PageRank[26] is employed here to compute a score for each word indicating how important it is. After convergence, the T% top scored words are extracted as keywords. Finally, adjacent keywords are collapsed and output as a keyphrase.

SingleRank [11] expands TextRank by constructing a weighted graph rather than a unweighted graph for each document. In this work, a weight is assigned to each edge according to the number of times the corresponding words co-occur within a window size. SingleRank prefers the window size of 10, while TextRank uses 2. After scoring words in the same way as TextRank, all noun phrases are taken into consideration and each phrase is scored by summing up the scores of words it contains. Based on SingleRank, Wan et al. [11] also make efforts to improve the performance by exploring textual-similar neighborhood documents. Inspired by TextRank, Boudin [27] explores various centrality measures, such as degree, closeness and betweenness, for keyphrase extraction task.

Recently, the idea of k-core degeneracy [28][29] is also applied in the word graph for keyphrase extraction [30]. Compared with those approaches solely based on centrality, k-core degeneracy takes better into account proximity between key words and variability in the number of extracted keywords through the selection of more cohesive subsets of nodes.

Along with the rise of deep learning, the distributed word representations [31][32], also called word embedding, are becoming popular. Wang et al. [33] propose a graph-based ranking approach that uses word embedding vectors as the background knowledge. The key contribution of this approach is the proposed weighting scheme, which is referred as word attraction score. Moreover, positional preference has been shown its potential for keyphrase extraction systems [34][8].

Existing approaches typically score individual words, and then aggregate words to obtain scores for phrases. This framework suffers from overgeneration error because all phrases that contain highly scored words are very likely to be

returned as keyphrases. In the next section, we propose a method which tries to capture essential properties of keyphrases. Our approach is designed to alleviate the issue of overgeneration error.

3 Proposed Approach

In this section, we start with the traditional framework for unsupervised keyphrase extraction systems. Then we introduce KeyPhraser, a fully unsupervised keyphrase extraction approach that directly operates on phrases.

3.1 Unsupervised Keyphrase Extraction

A classic unsupervised keyphrase extraction system typically contains three steps:

- The first step is to generate a list of candidate word that have potential to be keywords. Typically, words with certain part-of-speech tags such as adjectives and nouns are considered. An alternative way is simply filtering out stop words from the documents
- The second step is actually ranking or scoring candidate words, which are generated from last step. This is the core step and various ranking algorithms are proposed.
- The final step is called keyphrase formation, where the candidate words are used to form keyphrases through certain aggregation function like sum.

As we can see from Fig. 1a, current unsupervised keyphrase extraction systems typically assign scores to words firstly, and then form keyphrases according to the sum of weights of their component words. A phrase that contain a highly scored word are very likely to returned as a keyphrase. Therefore, current methods tends to suffer from overgeneration errors.

3.2 KeyPhraser

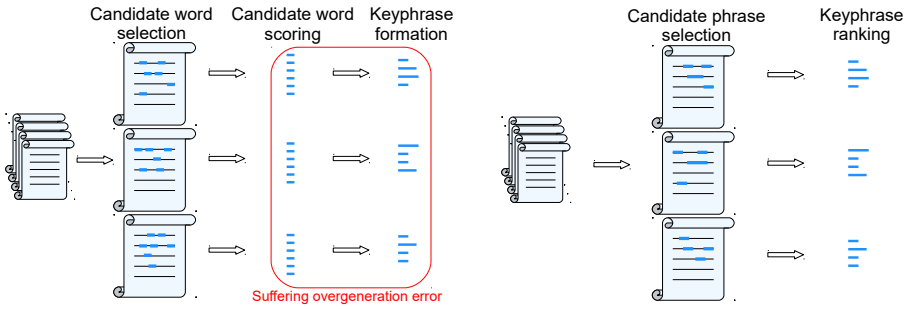
In order to alleviate this problem, we look for a scheme to directly operate on phrases instead of their component words (Fig. 1b). In other words, our method should be capable of extracting phrases from the text and then selecting keyphrases from these candidate phrases based on reasonable measures.

Therefore, the following questions come to us:

- What kinds of properties make a group of words into a phrase?
- What kinds of properties make a phrase into a keyphrase?
- What is special for scientific documents?

To capture these properties, we define four metrics in this section: concordance, popularity, informativeness and positional preference.

Let's start with the first question, which corresponds the candidate phrase selection part in Fig. 1b. Before doing any operation, a system should firstly



(a) Classic Scheme (b) Keyphraser Scheme

Fig. 1: Keyphrase Extraction Schemes

generate a list of quality candidate phrases from each document, where a quality phrase means a small group of words that appear contiguously in the text and serve as a whole semantic unit in certain context. In practice, extracting phrases from document turns out to be a nontrivial problem.

Concordance is also called phraseness, which measures the likelihood that a sequence of words can be considered as a phrase. Several definitions that quantify the discrepancy between the probability of their true collocation and the presumed collocation under independence assumption are used to capture concordance, such as pointwise mutual information [35] and Dice coefficient [33].

However, in order to achieve a reasonable concordance score, PMI and Dice coefficient require that the corpus of English text is large enough.

In the context of keyphrase extraction, part-of-speech tag is widely used to measure concordance. Typically, words tagged as adjectives or nouns are selected, then a continuous sequence of candidate words is considered as a phrase:

$$Conc(s) = \begin{cases} 1 & \text{if } s = [adj]^*[noun]^+ \\ 0 & \text{otherwise} \end{cases}$$

We use this scheme in KeyPhraser to extract phrases from documents because of two reasons. First, publicly available datasets for keyphrase extraction task typically contain hundreds of documents, which can not guarantee a good performance for PMI or Dice efficient; most existing keyphrase extraction algorithms extract candidate word by part-of-speech tags, therefore, we choose to be consistent with these works.

Now given a list of candidate phrases, we need to identify keyphrases out of them. This is called keyphrase ranking in Fig. 1b. To this end, we need to figure out the properties that make a phrase into a keyphrase.

Popularity is the first property coming to mind. As we know, keyphrases are those phrases that provide a brief and precise description for the given document. So they should occur with sufficient frequency in the given document. Intuitively,

term frequency is a good criteria to measure the popularity of a phrase. We use a sublinear variant of term frequency in KeyPhraser, which is:

$$Pop(s,d) = log(f(s,d)+1)$$

where $f(s,d)$ denotes the frequency of a phrase $s \in \mathcal{P}$ in the document d.

Informativeness For a given document, some candidate phrases tends to be less informative or unrelated to the main topics, even though they appear frequently. Generally speaking, these phrases are likely to be functional phrases in English. Therefore, it is difficult to measure informativeness only based on the information of the current document.

Inverse document frequency is a traditional information retrieval measure of how much information a word provides in order to retrieve a small subset of documents from a corpus. The IDF of a phrase is usually calculated as the average IDF scores of the words it contains. Here we take a phrase as an unit and customise the inverse document frequency for phrases:

$$Info(s) = log\frac{|\mathcal{C}|}{|d \in \mathcal{D} : s \in d|}$$

where $\mathcal{C}$ means the whole corpus, and $\mathcal{D}$ means the documents that contain candidate phrase s.

Positional Preference Where a phrase occurs in the document is also essential to the keyphrase extraction problem, especially for scientific papers. Intuitively, given a scientific document, keyphrases tends to appear not only frequently but also early. For instance, titles of scientific articles are very likely to contain keyphrases.

Previous work has shown the power of positional information of words [34][8]. In this paper, we define the positional preference of each phrase by considering all occurrence positions in the document:

$$Pos(s,d) = log(\sum_{\text{each s in d}} \frac{|d|}{op(s,d)+1})$$

where $op(v,d)$ denotes an occurrence position of phrase v in document d. An alternative way only takes the first occurrence position of a phrase into consideration.

$$Pos(s,d) = log\frac{|d|}{fop(s,d)+1}$$

where $fop(v,d)$ denotes the first occurrence position of phrase v in document d.

Finally, In order to build a keyphrase extraction system based on above measures, one can aggregate them in multiple ways. Statistical method like TF-IDF has been proven to be a strong and robust baseline according to many previous work despite the simplicity of aggregation function. Therefore, we also use product to aggregate above measures.

$$Keyphraser(s,d) = Conc(s)Pop(s,d)Info(s)Pos(s,d)$$

In this paper, we explore two different versions of KeyPhraser: *KeyPhraser-full* which use all occurrence positions and *KeyPhraser-fp* which only use the position of first occurrence.

4 Experiments and Results

In this section, we conduct experiments on real datasets to demonstrate the effectiveness and efficiency of our proposed approach to the task of keyphrase extraction.

4.1 Dataset and Experiment Settings

In order to evaluate the performance of our method, we conducted experiments on two public datasets, which were made available by Gollapalli and Caragea[1]. The datasets consist of research papers from two top-tier conferences: World Wide Web (WWW) and Knowledge Discovery and Data Mining (KDD). All titles and abstracts are used for keyphrase extraction, and the author assigned keyphrases are used as ground truth for evaluation.

In specific, the KDD dataset contains 755 papers and the WWW dataset consists of 1331 papers. (The KDD dataset actually contains 834 papers, but 79 of them do not have corresponding ground truth files. Similar for WWW.) The average numbers of ground truth keyphrases for each paper in these two datasets are 3.8 and 4.6 respectively. The average number of words in each ground truth keyphrase is 1.8 for the KDD dataset and 1.9 for the WWW dataset. There are few ground truth keyphrase consisting of more than 4 words. Therefore, we set 4-grams as the threshold for candidate phrases for all method used in the experiments.

In our experiments, we use average precision, recall and F1-score as performance metrics, since they are widely used in keyphrase extraction task. To demonstrate the effectiveness of the proposed approach, we compared it with popular baselines and state-of-the-art algorithms: TF-IDF, TextRank, SingleRank and PositionRank.

For most keyphrase extraction approaches, the number of phrases as output are typically determined by users. Here we examine the top k performance of our method, where k is set with the range from 1 to 8. The range is determined by the following three reasons: Firstly, the average number of ground truth keyphrases of the datasets is around 4; Secondly, overgeneration error results in lower precision, which means this type of error occurs more frequently for small k; Finally, in practice, a keyphrase extraction system is not expected to generate plenty of phrases, otherwise the generated keyphrases will be less usefull.

Please note that TextRank is kind of special, as it requires a ratio (of top-ranked words) instead of a specific k as input. For fair comparison, we use corresponding ratio for each k, so that TextRank will generate almost the same number of phrases as others.

[1] `https://www.cse.unt.edu/~ccaragea/keyphrases.html`

Window size is a typical parameter of graph-based keyphrase extraction methods, such as PositionRank and SingleRank. While this parameter seems to have a great impact on the built word graph, previous work has shown that graph-based methods are not really sensitive to it. To be consistent with other work, we set window size of 10 for PositionRank and SingleRank, and window size of 2 for TextRank.

Some previous work use Porter Stemmer to reduce both predicted and ground truth keyphrases to a base form. In this way, the number of miss-matched pairs of keyphrases due to the gap in lexical form will be decreased. However, Porter Stemmer is inappropriate under some circumstances. For instance, "clusterings" and "clusters" usually don't share the same meaning in computer science context. In our experiments, we only use simple ad-hoc processing to match keyphrases in singular/plural form.

4.2 Results and Discussion

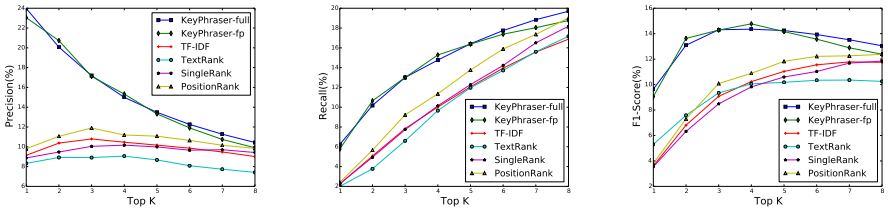


Fig. 2: Performance on the KDD dataset

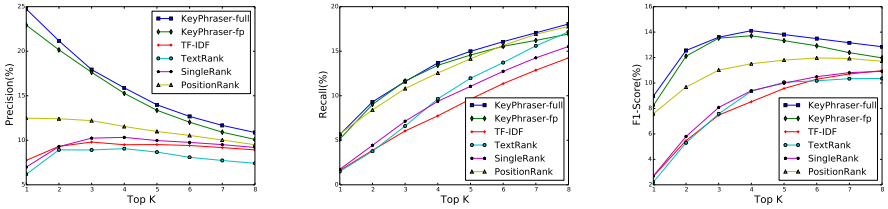


Fig. 3: Performance on the WWW dataset

Fig. 2 shows the performance of our method comparing with TF-IDF, TextRank, SingleRank and PositionRank on the KDD dataset, and Fig. 3 is for the WWW dataset. As can be seen from the figures, KeyPhraser significantly outperforms other approaches on both datasets.

The major improvements on F1-scores come from the substantial improvements on precision, especially for small k. This is because current methods typically rank a candidate phrase by aggregate scores of words it contains. On the

contrary, our method directly operates on phrases, which turns out to be effective to alleviate the overgeneration issue.

In particular, for the KDD dataset, state-of-the-art method achieves 9% on precision when k equals 1, while KeyPhraser achieves 23% for the same k, which means the improvement by our approach at this point is as high as 155%. For example, KeyPhraser achieves F1-scores of 14.3% and 14.1% for k equals 3 and 5 respectively on the same dataset, as comparison, the best state-of-the-art method, PositionRank, achieves F1-scores 10.1% and 11.8% for corresponding k.

Generally speaking, our method tends to find the "correct" keyphrases much "faster" than others. We can easily conclude that based on a preliminary analysis of recall and precision curves:

- First of all, if you look at the recall curves of all methods, a obvious finding is that they tends to converge when k is large enough. This is true because each method in the plot has employed part-of-speech tags to generate candidate phrases or words, which means the pool where the keyphrases are selected from is pretty much the same. In other words, these methods share the same upper bound of recall. (One can learn more about upper bound of recall from [36])
- Now look at the precision curves. For small k, KeyPhraser outperforms other methods by a substantial improvement. This is due to the fact that overgeneration error occurs more frequently when k is small. Along with the number of output getting larger, the difference between returned keyphrases by difference methods becomes less significant, which is reflected in the plots.

For real systems, performance improvement for small k is much more useful. Because a document usually contains a few keyphrases. A keyphrase extraction method that generates a bunch of phrases to obtain a good performance is not helpful in practice.

Table 2 shows result of top 8 predicted keyphrases by different methods for a instance paper from the KDD dataset, where the ground truth keyphrases are marked in bold. As we can see, compared with existing methods (upper part of the Table 2), our methods (lower part of the Table 2) alleviate the overgenration errors and obtains a higher precision. In other words, our methods tend to find ground truth keyphrases faster.

Beside the cheerful performance on effectiveness, KeyPhraser remains a linear time complexity to the corpus size. The efficiency is due to the simplicity of the aggregation function of measures. In specific, on the KDD dataset, KeyPhraser is 3x faster than graph-based methods and 2x faster on the WWW dataset.

Error Analysis. Hasan et al. [10] classify all errors of keyphrase extraction systems into four categories: overgeneration error, infrequency error, redundant error and evaluation error. essentially, redundant error and evaluation error are kind of similar as they both stem from two phrases being semantically equivalent. Overgeneration error comes from generating multiple phrases that contain a popular word without the phrase making much sense. While infrequency error come from a keyphrase appearing only once or twice in the entire document.

Table 2: Predicted Keyphrases Comparison

k	SingleRank	PositionRank
1	original k-partite graph	original k-partite graph
2	**k-partite graph**	**k-partite graph**
3	hidden structures	various structures
4	various structures	hidden structures
5	local cluster structures	local cluster structures
6	global cluster structures	global cluster structures
7	**relation summary network**	**unsupervised learning**
8	general model	**relation summary network**
k	KeyPhraser-fp	KeyPhraser-full
1	**unsupervised learning**	**k-partite graph**
2	**k-partite graph**	hidden structures
3	hidden structures	**unsupervised learning**
4	data objects	**relation summary network**
5	multiple types	clustering approaches
6	**relation summary network**	data objects
7	general model	multiple types
8	local cluster structures	connections

Since the methods we are investigating do not dig in the semantics of the extracted phrases we believe that overgeneration, redundant and evaluation error are not usefully different and we classify the errors in the typical two category.

The first type of system errors is False Negative Error, this error happens when a gold phrase is not returned as a keyphrase. Infrequency error is a typical false positive error. Existing method are likely to miss it due to the difficulty to detect such an infrequent phrase. To recall these infrequent phrases, we may have to accept lower precision.

The other type of system errors is False Positive Error which happens when candidate phrases are incorrectly returned as keyphrases. Overgeneration error is a typical false negative error and certainly the most common one when manually looking at the automatically extracted key phrases.

5 Conclusion and Future Work

In this paper, we presented KeyPhraser, an unsupervised keyphrase extraction approach for scientific papers addressing overgeneration error. To this end, we look for a way to allow us directly operates on phrases instead of their component words. KeyPhraser takes each phrase as one semantic unit. Firstly candidate phrases are generated by concordance measure, and then they are scored by three other measures to determine whether a phrase is a keyphrase or not. Despite the simplicity of the mechanism, experiments carried on two datasets demonstrate KeyPhraser is an effective and efficient approach to keyphrase extraction.

In future, various concordance, informativeness and positional measures should be explored. For example, how to find a way to incorporate more positional in-

formation rather than just the position of first occurrence. And finding other effective aggregation functions of phrase measures seems promising. Moreover, it would be interesting to explore more phrase based approaches. For instance, we wonder how to build a phrase graph in a reasonable way and how is it compared with word graph.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1652442.

References

1. Frank, E., Paynter, G.W., Witten, I.H., Gutwin, C., Nevill-Manning, C.G.: Domain-specific keyphrase extraction. In: IJCAI. Volume 99. (1999) 668–673
2. Turney, P.: Learning to extract keyphrases from text. In: National Research Council Canada, Institute for information Technology, Technology Report. (1999) ERB–1057
3. Turney, P.D.: Learning algorithms for keyphrase extraction. Information Retrieval **2**(4) (2000) 303–336
4. Yih, W.t., Goodman, J., Carvalho, V.R.: Finding advertising keywords on web pages. In: Proceedings of the 15th international conference on World Wide Web, ACM (2006) 213–222
5. Kim, S.N., Kan, M.Y.: Re-examining automatic keyphrase extraction approaches in scientific articles. In: Proceedings of the workshop on multiword expressions: Identification, interpretation, disambiguation and applications, Association for Computational Linguistics (2009) 9–16
6. Jiang, X., Hu, Y., Li, H.: A ranking approach to keyphrase extraction. In: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, ACM (2009) 756–757
7. Lopez, P., Romary, L.: Humb: Automatic key term extraction from scientific articles in grobid. In: Proceedings of the 5th international workshop on semantic evaluation, Association for Computational Linguistics (2010) 248–251
8. Florescu, C., Caragea, C.: 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). Volume 1. (2017) 1105–1115
9. Boudin, F.: Reducing over-generation errors for automatic keyphrase extraction using integer linear programming. In: ACL 2015 Workshop on Novel Computational Approaches to Keyphrase Extraction. (2015)
10. Hasan, K.S., Ng, V.: Automatic keyphrase extraction: A survey of the state of the art. In: In Proceedings of the Annual Meeting of the Association for Computational Linguistics. (2014) 1262–1273
11. Wan, X., Xiao, J.: Single document keyphrase extraction using neighborhood knowledge. In: AAAI. Volume 8. (2008) 855–860
12. Kim, S.N., Medelyan, O., Kan, M.Y., Baldwin, T.: Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In: Proceedings of the 5th International Workshop on Semantic Evaluation, Association for Computational Linguistics (2010) 21–26

13. Dredze, M., Wallach, H.M., Puller, D., Pereira, F.: Generating summary keywords for emails using topics. In: Proceedings of the 13th international conference on Intelligent user interfaces, ACM (2008) 199–206
14. Medelyan, O., Frank, E., Witten, I.H.: Human-competitive tagging using automatic keyphrase extraction. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3, Association for Computational Linguistics (2009) 1318–1327
15. Kim, S.N., Medelyan, O., Kan, M.Y., Baldwin, T.: Automatic keyphrase extraction from scientific articles. Language resources and evaluation **47**(3) (2013) 723–742
16. Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge. In: Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing. EMNLP '03, Stroudsburg, PA, USA, Association for Computational Linguistics (2003) 216–223
17. Hulth, A., Megyesi, B.B.: A study on automatically extracted keywords in text categorization. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, Association for Computational Linguistics (2006) 537–544
18. Nguyen, T.D., Kan, M.Y.: Keyphrase extraction in scientific publications. In: Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers. Springer (2007) 317–326
19. Lopez, P., Romary, L.: Grisp: A massive multilingual terminological database for scientific and technical domains. In: In Seventh international conference on Language Resources and Evaluation (LREC). (2010)
20. Turney, P.: Coherent keyphrase extraction via web mining. In: International Joint Conference on Artificial Intelligence IJCAI-03. (2003)
21. Caragea, C., Bulgarov, F.A., Godea, A., Gollapalli, S.D.: Citation-enhanced keyphrase extraction from research papers: A supervised approach. In: EMNLP. (2014) 1435–1446
22. Gollapalli, S.D., Caragea, C.: Extracting keyphrases from research papers using citation networks. In: AAAI. (2014) 1629–1635
23. Bulgarov, F., Caragea, C.: A comparison of supervised keyphrase extraction models. In: Proceedings of the 24th International Conference on World Wide Web Companion, International World Wide Web Conferences Steering Committee (2015) 13–14
24. EI-Beltagy, S., Rafea, A.: Kp-miner:a keyphrase extraction system for english and arabic documents. In: Information Systems. (2009) 132–144
25. Mihalcea, R., Tarau, P.: Textrank: Bringing order into texts. In: In Proceedings of the Empirical Methods in Natural Language Processing. (2004) 404–411
26. Page, L., Brin, S., Motwani, R., Winograd, T.: Pagerank: Bringing order to the web. Technical report, Stanford Digital Libraries Working Paper (1997)
27. Boudin, F.: A comparison of centrality measures for graph-based keyphrase extraction. In: International Joint Conference on Natural Language Processing (IJCNLP). (2013) 834–838
28. Bollobás, B.: The evolution of random graphs. Transactions of the American Mathematical Society (1984) 257–274
29. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science (1999) 509–512
30. Rousseau, F., Vazirgiannis, M.: Main core retention on graph-of-words for single-document keyword extraction. In: Advances in Information Retrieval. Springer (2015) 382–393

31. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: arXiv preprint arXiv:1301.3781. (2013)
32. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. (2013) 3111–3119
33. Wang, R., Liu, W., McDonald, C.: Corpus-independent generic keyphrase extraction using word embedding vectors. In: In Proceedings of the Conference on Web Search and Data Mining Workshops. (2015) 834–838
34. Florescu, C., Caragea, C.: A position-biased pagerank algorithm for keyphrase extraction. In: AAAI. (2017) 4923–4924
35. Liu, J., Shang, J., Wang, C., Ren, X., Han, J.: Mining quality phrases from massive text corpora. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM (2015) 1729–1744
36. Wang, R., Liu, W., McDonald, C.: How preprocessing affects unsupervised keyphrase extraction. In: In Proceedings of the CICLing Conference on Intelligent Text Processing and Computational Linguistics. (2014) 163–176