Progress in Artificial Intelligence (2020) 9:77–83
https://doi.org/10.1007/s13748-019-00200-3

REGULAR PAPER

An unsupervised keyphrase extraction model by incorporating structural and semantic information

Linkai Luo[1] · Longmin Zhang[1] · Hong Peng[1]

Received: 30 March 2019 / Accepted: 10 October 2019 / Published online: 26 October 2019

Abstract
We proposed an unsupervised keyphrase extraction model that incorporates the structural information and the semantic information of a document. The structural information refers to the directed graph that is composed of keyphrase candidates and topics. The weight between two candidates is computed by their relative distance in the document and the positions of the corresponding sentences. Graph ranking algorithm is then applied to get the structural scores of the candidates. Then, the semantic score is obtained by the similarity between candidate and all sentences. The final score of a candidate is the sum of the structural score and the semantic score. The top N candidates with the highest scores are selected as the recommended keyphrases. The comparison experiments on three widely used datasets show that our model achieves the best results in the long documents and a competitive result in the short document. It indicates that our model is effective and is superior to the state-of-the-art unsupervised models.

Keywords Keyphrase extraction · Unsupervised model · Structural information · Semantic information · Graph-based model

1 Introduction

Keyphrase extraction is an important issue in natural language processing. Keyphrases are very helpful in many natural language processing tasks, such as text classification, text clustering, recommendation systems, and search engine [1].

The task of keyphrase extraction is to select the important and topical phrases that best describe a document [2,3]. According to the definition of the task, the core problem of keyphrase extraction is to determine which candidate phrases can best describe a document. Further, keyphrase extraction needs to determine the importance of each candidate phrases.

Many approaches have been proposed in keyphrase extraction. These approaches can be roughly divided into two categories: supervised methods and unsupervised methods.

✉ Hong Peng
penghong@xmu.edu.cn

Linkai Luo
luolk@xmu.edu.cn

Longmin Zhang
zhanglmxmu@163.com

[1] Department of Automation, Xiamen University, Xiamen, People's Republic of China

Supervised methods translate a keyphrase extraction problem into a binary classification problem, in which candidate phrases are classified as keyphrase or non-keyphrase [4]. All classification methods can be used in the supervised approaches. However, the supervised approaches require a lot of labeled data, which may be impractical in some domains. Thus, the unsupervised approaches attract more and more attention [1]. Many unsupervised methods have been used in keyphrase extraction, such as language modeling, clustering or graph-based ranking [4]. Among them, graph-based ranking provides impressive results [5].

The graph-based ranking is inspired by Google's PageRank that is proposed to analyze the link structure of the World Wide Web [3]. In graph-based ranking, a document is modeled as a graph where vertexes are phrases, and edges are the relationships between phrases. The main idea of graph-based ranking approaches is that a vertex is important if it is related to a large number of vertexes or related to important vertexes [4]. The graph-based ranking focuses on measuring the semantic relation between vertexes or designing vertex ranking functions [5]. There are a lot of studies about graph-based ranking. For example, Wan and Xiao [6] apply a graph-based ranking algorithm on a graph that is composed of a document and several adjacent documents similar to the document. Bougouin et al. [7] present a graph-based

keyphrases extraction approach named TopicRank. In TopicRank, candidate keyphrases are first clustered into topics, a graph-based ranking algorithm is then applied to each topic, and the most representative keyphrases from each topic are chosen [7]. Boudin [5] proposed an unsupervised keyphrase extraction model that encodes topical information within a multipartite graph structure. This model is an improvement of the TopicRank. As sorting candidate keyphrases in a single operation, it implicitly enforces topical diversity and avoids accumulation of errors [5].

Based on the TopicRank and its improvements, we proposed a new model that incorporates semantic information and structural information of a document to keyphrase extraction. In [5], the weight between two vertexes is firstly computed by the distance of the two candidates in different topics and then adjusted by the position of candidates. Though the position of candidate can provide important information, it is too sensitive and often leads to unexpected results [1]. To improve this insufficiency, we use the position of the sentence where the candidate is located to adjust the weight between two vertexes instead of the absolute position of candidate. In addition, we incorporate the semantic information of candidates in keyphrase extraction. Since keyphrases are single or multi-word phrases that best describe the document [3], the semantic relation between keyphrases and sentences certainly includes valuable information. Thus, incorporating the semantic information of candidates is helpful. Our contributions are as follows:

- We proposed an unsupervised model that incorporates the structural information and the semantic information of a document to score and rank candidate keyphrases.
- We aggregate the information from all positions of sentences that a candidate phrase is located in to adjust the weight between candidate phrases, which is an improvement of graphical presentation of a document.
- We proposed a new method to get the semantic information from a document, and it shows that the semantic information of a document is helpful in scoring and ranking candidate keyphrases.

We experimentally evaluate our model on three datasets that are widely used in keyphrase extraction, and its superiority is shown by the comparisons of unsupervised graph-based model that do not take into account sentence positions and semantic information, as well as some baselines for unsupervised keyphrase extraction. The rest of the paper is organized as follows: The related work is summarized in Sect. 2. The details of our proposed model are provided in Sect. 3. Experiments are conducted in Sect. 4. The conclusion and future work are presented in Sect. 5.

2 Related work

Many supervised and unsupervised approaches have been proposed in the task of keyphrase extraction [4]. Supervised methods take keyphrase extraction as a binary classification task. Unsupervised methods include language modeling, clustering or graph-based ranking, etc. Since the pioneering work of Mihalcea and Tarau [3], graph-based methods have been the most widely used unsupervised approaches for keyphrases extraction [7]. Many researchers have devoted a large amount of effort to develop better ways of modeling documents as graphs or ranking vertexes in graphs [5]. Bougouin et al. [7] present a graph-based keyphrase extraction approach named TopicRank. In TopicRank, similar keyphrases in a document are first clustered into a topic [7]. Then, a document can be represented as a graph in which vertexes are topics and the edge between two topics is weighted according to the position relationship of phrases in two topics [7]. As an improved approach of TopicRank, Boudin [5] proposed an unsupervised keyphrase extraction model that encodes topical information within a multipartite graph structure. Different from TopicRank, vertexes of graph in [5] are phrases and the edge between two phrases is weighted according to the position relationship of phrases if they are from different topics. Once the graph is created, the graph-based ranking model TextRank proposed by Mihalcea and Tarau [3] is used both in [5,7] to rank vertexes. In TopicRank, the phrase in a topic which first appears in the document is selected as keyphrase [7].

To capture the semantic relatedness of words in a document, word sense disambiguation is an essential predecessor step. Word sense disambiguation is the core task of human language understanding [8]. So far, researchers have already proposed many approaches for word sense disambiguation.

In this paper, we adopt the state-of-the-art approach proposed by Luo et al. [9] that integrate the context and glosses of the target word into neural network in order to make full use of labeled data and lexical knowledge. The gloss of a word is provided by WordNet that is a famous online lexical database in which English nouns, verbs, adjectives, and adverbs are organized into sets of synonyms [10]. Synonymy, antonymy, hyponymy, meronymy, troponymy, and entailment are semantic relations for words in WordNet [10]. Next, we will briefly describe the model proposed by Luo et al. [9].

The model proposed by Luo et al. [9] consists of gloss module, context module, memory module, and scoring module. The context module encodes a sequence of surrounding words of the target word into a distributed vector representation [9]. The gloss module encodes the gloss of three items, namely the target word, the corresponding hypernyms, and

hyponyms into distributed vector representations [9]. The memory module contains two steps: attention calculation and memory update [9]. The attention calculation models the inner relationship between the context and glosses [9]. The memory states are updated after the attention calculation [9]. Finally, the output of the context module and the output of the memory module are used in scoring module to calculate the scores for all the related senses in WordNet corresponding to the target word [9].

3 Proposed model

In this section, we will describe our approach in detail. Our approach can be divided into two steps. We first build a multipartite graph to represent document in which the edge weights are adjusted with sentence position, and structural score for candidates is obtained by the structure information of document. Semantic score for candidates is then computed by the semantic similarity between the candidate and all the sentences. The structural score and the semantic score are integrated to form the final score. In order to facilitate comparison with other approaches, our candidates are noun phrases that match the regular expression '(adjective)*(noun)+' and the phrases will be stemmed to reduce the number of mismatches. The overall framework is illustrated in Algorithm 1.

Algorithm 1: Overall framework

Input: positions of candidates P, set of candidates C, sentences positions of candidates P_s, set of Sentences S

adjacency matrix W, set of topics T ← Graph construction(P, C)
Graph weight adjustment(P_s, T, W)
Structural score S_{base} ← Graph ranking(W, C)
Semantic score S_{smtc} ← Semantic ranking(C, S)
Final score S_{final} ← Norm(S_{base}) + Norm(S_{smtc})

3.1 Graph-based ranking model

The graph-based ranking model can be subdivided into three stages: (1) the multipartite graph construction at phrases level; (2) the weight adjustment; (3) the structural scores of candidates.

3.1.1 Graph construction

We follow the approaches proposed by Bougouin et al. (2013) to group the candidates into topics in a document. Hierarchical agglomerative clustering (HAC) algorithm is used to automatically group similar candidates into topics [7]. Two candidates are considered similar if they have at least 25% of overlapping words and the candidates are stemmed to eliminate the influence of word form [7]. A directed graph is built as illustrated in Fig. 1 in which the vertexes are connected if they belong to different topics. The weight from vertex i to vertex j is defined as follows:

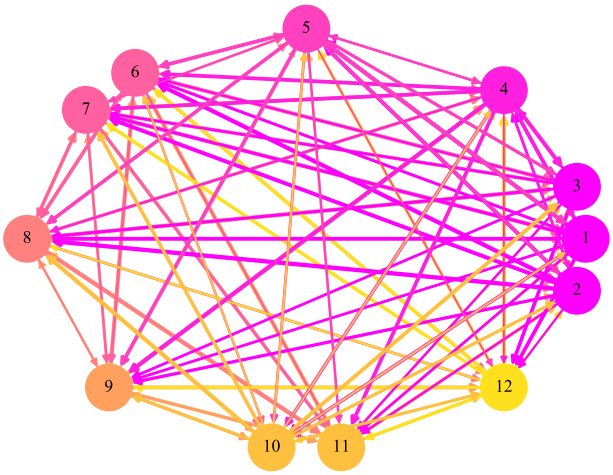

Fig. 1 Multipartite graph representation of a document, where the nodes are candidates and edges are position relationship between candidates. Nodes of the same color belong to the same topic

$$w_{ij} = \sum_{p_i \in P(c_i)} \sum_{p_j \in P(c_j)} \frac{1}{|p_i - p_j|} \tag{1}$$

where c_i and c_j are corresponding candidates and $P(c_i)$ is the set of the word positions of candidate c_i. Algorithm 2 describes the process of graph construction.

Algorithm 2: Graph construction

Input: positions of candidates P, set of candidates C
Output: adjacency matrix W, set of topics T

set of topics T ← Clustering(C)
for T_a, T_b *in* T **do**
  for c_i *in* T_a, c_j *in* T_b **do**
    for p_i *in* $P[c_i]$, p_j *in* $P[c_j]$ **do**
      $W[i][j] \leftarrow W[i][j] + \frac{1}{|p_i - p_j|}$
    end
  end
end
return W, T

3.1.2 Graph weight adjustment

The weight in (1) just captures the distance relations between candidates and is not sufficient to determine the importance of the candidates. The positions of the candidates can provide more information about the importance of the candidates [5].

In [5], the positions of the candidates are used for weight adjustment. Adjustment of the incoming edge weights for the

first occurring candidate of each topic is helpful for keyphrase extraction [5]. The adjustment strategy in [5] results in different importance if candidates are located in different positions in the same sentence. However, it is unreasonable to assign different importance to different candidates in the same sentence based on the order. For example, the active sentence and the passive sentence express the same meaning, but the order of the same words is different. It results in that the same words have different importance. Hence, it may be more reasonable and more robust that the importance of the candidates in a sentence is considered to be the same. Following [5], we adjust the incoming edge weights of the first occurring candidate of each topic. Similar to the method proposed by Boudin [5], candidates that occur at the beginning of the document gather information from the other candidates belonging to the same topic. Different from [5], our method adjusts the incoming edge weights with positions of the sentences in which the candidates located rather than absolute positions of the candidates. Weights of the incoming edge for the first occurring candidate of each topic are adjusted by the following equation:

$$w_{ij} = w_{ij} + \alpha \cdot \exp^{\sum_{p \in P_s(c_j)} \frac{1}{p}} \cdot \sum_{c_k \in T(c_j) \setminus \{c_j\}} w_{ki} \tag{2}$$

where $P_s(c_j)$ is the set of word positions of the sentences where the candidate c_i located, $T(c_j)$ is the set of candidates with the same topic as c_j, and α is a hyperparameter that controls the strength of the weight adjustment. According to the experiments of Boudin (2018), we empirically set $\alpha = 1.1$ [5]. The procedure of graph weight adjustment is illustrated in Algorithm 3.

Algorithm 3: Graph weight adjustment

Input: sentences positions of candidates P_s, set of topics T, adjacency matrix W
for T_a *in* T **do**
  $c_j \leftarrow$ FirstOccur(T_a)
  $c \leftarrow$ Incoming(c_j)
  for c_i *in* c **do**
    t $\leftarrow 0$
    for c_k *in* T_a *and* $c_k \neq c_j$ **do**
      t $\leftarrow$ t + $W[k][i]$
    end
    $t_p \leftarrow 0$
    for p *in* $P_s[c_j]$ **do**
      $t_p \leftarrow t_p + \frac{1}{p}$
    end
    $W[i][j] \leftarrow W[i][j] + \alpha \times \exp^{t_p} \times t$
  end
end

3.1.3 Graph ranking

After the graph is built, we compute the score of candidates by a graph-based ranking algorithm. Here, we use the widely used TextRank algorithm proposed by Mihalcea and Tarau [3]. TextRank algorithm is based on graph-based ranking algorithm named PageRank [3]. A graph-based ranking algorithm uses global information recursively computed from the entire graph to obtain the importance of the vertex [3]. The structural score of the candidate c_i is computed as follows:

$$S_{base}(c_i) = (1 - \lambda) + \lambda \cdot \sum_{c_j \in I(c_i)} \frac{w_{ij} S_{base}(c_j)}{\sum_{c_k \in O(c_j)} w_{jk}} \tag{3}$$

where $I(c_i)$ is the set that is made up of predecessors of c_i, $O(c_j)$ is the set that is made up of successors of c_j, and λ is a damping factor that integrates the probability of jumping from a given vertex to another random vertex in the graph [3]. According to the experiments of Mihalcea and Tarau [3], we empirically set $\lambda = 0.85$. The computation of structural score is illustrated in Algorithm 4.

Algorithm 4: Graph ranking

Input: adjacency matrix W, set of candidates C
Output: Structural score S_{base}
number of candidates N $\leftarrow$ Len(C)
for c_i *in* C **do**
  $S_{base}[c_i] = \frac{1}{N}$
  $Slast_{base}[c_i] = 0$
end
while $\frac{Sum(abs(S_{base} - Slast_{base}))}{N} > 1e-6$ **do**
  $Slast_{base} \leftarrow S_{base}$
  for c_i *in* C **do**
    t $\leftarrow 0$
    $c_{in} \leftarrow$ Incoming(c_i)
    for c_j *in* c_{in} **do**
      $c_{out} \leftarrow$ Outcoming(c_j)
      $t_d \leftarrow 0$
      for c_k *in* c_{out} **do**
        $t_d \leftarrow t_d + W[j][k]$
      end
      t $\leftarrow t + \frac{W[i][j] \times S_{base}[c_j]}{t_d}$
    end
    $S_{base}[c_i] = (1 - \lambda) + \lambda \times t$
  end
end
return S_{base}

3.2 Semantic relatedness model

The semantic relatedness model can be subdivided into two stages: (1) The correct meaning of words in a document is assigned by word sense disambiguation algorithm; (2)

the scores of candidates are assigned by semantic similarity between candidates and sentences in a document.

3.2.1 Word sense disambiguation

Our similarity measuring is based on the senses assigned by word sense disambiguation algorithm. In this paper, we adopt the state-of-the-art approach proposed by Luo et al. (2018) to assign senses to the words in a document [9].

3.2.2 Semantic ranking

Keyphrases can be directly or indirectly mentioned in many places in a document. And the semantic relatedness between candidates and sentences can provide effective assist for keyphrases extraction especially for short documents with less structural information. In order to obtain the semantic relationship between candidates and sentences, we calculate the similarity between candidates and sentences based on the senses assigned by word sense disambiguation algorithm.

A phrase mentioned in a sentence does not mean that every word in the sentence is similar to the words in the phrase, but at least one word is similar to the word in the phrase. Hence, our similarity between a phrase and a sentence is the maximum of the similarity between the words in the phrase and those in the sentence. The semantic score of candidate c_i is computed by:

$$S_{smtc}(c_i) = \sum_{s_j \subset S} \max_{u_k \in s_j, u_n \in c_i} [Sim(u_k, u_n)] \tag{4}$$

where S is a set of sentences, u_k and u_n are words in s_j and c_i, respectively, and Sim is the similarity calculation function in WordNet that is based on the shortest path that connects the senses. The computation of semantic score is illustrated in Algorithm 5.

3.3 Final score calculation

The above two steps produce two sets of score sequences for candidates. The final score is integrated by:

$$S_{final}(c_i) = Norm(S_{base}(c_i)) + Norm(S_{smtc}(c_i)) \tag{5}$$

where the weights of two scores are the same and Norm is the maximum and minimum normalization computed by:

$$Norm(Sc_i) = \frac{Sc_i}{max(Sc) - min(Sc)} \tag{6}$$

where Sc is the set of scores.

Algorithm 5: Semantic ranking

```
Input: set of candidates C, set of Sentences S
Output: Score S_smtc
Synsets ← WordSenseDisambiguation(C, S)
for c_i in C do
    S_smtc[c_i] ← 0
end
for c_i in C do
    for s_j in S do
        t ← 0
        for u_k in s_j do
            for u_n in c_i do
                t ← max(t, Similarity(Synsets[u_k], Synsets[u_n]))
            end
        end
        S_smtc[c_i] ← S_smtc[c_i] + t
    end
end
return S_smtc
```

4 Experiments and results

4.1 Datasets and evaluation measures

We carry out our experiments on three widely used datasets. The first dataset is Krapivin-2009 that has a high quality and consists of 2304 of scientific papers from computer science domain published by ACM [11]. Each paper has its keyphrases assigned by the authors and verified by the reviewers [11]. The second dataset is SemEval-2010 that consists of 244 scientific articles from the ACM Digital Library, and their keyphrases are carefully chosen by both authors and readers [12]. The last dataset is Hulth-2003 that consists of 2000 abstracts from the Inspec database, and each abstract has its keyphrases assigned by a professional indexer [13]. Some statistical information of these datasets is detailed in Table 1.

We evaluate the performance of our approach in terms of $F1$-score at the top N keyphrases. The $F1$-score is the harmonic average of the precision and recall and is computed as follows:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{7}$$

For the convenience of display, we multiply $F1$-score by 100.

Table 1 Average size of each article in the datasets

	Krapivin-2009	SemEval-2010	Hulth-2003
Articles	2304	244	2000
Sentences	781	345	6
Words	9198	7910	138
Candidate phrases	755	634	28
Keyphrases	5	15	10

Table 2 $F1$-scores computed at the top 5, 10 extracted keyphrases

	Krapivin-2009		SemEval-2010		Hulth-2003	
	$F1@5$	$F1@10$	$F1@5$	$F1@10$	$F1@5$	$F1@10$
TextRank	1.66	2.79	2.51	3.54	27.12	34.05
SingleRank	2.33	3.68	2.73	3.90	**29.92**	**35.71**
TopicRank	9.94	10.41	9.70	12.30	25.30	29.30
PositionRank	4.07	5.32	10.60	12.20	23.50	30.30
YAKE	9.85	10.58	10.12	13.08	13.77	15.73
MultipartiteRank	11.59	12.07	12.20	14.50	25.90	30.60
Proposed model	**12.19**	**12.41**	**12.69**	**16.63**	27.72	31.49

The bold value represents the maximum value for each column

4.2 Baselines and parameter settings

We compare the performance of our approach against that of six baselines. The first baseline is TextRank, which is the first graph-based keyphrases extraction method [3]. The second baseline is SingleRank, which is an improvement of TextRank by adding weighted edges between words that co-occur in a window of variable size $w > 2$ [14]. The third baseline is TopicRank which relies on a topical representation of the document [7]. The fourth baseline is PositionRank which incorporates information from all positions of a word's occurrences into a biased PageRank algorithm [1]. Unlike other baselines, the fifth baseline named YAKE is a feature-based method which uses statistical features such as word frequency to directly calculate every words' scores [15]. The last baseline is an approach which encodes topical information within a multipartite graph structure [5].

Our experiments are based on the pke toolkit [16]. The parameters are set to the values suggested by the authors.

4.3 Results

The results for the baselines and the proposed approach are detailed in Table 2. Overall, our approach achieves the best results in the datasets with long documents and achieves the competitive results in the datasets with short documents. MultipartiteRank obtains the best performance in the datasets with long documents among the baselines, which means that multipartite graph structure is effective in keyphrases extraction especially in long documents. Because TextRank and SingleRank construct graph based on words rather than phrases, they perform well in the datasets with short documents but poorly in long documents. The structural information of long documents is more complicated than that of short documents. The graph constructed by words cannot capture such complex information. This is why TextRank and SingleRank perform well in short documents but poorly in long documents. The results of our approach in the first two datasets are close to the results of MultipartiteRank. The reason is that our approach is based on MultipartiteRank. The first two datasets are composed of long documents in which MultipartiteRank performs much better than other baselines. The scores computed by semantic relationships are lower discrimination in long documents than that in short documents. It can also explain why our approach performs much better than MultipartiteRank in the third dataset, which is composed of short documents. In general, our approach achieves the best results in the datasets with long documents and achieves the competitive results in the datasets with short documents.

5 Conclusion

We introduced an unsupervised keyphrases extraction algorithm that combines semantic information and structural information, and demonstrated its effectiveness on three public datasets. In future work, we would like to apply more proper methods instead of clustering to break the document, especially the short document into topics.

Acknowledgements We thank Weiqiang Liu for his work in the revision of English expression during the manuscript's major revision and response during the minor revision.

References

1. 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, vol. 1, pp. 1105–1115 (2017)
2. Turney, P.D.: Learning algorithms for keyphrase extraction. Inf. Retr. **2**(4), 303–336 (2000)
3. Mihalcea, R., Tarau, P.: Textrank: bringing order into text. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (2004)
4. Hasan, K.S., Ng, V.: Automatic keyphrase extraction: a survey of the state of the art. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 1262–1273 (2014)
5. Boudin, F.: Unsupervised keyphrase extraction with multipartite graphs. In: Proceedings of the 2018 Conference of the North Amer-

ican Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 2, pp. 667–672 (2018)
6. Wan, X., Xiao, J.: Single document keyphrase extraction using neighborhood knowledge. AAAI **8**, 855–860 (2008)
7. Bougouin, A., Boudin, F., Daille, B.: Topicrank: graph-based topic ranking for keyphrase extraction. In: International Joint Conference on Natural Language Processing, pp. 543–551 (2013)
8. Raganato, A., Camacho-Collados, J., Navigli, R.: Word sense disambiguation: a unified evaluation framework and empirical comparison. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, vol. 1, pp. 99–110 (2017)
9. Luo, F., Liu, T., Xia, Q., Chang, B., Sui, Z.: Incorporating glosses into neural word sense disambiguation. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, pp. 2473–2482 (2018)
10. Miller, G.A.: WordNet: a lexical database for English. Commun. ACM **38**(11), 39–41 (1995)
11. Krapivin, M., Autaeu, A., Marchese, M.: Large dataset for keyphrases extraction. University of Trento, Trento (2009)
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, pp. 21–26 (2010)
13. Hulth, A.: 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, pp. 216–223 (2003)
14. Wan, X., Xiao, J.: CollabRank: towards a collaborative approach to single-document keyphrase extraction. In: Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pp. 969–976 (2008)
15. Campos, R., Mangaravite, V., Pasquali, A., Jorge, A.M., Nunes, C., Jatowt, A.: YAKE! Collection-independent automatic keyword extractor. In: European Conference on Information Retrieval, pp. 806–810 (2018)
16. Boudin, F.: pke: an open source python-based keyphrase extraction toolkit. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, pp. 69–73 (2016)