Key word extraction for short text via word2vec, doc2vec, and textrank

JUN LI

GUIMIN HUANG

CHUNLI FAN

ZHENGLIN SUN

HONGTAO ZHU

Key word extraction for short text via word2vec, doc2vec, and textrank

Jun LI[*], **Guimin HUANG, Chunli FAN, Zhenglin SUN, Hongtao ZHU**
School of Information and Communication, Guilin University of Electronic Technology, Guilin, P.R. China

Abstract: The rapid development of social media encourages people to share their opinions and feelings on the Internet. Every day, a large number of short text comments are generated through Twitter, microblogging, WeChat, etc., and there is high commercial and social value in extracting useful information from these short texts. At present, most studies have focused on extracting text key words. For example, the LDA topic model has good performance with long texts, but it loses effectiveness with short texts because of the noise and sparsity problems. In this paper, we attempt to use Word2Vec and Doc2Vec to improve short-text key word extraction. We first added the method of the collaborative training of word vectors and paragraph vectors and then used the TextRank model's clustering nodes. We adjusted the weights of the key words that were generated by computing the jump probability between nodes and then obtained the node-weighted score, and eventually sorted the generated key words. The experimental results show that the improved method has good performance on the datasets.

Key words: Key word extraction, short text, word2vec, doc2vec, textrank

1. Introduction

Currently, many people like to share what they see, hear, think, and feel on social media such as microblogs, WeChat, Twitter, and Facebook. The rapid development of social media has led to an explosion of the amount of text generated every day. The content of these texts include product reviews, expressed opinions, life sharing, etc. Although some of this content is meaningless, there is still a very large amount of data that has great commercial and social value. In particular, many research institutions have conducted research on social media data in recent years. The main feature of social media data is that the length of the text is shorter; therefore, compared with traditional long text, the biggest challenge for short text processing is the sparsity and noise of the textual semantics.

Key word extraction is a basic and important task in text processing. TF-IDF (term frequency-inverse document frequency) is a classic method for key word extraction, and the algorithm is simple and very efficient. However, TF-IDF only mines information according to word frequency and inverse document frequency and cannot reflect the deep semantic information of the text. The LDA topic model performs very well in dealing with long text problems, but the key words that it extracts are generally too broad to reflect the topic of the article, and it performs poorly on short texts. The TextRank [1] algorithm for key word extraction comes from the graph model and the iterative algorithm is built through networks and random walk, which is advantageous in model training. In this paper, we studied the issue of key word extraction from social media short texts via the improved TextRank. TextRank originates from PageRank [2], which is a way to measure the importance of

─────────────────
[*]Correspondence: lijun5221@126.com

website pages. It was originally used for Google's website page-sorting tasks from their search engine results. The algorithm also has excellent performance in key word ranking. Therefore, in our task, we hope to optimize the key word extraction problem by combining the word vector and TextRank algorithm.

Any text is made up of multiple words, and the word vector is a form of the words that computers can handle. Word2Vec is a way to train words into vectors. Through a large-scale corpus, this neural network model can capture the semantic relationships of words in their contexts. To obtain the semantic information between words, we use the Word2Vec [3, 4] training word vector. However, due to the semantic sparseness of short text, we use Doc2Vec [5] to train the paragraph vectors and improve the accuracy of key word extraction by using coordinated word vectors and paragraph vectors. Word2Vec and Doc2Vec map words and paragraphs, respectively, to low-dimensional dense spaces, and in large-scale training, they retain the correlation between words and paragraphs. On the one hand, this vector-formed word facilitates our mathematical processing and, on the other hand, due to the existence of semantic relevance, we can further carry out other related tasks with the text. Then, we use the TextRank model's clustering nodes and adjust the weights of the key words generated by computing the jump probability between nodes. Then, we obtain the node-weighted score, and eventually sort the generated key words.

In this paper, we first train the word vector and sentence vector through Word2Vec and Doc2Vec, respectively. Then, according to the similarity between the word vectors, we classify a document into several subclusters. We calculate the Euclidean distance of any word from the centroid of a subcluster. This distance reflects the difference between the word and the others near the centroid and is called the weighted value. The higher the weighted value is, the higher the jump probability between the adjacent nodes. In this way, we represent the semantic relationship between two nodes in the word vector space as the clustering weight influence between nodes, and we can obtain the transition probability matrix from the jump probability. Then, we sort the iterative calculation results, select the Top N nodes as the key word extraction results, and implement key word extraction.

The rest of this article is organized as follows. Section 2 reviews the related work on key word extraction. Section 3 discusses our approach. The experimental details and results analysis are in Section 4. Finally, the conclusions and future work are summarized in Section 5.

2. Related work

Key word extraction is an important technique for text mining, web retrieval, text classification, and so on. With the rapid development of the mobile Internet and the accelerated evolution of the Internet, the explosive growth of text data requires more advanced key word extraction techniques. The early key word extraction research focused on statistics, and the key words were sorted by calculating the cooccurrence frequency of words. Matsuo [6] presented a key word extraction algorithm that uses the word cooccurrence statistical information from a single document; the main advantages of this method are that it is simple and does not require a corpus. Ercan [7] proposed a key word extraction method using lexical chains where the lexical chains are composed of head nouns, which are from the representations of key phrases in the document. In recent years, text mining tasks on social media have become increasingly important. The task of extracting key words based on short texts is challenging because the text generated by social media is short and semantically sparse. Willyan [8] proposed the TKG approach, which extracts key words from a Twitter message. These short text messages from tweets are first expressed as a graph model and then analyzed and mined by using related methods from graph theory and link analysis. Yang [9] proposed a new approach, the topic-based TextRank, which takes the lexical meaning of the text unit into account. In the key word extraction task from short texts, in order to

eliminate the negative effects of short text semantic sparseness, many methods use clustering ideas to splice short texts into pseudo long texts. Sina Weibo is a Chinese microblogging blog where each microblog has no more than 140 characters, similar to Twitter, which is also typical short-text data. It has been widely studied because of its large amount of data and rich information. Xu [10] used Sina Weibo data as the research object for key word extraction tasks, implemented a method that combines the merits of LDA, Entropy and TextRank, and considered the influence of the topic discrimination of the extracted key word. In microblogging environments where research on topic models of short texts has gradually become a hot topic and a difficult task [11], researchers explore topic models for short messages. Önal [12] proposed an LSA-based key word extraction method that can identify semantic information in topic words or key words using LSA, and the results perform well in large document sets. The development of machine learning [13] and neural networks brings new methods and technologies to key word extraction. Deep learning methods such as the long short-term memory (LSTM) and recurrent neural networks (RNN) have been widely used in text-mining tasks in recent years. A new approach of automatic key word extraction was proposed, and researchers used LSTM neural networks, RNN [14] and the automatic extraction of text key words via bidirectional networking. Ding [15] proposed a method called BEIS, which is based on byte entropy iterative segmentation. At the beginning of the method, the text is divided into several parts, and these parts are gathered using coarse clustering. Then, the first key word is extracted from each subcluster. Since the Word2Vec trained word vectors can retain the semantic information, key word extraction based on Word2Vec is feasible. Xu proposed a Chinese text summarization algorithm based on Word2Vec [16], which is a graph-based ranking model. Ying used the TextRank algorithm to filter the spam in hot topics in microblogs [17]. Key word extraction using collective node weight (KECNW) [18] is an unsupervised graph-based method that independently weights nodes using multiple parameters. Georgios [19] proposed an unsupervised extractive summarization algorithm that can examine whether there is any potential overlap between the extractive summarization and argument mining.

The weights of edges are not taken into account in the original word graph of TextRank. Therefore, on the basis of word graph weighting, how to incorporate the external information of the document into the calculation process of TextRank is the key for key word extraction. Existing methods such as topic weighting and inverse document frequency weighting require preprocessing the dataset, and the results vary greatly depending on the dataset. The training data of Word2Vec is independent of the document to be extracted, and the TextRank is improved by using the word vector generated by the training and the text sentence vector. In theory, a more stable extraction result can be obtained.

3. Methodology

The key word extraction method based on TextRank uses the essentiality ranking of words in documents. In this paper, we construct the candidate key word graph to represent the structural relationships between them, and we use Word2Vec and Doc2Vec to capture the semantic information between words. Then, the word importance clustering is determined using the spatial position relationship of words in clusters, and the clustering weighting of TextRank is realized. We also construct the probability transfer matrix between words. Finally, the importance of nodes is acquired through iterative operations to realize key word sorting and extraction. The following discussion details this method.

3.1. Word graph construction

The TextRank-based key word extraction method extracts the adjacency relations of internal words to form a word graph and then calculates the importance of words according to their structural features in the word graph.

To build the candidate key word graph, we divide the text into sentences, segment words and part-of-speech tagging; retain the nouns, verbs and adjectives; and form a node set of word graph V. Then, the adjacency relationship between all words constitutes the edge set of the word graph E, thereby forming a candidate key word graph $G = (V, E)$. When constructing the edge, if the word a is adjacent to the word b, then two directed edges $a \rightarrow b$ and $b \rightarrow a$ are added to the word graph. That is, the word graph G is a directed graph, as shown in Figure 1.
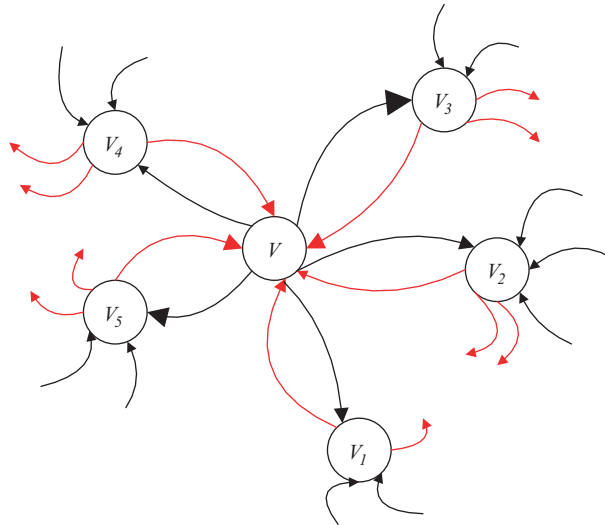

Figure 1. Candidate key word graph.

Given the graph $G = (V, E)$ above, $t(u)$ denotes the TextRank value of the node u. Then, $t(u)$ can be calculated by Eq. (1).

$$t(u) = d \sum_{d \in adj[u]} p(v \rightarrow u)t(v) + (1 - d) * \frac{1}{V},$$ (1)

where d is a damping factor that can be set between 0 and 1, and it is usually set to 0.85 [2]. $adj[u] = \{v | (v \rightarrow u) \in E\}$ denotes the set of adjacent nodes of node u, and $p(v \rightarrow u)$ represents the random jump probability from node v to u.

The traditional TextRank algorithm uses a uniform jump strategy between adjacent nodes, and the jump probability between nodes $p(v \rightarrow v_i)$ can be calculated by Eq. (2).

$$p(v \rightarrow v_i) = \begin{cases} \frac{1}{\deg(v)}, if \exists (v \rightarrow v_i) \in E, \\ 0, Otherwise \end{cases}$$ (2)

where $\deg(v)$ is the degree of the node v. As in Figure 1, the probability of the node v jumping to any neighboring node is $p(v \rightarrow v_i | i \in [1, \cdots, 5]) = 0.2$. This is a uniform optimization strategy. In fact, the degrees of nodes are not the same according to the influence of each node. To optimize the assignment of the jump probability using the external information of words and sentences, we believe that the use of word and sentence vectors to improve TextRank, in theory, can obtain more stable results.

TextRank was derived from the improved PageRank algorithm that was used by Google, and it can sort the importance of web pages with search engine results. It calculates the PageRank value of a web page using

Eq. (3):

$$PR(P_i) = (1 - d) + d * \sum_{j \in In(P_i)} \frac{1}{|Out(P_j)|} PR(P_j). \tag{3}$$

Obviously, this is an iterative formula. PR is the abbreviation of PageRank, which indicates the PageRank value of a web page, P_i represents a web page, and P_j represents a web page connected to P_i.

The TextRank algorithm is a graph-based ranking model, which is essentially a way of deciding the importance of a vertex within a graph based on the global information recursively drawn from the entire graph [1]. Eq. (4) gives the algorithm:

$$WS(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j). \tag{4}$$

In comparison with Eq. (3) of PageRank, Eq. (4) adds a weight item w_{ji}, which represents the different degree of importance between the two vertices, V_i and V_j. We will describe the details of the weight update in the next section. We sort the importance of each node weight value, and then obtain the candidate key words.

3.2. Word2vec and doc2vec cluster weighting

Mikolov [3, 4] shared the word vector training tool Word2Vec in 2013, which uses neural network models to automatically learn the appearance of words in the corpus and map them to a low-dimensional dense space. That is, $word \rightarrow R^n$, where n represents the dimensionality, which is usually set to 50–300. The word vector generated by Word2Vec overcomes the sparseness problem in traditional text representations and contains semantic information between words. Therefore, we can use this semantic information to weight the transfer probability between word graph nodes from TextRank. Zhao and Wen use Word2Vec and TextRank to solve the problem of key word extraction[20, 21].

We think that a document can be clustered into several sub-clusters according to the similarity between word vectors. The farther a word is from the centroid of the subcluster, the more it can reflect the different information of the subcluster that is different from the words near the centroid. Therefore, we cluster these words by calculating the similarity between any two words using Eq. (5) below.

$$Similarity(W_i, W_j) = cos\theta = \frac{W_i, W_j}{\|W_i\| \cdot \|W_j\|}, \tag{5}$$

where W_i, W_j respectively represent the two words, $Similarity(W_i, W_j)$ indicates the similarity between them and contains semantic relevance.

The word vector reflects the correlation between words, but it is unordered and therefore contains almost no semantics of the sentence, especially on short texts. In this paper, we add the Doc2Vec-trained paragraph vector to the TextRank model. The sentence vector retains the correlation between the texts, thus improving the accuracy of key word extraction in texts. Le [5] proposed this method of representing sentences and documents, called Doc2Vec, they acquired the labels of the sentences through joint training, and then perform the sentence classification task. However, in our method, we only want to train the sentence vector that contains the candidate key word, and then combine word vectors and sentence vectors.

Given document $d = \{s_1, s_2, \cdots, s_m\}$, s_m represents m sentences, and the candidate key word set is $k = \{w_1, w_2, \cdots, w_n\}$. At the beginning of the experiment, we use Word2Vec to train the word vector, and $\vec{w_i}$

denotes word vector of the word w_i. Any sentence s_i can be represented by words as $s_i = \{w_{i1}, w_{i2}, \cdots, w_{in}\}$. If the key word w_i appears in the sentence s_i, we replace the key word vector $\vec{w_i}$ with the sentence vector $\vec{s_i}$, and $\vec{s_i}$ is obtained by weighting the word vectors in the sentence. The word vectors of documents are clustered using K-means clustering and the results are expressed as $C = \{C_1, C_2, \cdots, C_k\}$. We randomly select the value of K from the range of 2 to 10. After comparing the results, we choose K = 5 in the clustering of short text corpus in this experiment. Eq. (6) can calculate the degree of importance of the weight value of any word w_i in subcluster C_j, where $C_j \in C$.

$$Weight(w_i) = \frac{d(\vec{w_i}, \vec{c_j})}{\sum_{v \in C_j} d(\vec{v}, \vec{c_j})} \times |C_j|, \tag{6}$$

where $\vec{c_j}$ corresponds to the vector of the centroid of the subcluster C_j, $d(\vec{v}, \vec{c_j})$ denotes the Euclidean distance between vectors $\vec{v}$ and $\vec{c_j}$ in vector space, $d(\vec{w_i}, \vec{c_j})$ denotes the Euclidean distance between the word vector $\vec{w_i}$ and the centroid vector $\vec{c_j}$, and $|C_j|$ represents the number of words in subcluster C_j.

After performing the cluster analysis and calculating the importance of voting for each word, we can calculate the transition probability between nodes through the results from Eq. (6) by using Eq. (7) as follows:

$$p(i \rightarrow j) = \frac{Weight(j)}{\sum_{w \in adj[i]} Weight(w)}. \tag{7}$$

3.3. Transfer matrix calculation and key word extraction

According to the theory of graph analysis [19], given the transition probability of the graph model, the importance of the node can be calculated through iterative calculations. Therefore, the transfer matrix M is given in Eq. (8).

$$M = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \cdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix}, \tag{8}$$

where p_{ij} represents the transition probability from node i to node j. That is, $p_{ij} = p(i \rightarrow j)$, and column j in M represents the probability distribution of jumping from the node j to the other nodes, and the sum of the jump probabilities of each column is 1. Then, we use the transfer matrix M as the weight update Eq. (4), and the new iteration Eq. (9) is as follows:

$$WS(V_i) = (1 - d) * \frac{e}{n} + d * M * WS(V_j). \tag{9}$$

Here, e is a vector where all components have the value 1 and dimension n. When two adjacent calculative results are similar, the iteration stops. Then, the current weights of all vocabulary nodes are sorted.

The main steps of the algorithm are as follows.

1) Divide the given test T into complete sentences, i.e. $T = \{S_1, S_2, \cdots, S_m\}$, where T is divided into m independent sentences.

2) For each sentence $S_i \in T$, first filter stops words operation, and then performs the part of speech (POS) tagging. Only reserve words with specified parts of speech, such as nouns, verbs, adjectives, so that

$S_i = \{w_{i,1}, w_{i,2}, \cdots, w_{i,n}\}$, where $w_{i,j}$ are reserved key words, and n is the number of key words. Then, train the sentence vector when the candidate key word appears in the sentence.

3) Construct a candidate key words graph $G = (V, E)$, where V is a set of nodes that consists of the candidate key words generated by step (2), and then use the cooccurrence relationship to construct an edge between any two points. Two edges exist between nodes only if their corresponding vocabulary cooccurs in a window of length K, where K represents the size of the window. That is, the number of the maximum cooccurrence words is K.

4) Calculate the weight of each node by iterating.

5) Sort the importance of each node weight value, and then obtain the candidate key words.

The procedure is described in Algorithm 1.

Algorithm 1 Pseudocode of key word extraction.

Require: Texts Collection;
Ensure: Key words;
 1: Segment T into sentences $S_1, S_2, \ldots, S_m \leftarrow$ Short Texts set T;
 2: Word vector $\leftarrow$ words set preprocessed;
 3: Document vector $\leftarrow$ If key word W_i appear in sentence S_i;
 4: Cosine similarity array $\leftarrow$ word vector;
 5: For i in range (iteration), set the number of iterations as 100: $WS(V_i) = (1 - d) * u + d * M * WS(V_j), d = 0.85, (formula(9))$;
 6: Sort $WS(V_i)$;
 7: Choose top N key words after sorting;
 8: end for;
 9: end.

---

4. Experiment

4.1. Dataset

We use the Wikipedia Chinese Corpus[1] (which can be downloaded at the following link) to train the Word2Vec word vector. The Wikipedia Chinese Corpus contains long documents, and each document has a title that describes this document. We use Sina Weibo as short text data[2], The dataset contains 60,000 short messages, which is a small dataset similar to Twitter. In our experiment, the Sina Weibo dataset is divided into training set and testing set, with a ratio of 8 to 2. In the preprocessing, we remove the stop words, clean up the special symbols and delete the duplicates. We use the Chinese segment tool Jieba (which can be installed directly in Python) to process it. In the training, we use the Word2Vec and Doc2Vec modules of Gensim, and set some parameters as shown in Table 1.

The test dataset[3] was from the data used by Xia [22]. It includes 1524 articles from the Southern Weekend website, and it extracts the title and the body of the articles. The average number of characters and key words in each document in the dataset is 2629.101 and 3565, respectively.

For comparing the performance of our method with those of other works, we also added another Twitter dataset[4], which contained 30,000 Twitter short messages, and we used the Wikipedia corpus to train the word

---

[1] http://download.wikipedia.com/zhwiki/latest/zhwiki-latest-pages-articles.xml.bz2
[2] https://archive.ics.uci.edu/ml/machine-learning-databases/00323/
[3] https://github.com/iamxiatian/x-extractor/blob/master/data/articles.xml
[4] https://tianchi.aliyun.com/datalab/index.htm?spm=5176.100065.5490697.4.562b5eddWu2xQy.

Table 1. Some parameters in word2vec and doc2vec.

Parameters	Explanations	Values	Parameters	Explanations	Values
size	Dimension of word vectors	200	size	Dimension of feature vectors	200
sg	Training model: 0:CBOW model 1:Skip-gram model	1	dm	Training algorithm: 1:DM algorithm 0:DBOW algorithm	1
hs	Training method: 0:Negative Sampling method 1:Hierarchical Softmax method	1	hs	Training method: 0:Negative Sampling method 1:Hierarchical Softmax method	0
min_count	Size of minimum word frequency	5	min_count	Size of minimum word frequency	2
Others		Defaults	Others		Defaults

vectors.

4.2. Experimental results and analysis

To facilitate the analysis of the results, we use the accuracy rate P, the recall rate R, and the $F-measure$ as evaluation criteria. The following formulas given in Eqs. (10)–(12) are their respective calculation formulas.

$$P = \frac{The number of keywords extracted correctly}{The number of all keywords extracted}. \tag{10}$$

$$R = \frac{The number of keywords extracted correctly}{The number of standard keywords}. \tag{11}$$

$$F - measure = \frac{2 * P * R}{P + R}. \tag{12}$$

In this paper, we compare the traditional TF-IDF key word extraction algorithm and the LDA model with our TextRank algorithm via Word-Doc2Vec (Word-Doc2Vec TextRank). We also compare the W-TextRank [20] and PW-TextRank [22] algorithms, the W-TextRank algorithm combined with the Word2Vec and TextRank, and the PW-TextRank algorithm that added positional information for the words in the training. We set the number of key words that are extracted as 3, 5, 7, and 10, and then test the long document dataset from Southern Weekend News and the short text dataset from Sina Weibo. The results of the experiments on these two sets are given in Tables 2 and 3, respectively, where P, R, and F represent the accuracy rate, recall rate and F-measure, respectively. Table 4 shows the experiment results of the Twitter data.

Table 2. The experiment results of Southern Weekend (long texts).

	N = 3			N = 5			N = 7			N = 10		
	P	R	F	P	R	F	P	R	F	P	R	F
TF-IDF	32.4%	36.8%	34.5%	31.3%	32.8%	32.0%	30.3%	36.4%	33.1%	28.0%	42.3%	33.7%
LDA	40.2%	46.3%	43.0%	38.6%	46.2%	42.1%	36.6%	52.1%	43.0%	34.0%	48.3%	39.9%
Word-Doc2Vec TextRank	36.6%	45.4%	40.5%	36.9%	46.7%	41.2%	37.1%	48.2%	41.9%	35.2%	50.3%	41.4%

Tables 2 and 3 respectively show the results of the tests on the Southern Weekend and Sina Weibo data. As seen from Table 2, the traditional LDA model performs better than TF-IDF on long texts, and our model is not the best. We believe that there are two main reasons for this. One is the size of the training corpus, and the

Table 3. The experiment results of Sina Weibo (short texts).

	N = 3			N = 5			N = 7			N = 10		
	P	R	F	P	R	F	P	R	F	P	R	F
TF-IDF	30.2%	33.3%	31.7%	27.9%	41.5%	33.4%	25.2%	35.7%	29.5%	22.6%	36.8%	28.0%
LDA	31.4%	28.3%	29.8%	30.5%	42.0%	35.3%	29.8%	34.5%	32.0%	26.1%	35.8%	30.2%
Word-Doc2Vec TextRank	38.4%	47.5%	42.5%	36.9%	48.6%	41.9%	35.3%	55.4%	43.1%	33.2%	50.9%	40.2%

Table 4. The experiment results of Twitter (short texts).

	N=3			N=5			N=7			N=10		
	P	R	F	P	R	F	P	R	F	P	R	F
TF-IDF	33.6%	35.8%	34.7%	28.6%	38.9%	33.0%	25.3%	37.2%	30.1%	23.4%	35.7%	28.3%
LDA	37.6%	30.4%	33.6%	35.1%	40.2%	37.5%	33.3%	36.8%	35.0%	29.6%	31.2%	30.4%
Word-Doc2Vec TextRank	46.7%	42.5%	44.5%	41.5%	46.2%	43.7%	37.6%	50.4%	43.1%	36.6%	48.9%	41.9%

other is that the sentence vectors are weighted using simple word vectors to extract sentence semantics, but the expression of key words is insufficient, especially on long texts. From Table 3, the performance of the F-measure in the LDA model on short texts deteriorates from N = 5 to N = 7, but our model is always the best in short-text performance. We believe that traditional models do not perform as well as models that rely on large-scale training on short-text key word extraction problems with fewer key words. Tables 3 and 4 respectively show the performances using the Sina Weibo and Twitter data, and our method has good performance when comparing all three algorithms. We can also see that the algorithm performs better in English text data than in Chinese, and we think the reason might be the use of the Wikipedia corpus to train the model.

We also compared other TextRank-based key word extraction algorithms [23, 24]. Xia [25] proposed a word position weighting TextRank (called PW-TextRank) key word extraction, and Zhao [20] proposed a key word extraction method based on Word2Vec and TextRank (called W-TextRank). We compared the performances of five models on the Sina Weibo dataset, and the ratio of training data to testing data is 8 to 2. The experiment results are presented in Table 5.

Table 5. The experiment results of several textrank (Sina Weibo).

	N = 3			N = 5			N = 7			N = 10		
	P	R	F	P	R	F	P	R	F	P	R	F
TF-IDF	30.2%	33.3%	31.7%	27.9%	41.5%	33.4%	25.2%	35.7%	29.5%	22.6%	36.8%	28.0%
LDA	31.4%	28.3%	29.8%	30.5%	42.0%	35.3%	29.8%	34.5%	32.0%	26.1%	35.8%	30.2%
W-TextRank	36.5%	15.0%	21.3%	33.7%	25.0%	28.7%	30.4%	25.0%	27.4%	23.2%	35.0%	27.9%
PW-TextRank	35.6%	30.6%	32.9%	27.0%	38.3%	31.7%	21.7%	42.8%	28.8%	17.0%	47.9%	25.1%
Word-Doc2Vec TextRank	38.4%	47.5%	42.5%	36.9%	48.6%	41.9%	35.3%	55.4%	43.1%	33.2%	50.9%	40.2%

To fully observe the differences between the key word extraction methods, we assess the P, R, and F-measure values under different Ns, where N indicates the first N key words extracted, and the range of N is from [1] to [10]. Figure 2 shows a comparison of the accuracy rate, P, of the various models. Figure 3 shows a comparison of the recall rate, R, of the various models. Figure 4 shows a comparison of the F-measure of the

various models. We can clearly see that our Word-Doc2Vec-TextRank model has achieved good P, R, and F values.
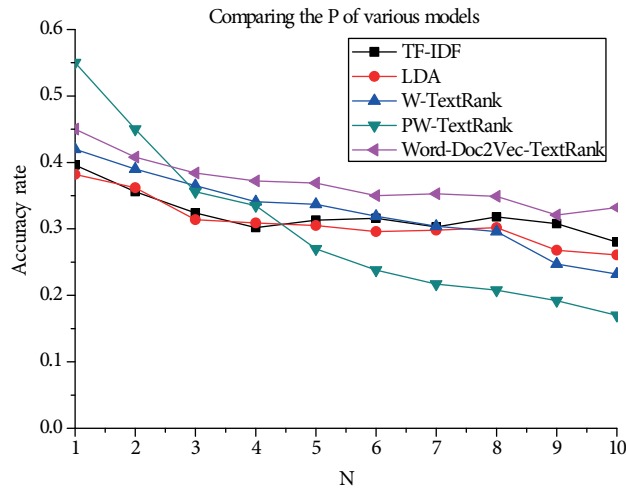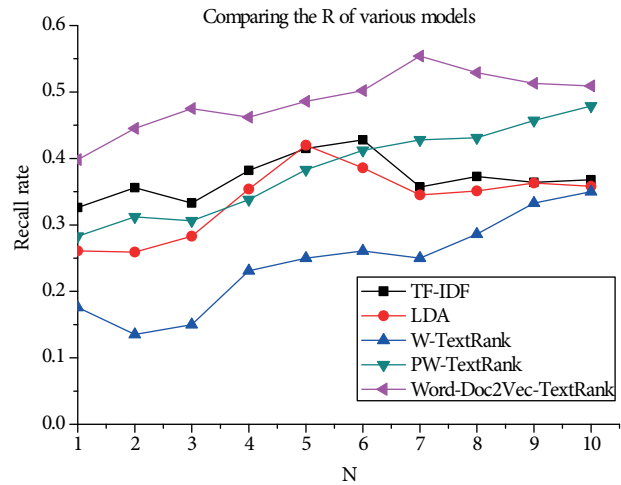

Figure 2. Comparing the P of various models.

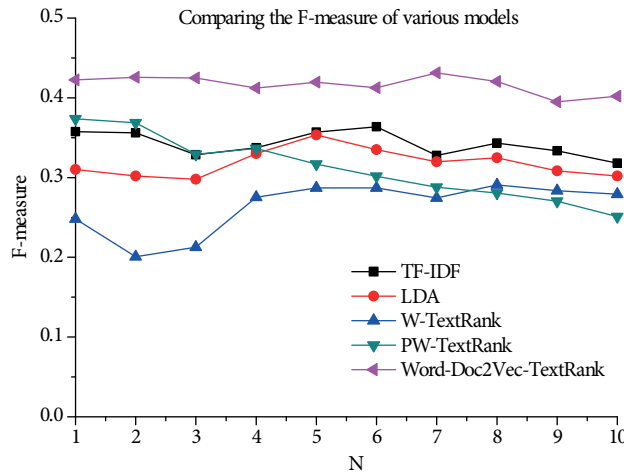**Figure 3**. Comparing the R of various models.

Figure 4. Comparing the F-measure of various models.

The following three figures show the comparison curves of the accuracy rate P, the recall rate R, and the F-measure values in the five models, respectively, and the performance results all are from the tests of short texts. In Figure 2, overall, the accuracy rate decreases with the increase of the number of top key words extracted (N). PW-TextRank performs well with a small number of key words $(N < 3)$, but as the number of key words extracted increases, the model's performance deteriorates sharply, which shows that our model is more stable. The performance of the W-TextRank model is stable, but the accuracy rate is lower than our model. We believe that this is due to the addition of the Doc2Vec-trained paragraph vectors to the model, which makes the semantic similarity more complete. This also reflects the feasibility of our model. In Figure 3, the performance of the traditional TF-IDF model is straight and narrow, and our model's performance is the best on all curves. As the value of N increases, the PW-TextRank model and our model reach similar results. In Figure 4, our model's performance is still the best and most stable with respect to the F-measure, and from the

curve, when the number of extracted key words N is 7, the F-measure reaches a maximum of 43.1%. Compared to Xia's work [22], we believe that the addition of Doc2Vec can improve the accuracy of key word extraction.

5. Conclusion

In this paper, we focused on short text key word extraction tasks. In our approach, we used the Word2Vec training word vectors to obtain the semantic relationships between words and relied on Doc2Vec to train the sentence vectors to curb the sparsity of short texts. Then, we used the improved TextRank algorithm to extract the key words. Our main work and contribution has added the sentence vector for key word extraction and joined the vectors to update the weights in the TextRank algorithm. To our knowledge, this is the first work that extracts key words using Doc2Vec. In the experiment, we compared the traditional TF-IDF and LDA models and added the TextRank related model. For further comparison, we also tested the performances of the models on long texts and short texts. The results show that our model performs best in short-text key word extraction tasks, and it is also smooth and reliable in long texts.

The experiment found that in addition to the exactly right key words, some other key words were mostly similar with the artificially marked key words. However, these words were not treated as the accurately computed results properly reflected in precision, recall and F-measure value. We will focus on solving the problem of synonyms in a future research.

Acknowledgments

References

[1] Rada M, Paul T. TextRank: bringing order into texts. In: 2004 Conference on Empirical Methods in Natural Language Processing; 25-26 July 2004; Barcelona, Spain. pp. 404-411.

[2] Page L, Brin S, Motwani R, Winograd T. The Pagerank Citation Ranking: Bringing Order to the Web. San Francisco, USA: Stanford InfoLab Press, 1999.

[3] Mikolov T, Sutskever I, Chen K, Corrago G, Dean J. Distributed representations of words and phrases and their compositionality. In: 27th Conference on Neural Information Processing Systems; 5-10 December 2013; Lake Tahoe, Nevada, USA. pp. 1-9.

[4] Mikolov T, Chen K, Corrago G, Dean J. Efficient estimation of word representations in vector space. In: International Conference on Learning Representations; 2-4 May 2013; Scottsdale, Arizona, USA. pp. 1-12.

[5] Le Q, Mikolov T. Distributed representations of sentences and documents. In: 31th International Conference on Machine Learning; 21-26 June 2014; Beijing, China.

[6] Matsuo Y, Ishizuka M. Key word extraction from a single document using word co-occurrence statistical information. International Journal on Artificial Intelligence Tools 2004; 13(1): 157-169.

[7] Ercan G, Cicekli I. Using lexical chain for key word extraction. Information Processing Management 2007; 46(6): 1705-1714.

[8] Willyan D, Leandro N. A key word extraction method from twitter messages represented as graphs. Applied Mathematics and Computation 2014; 240(1): 308-325.

[9] Yang K, Chen Z, Cai Y, Huang D, Leung H. Improved automatic key word extraction given more semantic knowledge. In: International Conference on Database Systems for Advanced Applications; 16-19 April 2016; Dallas, USA. pp.112-125.

[10] Xu S, Guo J, Chen X. Extracting topic key words from sina weibo text sets. In: International Conference on Audio, Language and Image Processing; 11-12 July 2016; Shanghai, China. pp. 668-673.

[11] Manna S, Oras P. Exploring topic models on short texts: a case study with crisis data. In: Second IEEE International Conference on Robotic Computing; 31 January- 2 February 2018; CA, USA. pp. 377-382. doi:10.1109/ICALIP.2016.7846663

[12] Önal S. Using latent semantic analysis for automated key word extraction from large document corpora. Turkish Journal of Electrical Engineering Computer Sciences 2017; 25: 1784-1794. doi:10.3906/elk-1511-203

[13] Bhavneet K, Sushma J. Key word extraction using machine learning approaches. In: 3rd International Conference on Advances in Computing, Communication Automation; 15-16 September 2017; Dehradun, India. pp. 1-6. doi:10.1109/ICACCAF.2017.8344699

[14] Wang Y, Zhang J. Key word extraction from online product reviews based on bi-directional LSTM recurrent neural network. In: IEEE International Conference on Industrial Engineering and Engineering Management; 10-13 December 2017; Singapore, Singapore. pp.2241-2245. doi:10.1109/IEEM.2017.8290290

[15] Ding S, Zhang X, Li O, Li S. Key word sequence extraction based on byte entropy iterative segmentation. In: 3rd IEEE International Conference on Computer and Communications; 13-16 December 2017; Chengdu, China. pp.1530-1535. doi:10.1109/CompComm.2017.8322796

[16] Xu C, Liu D. Chinese text summarization algorithm based on word2vec. Journal of Physics: Conference Series 2018; 976: 1-6.

[17] Ying K, Pan J, Wu M. Research on sentiment analysis of micro-blog's topic based on textrank's abstract. In: 2017 International Conference on Information Technology; 27-29 December 2017; Singapore, Singapore. pp. 86-90. doi:10.1145/3176653.3176698

[18] Saroj K, Monali B, Jacob S. A graph based key word extraction model using collective node weight. Expert Systems with Applications 2018; 97(1): 51-59.

[19] Georgios P, Vangelis K. Identifying argument components through textrank. In: 3rd Workshop on Argument Mining; 7-12 August 2016; Berlin, Germany. pp. 94-102.

[20] Zhao D, Du N, Chang Z, Li Y. Key word extraction for social media short text. In: 14th Web Information Systems and Applications Conference; 11-12 November 2017; Liuzhou, China. pp. 251-256. doi:10.1109/WISA.2017.12

[21] Wen Y, Yuan H. Research on key word extraction based on word2vec weighted textrank. In: 2nd IEEE International Conference on Computer and Communications; 14-17 October 2016; Chengdu, China. pp. 2109-2113. doi:10.1109/CompComm.2016.7925072

[22] Xia T. Study on key word extraction using word position weighted textrank. New Technology of Library and Information Service 2013; 29(9): 30-34.

[23] Wang Q, Sheng V, Wu X. Keyphrase extraction with sequential pattern mining. In: 31st AAAI Conference on Artificial Intelligence; 4-9 February 2017; San Francisco, USA. pp. 5003-5004.

[24] Liu X, Song Y, Liu S, Wang H. Automatic taxonomy construction from key words. In: 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 12-16 August 2012; Beijing, China. pp.1433-1441.

[25] Xia T. Extracting key words with modified textrank model. Data Analysis and Knowledge Discovery 2017; 2: 28-34.