Chinese News Keyword Extraction Algorithm Based on TextRank and Topic Model

Ao Xiong and Qing Guo(✉)

Beijing University of Posts and Telecommunications, Beijing 100876, China
xiongao@bupt.edu.cn, guoqingbupt@163.com

Abstract. TextRank tends to choose frequent words as keywords of a document. In fact, some infrequent words can also be keywords. In order to improve this situation, a Chinese news keyword extraction algorithm LDA-TextRank based on TextRank and LDA topic model is proposed. The algorithm is a single document, unsupervised algorithm. It defines the diffusivity of two candidate words, constructs a new weight formula, and improves the weight of the edges in the text graph. At the same time, it combines with the LDA topic model, and the damping factor in TextRank is adjusted by calculating the word's topic relevance of the document. The experiment was carried out on the Chinese corpus. The results show that compared with TextRank, LDA-TextRank has an improvement in Precision, Recall and F1-measure.

Keywords: Keyword extraction · TextRank · LDA topic model

1 Introduction

Keywords are the summary of an article or a document. Keywords can be defined as a set of words or phrases that can summarize the topic of the article [1]. Keywords have important practical values in many fields, such as text classification and clustering, literature information retrieval, recommendation systems, etc. In most cases, however, the document does not provide keywords, so it is necessary to design an algorithm that automatically extracts keywords.

The algorithm of keyword extraction can be classified into the supervised and the unsupervised. As for the supervised algorithms, Witten et al. designed the KEA [2] system and proposed an algorithm for keyword extraction using Naïve Bayesian machine learning method. The algorithm only uses two features: word's TF-IDF (Term Frequency-Inverse Document Frequency) and its first occurrence in the document. The algorithm has to train a large number of labeled corpus to get the model. Turney et al. [1] designed a keyword extraction algorithm based on C4.5 decision tree and GenEx system based on the genetic algorithm to extract keywords. For unsupervised algorithms, Sparck [3] first proposed the concept of IDF (Inverse Document Frequency). Salton et al. [4] discussed the application of Term Frequency-Inverse Document Frequency (TF-IDF) in the field of information retrieval. Since then TF-IDF was regarded by scholars as a simple and basic algorithm for keyword extraction. TF-IDF treats the document as a Bag of Words (BOW) model, which means the order of words does not affect the results of the algorithm. In 2004, TextRank algorithm was proposed by

S. Han et al. (Eds.): AICON 2019, LNICST 286, pp. 334–341, 2019.
https://doi.org/10.1007/978-3-030-22968-9_29

Mihalcea et al. [5]. The algorithm originates from Google's PageRank algorithm for page ranking [6]. The words in the document are regarded as nodes, and the number of co-occurrences in the fixed-length window is used as the weight of the edges between the nodes, thus establishing the relation between the words. By constructing a text network to iterate, the score of each node is finally obtained, and the keywords are determined according to the scores.

At present, TextRank has been widely studied and various keyword extraction algorithms based on TextRank have been proposed. For example, Gu et al. [7] use only the TF of the candidate words to construct the weight between the nodes, and brings the factor of whether the candidate word is in the title to weight the final score. Since the IDF of the candidate word is not used, the computational complexity of the algorithm is reduced accordingly. Li et al. [8] mine the corpus of Wikipedia, calculate the TF-IDF of each term and convert them into vectors. Each element in the vector is its TF-IDF, and the cosine similarity of two vectors is used to represent the weight of the edge. The algorithm utilizes the information of Wikipedia, and improves the performance of TextRank in short documents. In addition, the first N words, whether the word is in the first sentence of a paragraph can also be used for keyword extraction [9]. Recent years, the LDA [10] topic model has also aroused attention in keyword extraction. For example, Liu [11] has carried out a systematic study on keyword extraction. The LDA topic model is combined with the TextRank algorithm. By calculating the scores of words in different topics, and finally weighting according to the topic weights, the scores of candidate words are obtained.

Based on TextRank, this paper introduces the concept of diffusivity, constructs a new formula for edge weights, and introduces the LDA topic model into TextRank to calculate the topic relevance of each word in the document, thereby changing the jump probability of the damping factor term and increasing the score of the word with high relevance of the topic. The experiment uses the open sourced jieba[1] package as the word segmentation tool, and compares the proposed algorithm with TextRank on the Chinese corpus.

The rest is organized as follows. In Sect. 2, the proposed algorithm will be described in detail. In Sect. 3, experiments and results will be shown. Section 4 will conclude the work.

2 Algorithm Description

2.1 Diffusivity Between Two Candidate Words

For a Chinese document, we have first to segment it into words using jieba, which an open sourced word segmentation tool for Chinese. After that, we need to filter out unwanted words based on part of speech (POS) and stop words. Finally, we only keep verbs, nouns and non-stop words as candidate words, each of which could probably be keywords. Additionally, we need also segment the document into sentences according to punctuations like full stop, ellipsis, exclamation mark, etc.

[1] https://pypi.org/project/jieba/.

Some statistics need to be done as followings.

- For each candidate word W_i, count the number of sentences containing W_i as N_i;
- For each pair of candidate words (W_i, W_j), count the number of sentences that contain both W_i and W_j as N_{ij}.

The definition of the diffusion of two candidate words is as follows.

$$u_{ij} = \frac{N_i + N_j - 2N_{ij} + 0.5}{N + 0.5} \tag{1}$$

where N represents the total number of sentences of the text.

The diffusivity of two words indicates how dispersed the two words are in the article. 0.5 here is a smoothing factor, which avoids that u_{ij} could be zero. From its definition, two following conclusions can be got.

- $u_{ij} = u_{ji}$.
- If words W_i and W_j always appear in the same sentences, which results in $N_i = N_j = N_{ij}$, thus u_{ij} would be close to zero.

2.2 Relation Between Two Candidate Words

Compared with TextRank, the proposed algorithm calculates the relation between two candidate words in a little more complicated way. First, as in TextRank, we need to define a co-occurrence window length $l(l \geq 2)$ and count the times of every pair of words W_i and W_j where they co-occur within the window length l, denoted as c_{ij}. Then we define the relation between words W_i and W_j as the following

$$w_{ij} = \mathrm{c}_{ij} \cdot u_{ij} \tag{2}$$

Equation (2) shows that the relation between two candidate words is a balance of co-occurrence time and diffusivity.

Considering that the non-candidate words can provide the distance information to judge the relation between the two words, the words in the co-occurrence would also contains non-candidate words [11].

Then a graph would be built where all the candidate words are set as nodes in the graph and the relation between two candidate words are set as the edge weight. It should be noted that the graph is non-directed.

2.3 Bringing in LDA Topic Model

LDA (Latent Dirichlet Allocation) model was first proposed by Blei et al. in 2003 in order to build a document topic model [10]. The model is a generation model in which for each document d, it can be represented as a Multinomial distribution of K topics. At the same time, the words in the topic and vocabulary also satisfy a Multinomial distribution, which is the Dirichlet prior distribution with hyperparameters α or β. Therefore, for a document d, it can be regarded as extracting a topic from the topic

distribution θ, and extracting a word from the word distribution φ corresponding to the topic, repeating N times, that is, generating an article containing N words. The joint distribution can be obtained by Eq. (3).

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \tag{3}$$

The LDA model can be trained by Gibbs Sampling algorithm [12]. Then parameter θ and φ would be estimated. Supposing there are K topics, we could get the estimated probability of word w in topic k, $p(w|k)$ via φ, and estimated probability of topic k in document d, $p(k|d)$ via θ. Hence the relevance between word w and document d could be calculated by Eq. (4).

$$p(w|d) = \sum_{k=1}^{K} p(w|k) p(k|d) \tag{4}$$

The equation of TextRank model can be defined as shown below

$$S(W_i) = (1-p) + p \cdot \sum_{W_j \in In(W_i)} \frac{c_{ji}}{\sum_{W_k \in Out(W_j)} c_{jk}} S(W_j) \tag{5}$$

where p is a damping factor, which guarantees that the algorithm can reach convergence. It means each node in the graph can be reached by the probability of p through other nodes connected to it and by the probability of 1-p through any other nodes in the graph. $In(W_i)$ represents the set of nodes that point to W_i and $Out(W_i)$ represents the set of nodes pointed to by W_i. Since the graph is an undirected graph, $In(W_i)$ and $Out(W_i)$ means the same set.

In the proposed algorithm, we replace c_{ij} in Eq. (5) with w_{ij} in Eq. (2). After experiments, we found that the following equation would get better results.

$$S(W_i) = (1-d) \cdot \exp(p(W_i|d)) + d \cdot \sum_{W_j \in In(W_i)} \frac{w_{ji}}{\sum_{W_k \in Out(W_j)} w_{jk}} S(W_j) \tag{6}$$

In Eq. (6), the damping factor varies according to different words. Every node in the graph is required to give an initial score which is often a small value between 0 and 1 and then its score would be calculated iteratively according to Eq. (6). The process would stop until the maximum number of iterations is reached or the scores reach to a convergence. Particularly, some candidate words in the document may not be in the LDA topic model. Here we use the average $p(W_i|d)$ of other candidate words in that document to replace it.

2.4 Getting Keywords

First, we need to train the LDA topic model using the corpus. For every word W_i in document d, we have to calculate $p(W_i|d)$ in Eq. (4) and for every pair of word W_i and W_j we have to calculate w_{ij} in Eq. (2). Then we initialize all the nodes' scores to a value close to zero and calculate the score of each node in the graph iteratively according to Eq. (6). When the iteration repeats 100 times or the scores reach to a convergence, the iteration stops. Finally, we rank the words by their scores and pick the words with highest scores as keywords.

3 Experiment and Results

3.1 Corpus and Evaluation

The corpus in the experiment is the news articles released in September 2017 from ***South Daily***. Every news article would be given several keywords by the editor. We take these words as reference keywords. 500 articles with no less than 5 reference keywords are randomly chosen as the test corpus.

Precision (P), Recall (R) and F1-measure ($F1$) is used to evaluate the experiment [13]. Their definitions are as follows

$$P = \frac{|A \cap B|}{|A|}, R = \frac{|A \cap B|}{|B|}, F1 = \frac{2PR}{P+R} \tag{7}$$

where A is the set of keywords extracted by the algorithm, B is the set of the reference keywords and $|A|$ is the number of elements in A.

3.2 Results

We name the proposed algorithm LDA-TextRank. After experiments, we found that in different co-occurrence window length l and damping factor p, LDA-TextRank outperforms in Precision, Recall and F1-measure compared to TextRank. When $l = 10$, $d = 0.5$ and topic number $K = 50$, they both perform their best. In this condition, experiments are conducted when number of keywords varies from 1 to 15.

Figure 1 shows that when the number of extracted keywords is small, the Precision curve of LDA-TextRank is above that of TextRank. Figures 2 and 3 show that when the keywords number is small, the Recall curves and F1-measure curves of two algorithm almost overlap. However, when the keywords number increases, LDA-TextRank performs better than TextRank. Particularly, F1-measure reaches the peak at the number of 5. This is because most documents in the corpus have 5 reference keywords. Figure 4 shows that the TextRank's curve is inside the LDA-TextRank's curve, which means at the same Precision (or Recall), LDA-TextRank's Recall (or Precision) is higher than TextRank's. In conclusion, the results show that LDA-TextRank outperforms TextRank.

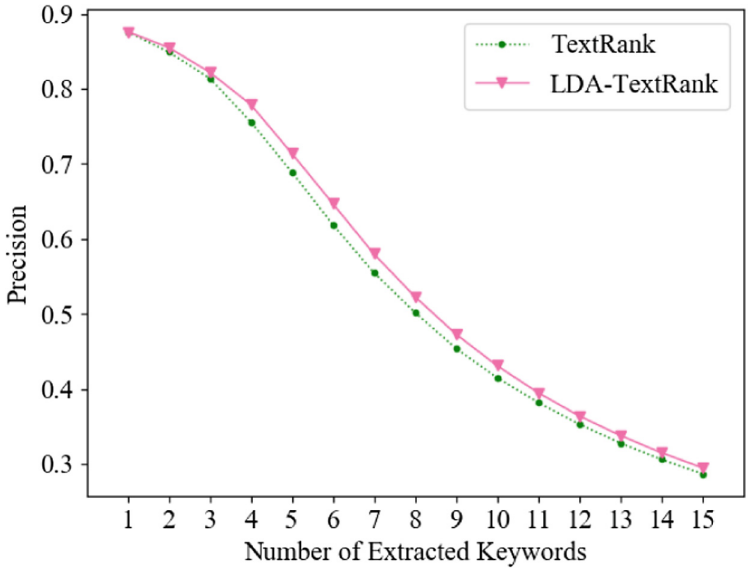

Fig. 1. Precision curves when the number of extracted keywords varies from 1 to 15

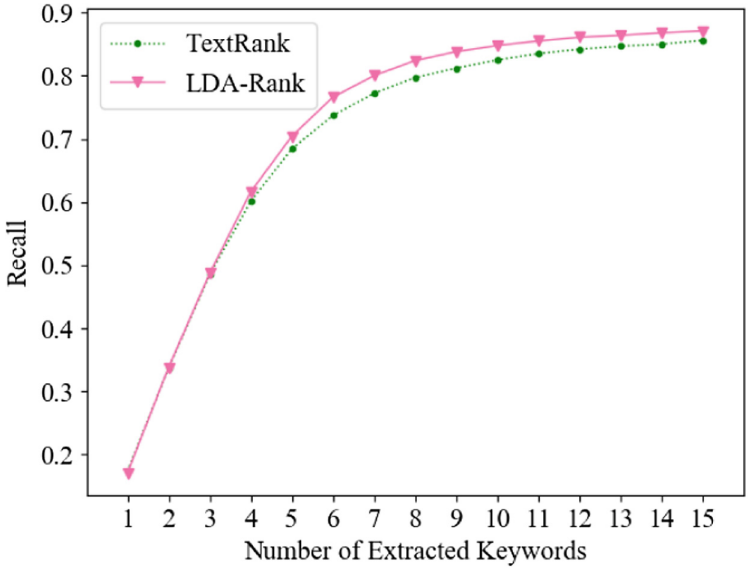

Fig. 2. Recall curves when the number of extracted keywords varies from 1 to 15

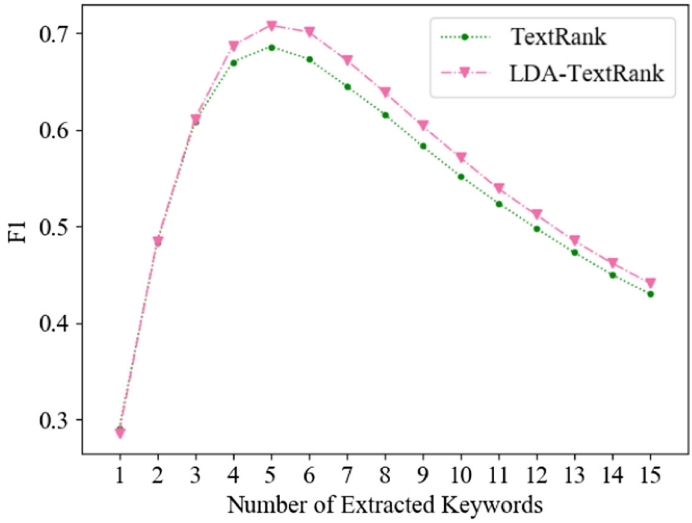

Fig. 3. F1 curves when the number of extracted keywords varies from 1 to 15

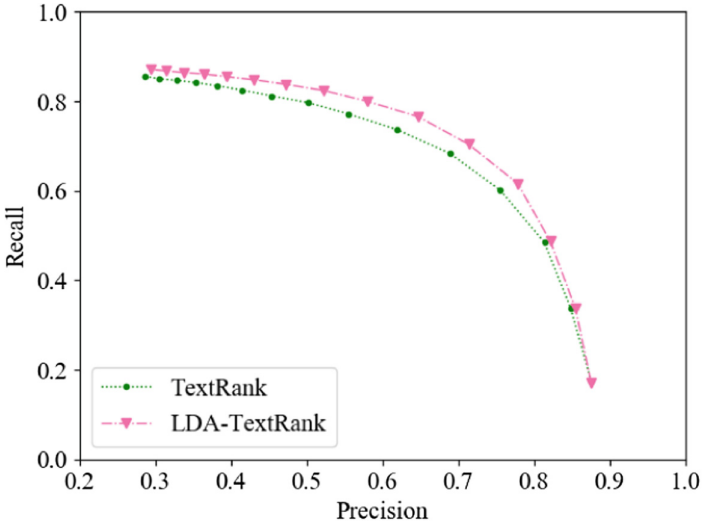

Fig. 4. Precision-Recall curves when the number of extracted keywords varies from 1 to 15

4 Conclusion

The paper optimizes the TextRank algorithm by bringing the concept of diffusivity and integrate LDA topic model in the proposed algorithm. Hence the algorithm actually extracts keywords in the level of the whole text instead of only within a co-occurrence window and integrating the topic model would allow the algorithm to catch some

important topic words in a document properly. Results show that the proposed algorithm outperforms TextRank in Precision, Recall and F1-measure.

References

1. Turney, P.D.: Learning algorithms for keyphrase extraction. Inf. Retrieval **2**(4), 303–336 (2000)
2. Frank, E., Paynter, G.W., Witten, I.H., et al: Domain-specific keyphrase extraction. In: 16th International Joint Conference on Artificial Intelligence (IJCAI 99), pp. 668–673. Morgan Kaufmann Publishers Inc., San Francisco (1999)
3. Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. J. Documentation **28**(1), 11–21 (1972)
4. Wu, H., Salton, G.: A comparison of search term weighting: term relevance vs. inverse document frequency. In: Proceedings of the 4th Annual International ACM SIGIR Conference on Information Storage and Retrieval, pp. 30–39. ACM Press, New York (1981)
5. Mihalcea, R., Tarau, P.: TextRank: bringing order into text. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pp. 404–441. ACL, Stroudsburg (2004)
6. Wu, X., Kumar, V., Quinlan, J.R., et al.: Top 10 algorithms in data mining. Knowl. Inf. Syst. **14**(1), 1–37 (2008)
7. Gu, Y.R., Xu, M.X.: Keyword extraction from News articles based on PageRank algorithm. J. Univ. Electron. Sci. Technol. China **46**(5), 777–783 (2017)
8. Li, W., Zhao, J.: TextRank algorithm by exploiting Wikipedia for short text keywords extraction. In: 2016 3rd International Conference on Information Science and Control Engineering (ICISCE), pp. 683–686. IEEE, Piscataway (2016)
9. Siddiqi, S., Sharan, A.: Keyword and keyphrase extraction techniques: a literature review. Int. J. Comput. Appl. **109**(2), 18–23 (2015)
10. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**(Jan), 993–1022 (2003)
11. Liu, Z.Y.: Research on Keyword Extraction Using Document Topical Structure. Tsinghua University, Beijing (2011)
12. Casella, G., George, E.I.: Explaining the Gibbs sampler. Am. Stat. **46**(3), 167–174 (1992)
13. Powers, D.M.: Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. J. Mach. Learn. Technol. **2**(1), 37–63 (2011)