Extracting Topics based on Word2Vec and Improved Jaccard Similarity Coefficient

Chunzi Wu
Data Science and Service Center
School of Computer Science, Beijing University of Posts and Telecommunications
Beijing, China
Email: wuchz0328@hotmail.com

Bai Wang
Data Science and Service Center
School of Computer Science, Beijing University of Posts and Telecommunications
Beijing, China
Email: wangbai@bupt.edu.cn

***Abstract*—To extract key topics from news articles, this paper researches into a new method to discover an efficient way to construct text vectors and improve the efficiency and accuracy of document clustering based on Word2Vec model. This paper proposes a novel algorithm, which combines Jaccard similarity coefficient and inverse dimension frequency to calculate the importance degree between each dimension in text vector and the corresponding document. Text vectors is constructed based on the importance degree and improve the accuracy of text cluster and key topics extraction. The algorithm is also implemented on MapReduce and the efficiency is improved.**

***Keywords*—Jaccard similarity coefficient;MapReduce; Kmeans;Word2vec.**

I. INTRODUCTION

With the continuous development of technology, people have been able to obtain large amount of news information every day. It is difficult to catch the key topics in a certain event, especially when people do not focus on the event at the first time. To solve the problem, editors in most news websites select main contents of events from enormous amount of news and present them to readers. It improves user experience indeed, but it is time consuming for editors to select topics manually. For this reason, we figure out a novel method to process news articles and generate key topics from them.

An event, as we mentioned before, owns several development stages from its occurrence to termination in a certain time period. We aim to extract key topics from event, the key topic is a topic which can represent one of the main development parts in an event.

The process of topic extraction is actually the document clustering. Document clustering has been researched by loads of researchers before. The key point of document clustering is document feature extraction, for example, TFIDF algorithm, LDA, etc. People firstly extract features from document, then cluster the document based on the features. The most normal way is constructing text vector, which means mapping document to vector with high dimension. So in this paper, we will concentrate on how to improve the accuracy of text vectors.

In the year of 2013, Google published a word vector training model called 'Word2Vec'[1], which help people train word to word embedding using deep learning. The model could not only train words, but also be suitable to train short document which only contains one sentence or less. However, Word2Vec model cannot directly apply to long document training, so in this paper, we are dedicate to find an efficient way to use the word embedding from Word2Vec model to construct text vectors and improve the accuracy.

Several researchers, for example, Xiaoming Zhang et al.[2] dedicate to detect topics from twitter. This paper proposes a novel algorithm to transfer word vectors to text vectors and extract key topics from news set. We firstly train word into vector using Word2Vec model and merge them into clusters, then we treat each word cluster as a dimension and calculate the similarity between document and dimension, which using Jaccard similarity coefficient and inverse dimension frequency. The similarity will be treat as the number of each dimension of every document.

Yanhong Yuan, Liming He, et al. [3] also utilized word clusters as dimensions, but the following process method is totally different. They simply count the number of words both occurred in a certain word cluster and every document, then use the number as the number in the dimension. But in this paper, after we get the word cluster, we will use statistic method to dig the relationship between word clusters and documents.

Our major contribution includes:

1. We use Jaccard similarity coefficient to calculate the relationship between a word cluster and a certain document instead of counting the number of co-occurrence words.
2. After calculating similarity by Jaccard similarity coefficient, we use inverse dimension frequency to calculate the importance of the word cluster to a certain document inspired by the idea of inverse document frequency in TFIDF algorithm. In our algorithm, it is called 'inverse dimension frequency'. We combine the two parts together and put forward a innovation model called Jaccard Similarity-Inverse Dimension Frequency (JS-IDF), which improve the accuracy of key topics extraction.

978-1-5386-1600-0/17 $31.00 © 2017 IEEE
DOI 10.1109/DSC.2017.70

3. The proposed algorithm is executed based on parallelized computing model MapReduce to improve the efficiency.

The first part of the paper is introduction, the second part will introduce some relative works of Word2Vec model, text vector construction and parallel platform. Our algorithm (JS-IDF) on construct text vectors will be explained in part III and document cluster process will be put in part IV. In fifth part, we will introduce why and how we execute our program on MapReduce and the sixth part will be the experiment and result. The seventh part will be the conclusion.

II. RELATIVE WORK

A. Word2Vec Model

Word2Vec is published by Google in 2013, it is an efficient tool that transfer word to distribution representation. It use distributed representation and transfers words into vectors in K dimensions. The advantage of Word2Vec model is that the model can reduce dimensions efficiently and contains abundant semantic meaning.

There exist two kinds of methods to transfer word to vector, one is one-hot representation, the other is distributed representation.

1) One-hot representation

It is the simplest way to map a word to a vector, which utilizes a vector with high dimensions to represent a words. The length of vector equals to the length of dictionary, which usually reach the length of 10^5. In this representation, the position of a certain word is '1' and the rest of positions are '0'. For example, 'potato' is represented as [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, …], and 'tomato' is represented as [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, …]. This method is able to transfer a word simply, but is lack of semantic meaning of the word. Both tomato and potato are belong to food, but we barely cannot use one-hot representation to calculate their similarity or underneath relationship. Obviously, one-hot representation cannot show the similarity between words.

2) Distributed representation

Distributed representation is proposed by Hinton [4] in 1986. It can map a word to a vector with relatively low dimensions, which is always 100 or 200. It can guarantee that words with similar meaning would have closer distance in vector space.

Word2Vec model trains words based on the idea of distributed representation. It uses two kinds of models, they are CBOW model and Skip-gram model (Figure 1). CBOW model uses context of the word w(t) to predict the current word, and Skip-gram model uses the word w(t) to predict its context.

The training process of CBOW and Skip-gram is similar, here we take CBOW model as an example to explain the training process. The input of CBOW model is *c* words from preceding part and following part of the word w(t) respectively, then the projection layer would add them up. The output layer uses the words occurred before in training materials as leaf nodes, and their occurrence time as weight to construct a Huffman tree. The model predicts the word w(t) through Random Gradient Ascent algorithm.

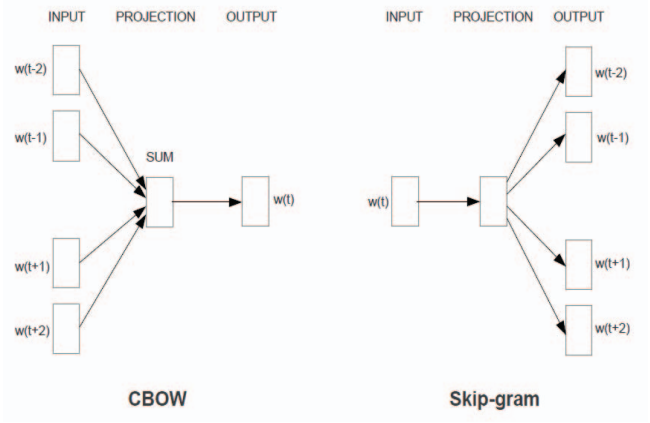

Fig. 1. CBOW and Skip-gram model [1]

After the training finished, we can get distribution representation of all the words. When we use word embedding to represent a word, we can find something like: 'king'-'man'+'woman'='queen' [5]. The wording embedding is efficient for the semantic meaning of a word.

Word2Vec model is applied to various research fields today. For example, words cluster research [6], dimensions reduction in massive text materials [7], text classification [8] and recommended algorithm [9] based on both Word2Vec model and LDA model, identity of named entity [10], etc.

B. Text Vector Construction

As we mentioned before, Word2Vec model is not suitable to long document training, so we need to find an efficient way to construct text vector based on distribution representation.

Before document cluster, the important step is generating text vector. The traditional way of generating text vector is 'bag of words'. This method always utilize word weighted model to extract feature words from document, such as TFIDF model, LDA model etc., then vectorize text into vector. LDA model is popular in recent years since it considers the semantic meaning of words. Peng Wang et al. [11] improved the accuracy of hierarchical clustering, solving the problem that the dimension of vector is high but sparse, and lack of semantic meaning in traditional method. Ruiji Fu, Bing Qin et al. [12] uses multi-LDA models to classify documents, they select 'good topics' based on the entropy of word distributional probabilities in the topics and improve the efficiency of labelling process.

Although LDA model contains semantic meaning of words, it is not enough to text analysis. In the paper, we use Word2Vec model, whose result contains more semantic meaning. Quoc Le et al. [13] proposes Doc2Vec model, which improves Word2Vec model and trains text vectors directly. Ming Tang et al. [14] combine TFIDF algorithm and Word2Vec model, weight word embedding by TFIDF values,

sum word vectors in each document up and the result is text vector. Joseph Lilleberg and Yun Zhu et al. [15] classify text with their semantic features based on support vector machines and Word2Vec model. Yanhong Yuan et al. [3] cluster words into word clusters firstly and treat them as dimensions, then count the number of the same words occurred in both document and dimension, the number of count will be the number of each dimension of certain document.

C. Parallelization

MapReduce, since it was introduced in 2004 by Jeffery Dean and Sanjay Ghemawat et al. [16], has become one of the most important programming models in data processing. It consists of two major steps of using "map" and "reduce" functions. This model use map() function to filter and sort data and generate "key-value" pair. The key-value pair will be used by reduce() function to perform summary operation. MapReduce can ensure that all the key-value pair can be run in different reduce() function.

In the paper, we use MapReduce to enhance the efficiency of the algorithm and the result is acceptable.

III. TEXT VECTOR CONSTRUCTION

First step of extracting key topics from news set is constructing text vector. In our method, text vector construction consists of two parts, word cluster and JS-IDF algorithm calculation. Figure 2 is a flow chat to show the whole process.

Definition 1 (Event). An event owns several development stages from its occurrence to termination in a certain time period.

Definition 2 (Key topic). A topic which can represent one of the main development parts in an event.

A. Process document into words and words training

For text set K which contains M documents, we use the open-source tool HanLP to cut each documents K_i (i=1, 2, 3, …, M) into words. We integrate all the documents into a text file T, in which each document written in a single line. Then we regard text file T as the input of Word2Vec model.

As a result, we can obtain n-dimension vector ω of each word, $\omega = (v_1, v_2, v_3, \ldots v_n)$. In this paper, we set the n as 200 and the window size as 8.

B. Cluster words

After we obtain distributed representations of all the words, we begin to cluster words into word clusters. We use singlepass algorithm to cluster in the beginning, the result is word cluster C, $C = (C_1, C_2, C_3 \ldots C_n)$. The next step is calculating center of each cluster. Then Kmeans algorithm is used to cluster all the words again to improve the accuracy of cluster result , the initial input are the center points from cluster C. All the cluster processes are based on cosine similarity, which is shown in Equation (1). The detailed cluster process will be introduced in part IV.

$$\cos(\omega_i, \omega_j) = \frac{\vec{\omega i} \times \vec{\omega j}}{\left|\vec{\omega i}\right| \times \left|\vec{\omega j}\right|} \quad (1)$$

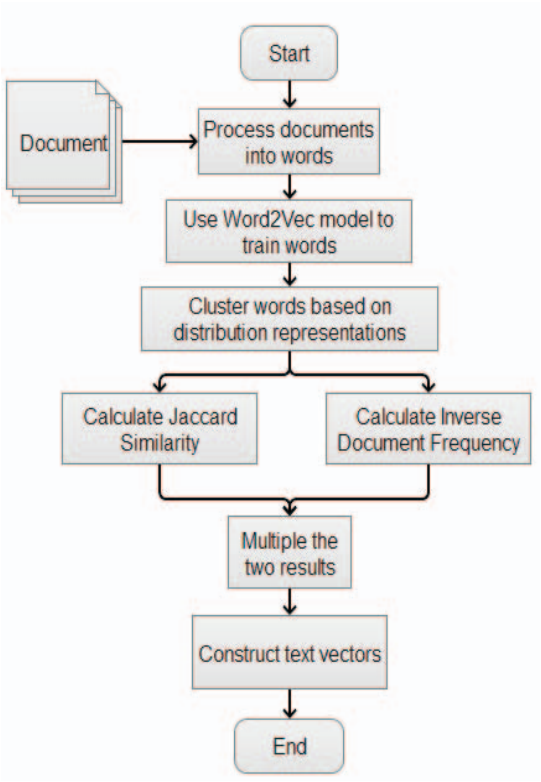

Fig. 2. Text vector construction process

We regard the word clusters we obtain after Kmeans algorithm as dimensions in text vectors. Each cluster corresponds to one dimension. We use the average number of words in news article to determine the number of word clusters. The details will be explained in part VI.

C. Jaccard Similarity-Inverse Dimension Frequency

1) Jaccard Similarity Coefficient

We will calculate the similarity between each dimension and every document. For each document, all the word clusters represent all the dimensions of their text vector. Yanhong Yuan et al. [3] also use the idea of word clusters. But they only count the number of co-occurrence words between dimension and document.

Now we have the word clusters D, $D=(D_1, D_2, D_3, \ldots, D_L)$ each word cluster is one dimension D_i, D_i is consists of words that have similar meaning, $D_i = (W_{i1}, W_{i2}, W_{i3}, \ldots, W_{iN})$. Document set is M, $M=(M_1, M_2, M_3, \ldots, M_l)$ and each document is defined $M_j = (w_{j1}, w_{j2}, w_{j3}, \ldots, w_{jn})$. M_j is composed by words of the document. We use Jaccard similarity coefficient to measure the similarity between D_i and M_j.

Jaccard similarity coefficient is usually used to measure the similarity of two sets. In this paper, dimension D_i and document M_j are also the word sets. It is represented by $J(D_i, M_j)$, the formula is shown in Equation (2).

$$J(D_i, M_j) = \frac{|D_i \cap M_j|}{|D_i \cup M_j|} \quad (2)$$

In equation (2), '$D_i \cap M_j$' means the number of words that co-occurred in both dimension *i* and document *j*. '$D_i \cup M_j$' means total count of words occurred in dimension *i* and document *j*. If a word appears more than once in both dimension *i* and document *j*, the word will be count only once in our calculation.

2) Inverse Dimension Frequency

When Jaccard similarity coefficient is calculated, we find a problem that we cannot tell whether one dimension is important or not to a certain document. Our purpose is extracting feature dimension to a certain document, for example, if words in one dimension are only contained by one certain document, the dimension is important to the document. Otherwise, if words in the dimension are contained by most documents, the dimension is not that important to those documents.

To solve the problem, we propose a novel algorithm which is inspired by TFIDF algorithm. We call it as JS-IDF algorithm. In our JS-IDF algorithm, *js* is Jaccard similarity coefficient and *idf* is inverse dimension frequency. The calculation of *idf* in our algorithm borrows the idea of *idf* calculation in TFIDF, but they are totally different. In our method, *idf* calculation is occurred between two word sets, but in TFIDF, the operation is happened between single word and word set. Equation (3) and Equation (4) show the details.

$$idf(D_i) = \log \frac{|K|}{\{i : W_{iN} \in (D_i, M_j)\} + 0.5} \quad (3)$$

$$jsidf_{ij} = J(D_i, M_j) * idf(D_i) \quad (4)$$

In equation (3), '*K*' represent total number of document in document set, and the denominator in equation (3) represent that for a certain dimension *i*, the number of documents which contains any of the words in this dimension. The reason of adding *0.5* to the denominator is to avoid the situation that denominator equals to 0. After the ratio is calculated, we do log calculation to the ratio. The result will be the idf value.

Equation (4) shows the process of computing *jsidf* value. For each document, *jsidf* value of one dimension is the product of Jaccard similarity coefficient between the document and the dimension and the inverse dimension frequency of the certain dimension.

D. Text Vector Construction

The *jsidf* values are used to build text vector. For every document M_j, we calculate *jsidf* value between M_j and dimensions D_i. The *jsidf* values $jsidf_{ij}$ are the number of dimension *i* for document *j*. Then we use all the *jsidf* values of document M_j to construct text vector V_j, $V_j = (jsidf_{1j}, jsidf_{2j}, jsidf_{3j} \ldots jsidf_{Lj})$, L represents the number of dimensions as we mentioned before.

Table 1 shows the pseudo-code of this process.

TABLE I. IMPLEMENTATION ON TEXT VECTOR CONSTRUCTION

1	**Input:** *M*, where *M* contains all documents in document set.
2	*JS_Map* ← js_{ij}, where *js* is the jaccard similarity coefficient for every document i-dimension j pair
3	*IDF_Map* ← idf_i, where idf_i is the inverse dimension frequency for every dimension.
4	**Output**: *V*, which are the collection of all the text vectors.
5	For every *js* ∈ JS_Map, do
6	$jsidf_{ij} = js_{ij} * idf_i$
7	*JSIDF_Map* ← $jsidf_{ij}$, where key is document-dimension pair, value is corresponding *jsidf* value.
8	For every *j* ∈ M, do
9	Find corresponding *jsidf* value of *j* in JSIDF_Map
10	*V* ← v_j, where v_j is vector of document *j*.

IV. DOCUMENT CLUSTER AND KEY TOPICS EXTRACTION

When we finish document vector construction, we utilize cluster algorithm to extract key topics from the document set. Two kinds of cluster algorithm are used, they are Single-pass algorithm and Kmeans algorithm. In part III, when we cluster words, we use the same process to obtain word clusters. In this part, we will take text cluster as an example to explain.

Figure 3 shows the process of document cluster.

A. Document cluster

1) Single-pass algorithm

Single-pass algorithm is a typical heuristic cluster method. It is easily affected by the input order, results of inputs with different order will be different. People need to set a threshold beforehand. The algorithm's input is text vectors and output is text clusters. The detailed steps are as following.

a) Step 1: Set a threshold *t*.

b) Step 2: Take the first input text vector as the first cluster.

c) Step 3: Calculate the cosine similarity of existed clusters $(c_1, c_2 \ldots c_n)$ and the next entered vectors *v* respectively. Take the maximum value *max* among all the similarities, compare it to *t,* if *max*≥*t*, add *v* to the cluster c_i that *max* belongs to; otherwise, treat *v* as a new cluster c_{n+1}.

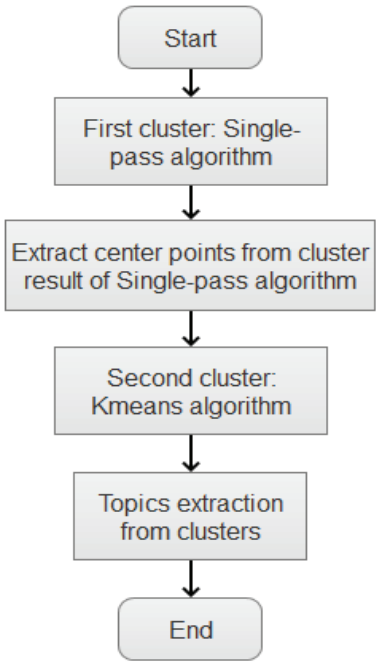

Fig. 3. Document cluster process

d) Step 4: if vector *v* is added into an existed cluster, recalculate the centroid of the cluster. But in this paper, we only utilize Single-pass algorithm to generate initial input of Kmeans algorithm, so the accuracy requirement is relatively low. As a result, we simply treat the first document vector in the cluster as its centroid to improve the efficiency.

e) Step 5: Repeat step 2 – step 4, until the algorithm finished.

The result of the first cluster is word clusters, we calculate centroid of each cluster and take them as the initial input of Kmeans algorithm.

2) Kmeans algorithm

Kmeans algorithm is one of the partitioning clustering methods. The idea of Kmeans algorithm is partitioning the points roughly at first and correcting the result based on iterative operation. People need to choose several vectors as the initial input before the algorithm. In this paper, we use the centroids of words clusters which are the result of Single-pass algorithm as the initial input.

The detailed steps of Kmeans are as following.

a) Step 1: Regard the centroids set $K = (K_1, K_2 \ldots K_n)$ from the first cluster process as the initial center points. Each center point represent a cluster.

b) Step 2: Calculate similarities of every new entered vector *v* and existed centroids separately, find the maximum value *max,* add *v* to the cluster that *max* belongs to.

c) Step 3: Repeat *step 2*, until all the vectors have been re-classified.

d) Step 4: Recalculate the centroids of cluster.

e) Step 5: Compare the centroids from *step 4* with the centroids from last execution, if the two result are identical, end the execution; otherwise, update the centroids by the result in *step 4* and repeat *step 2 – step 5*.

In practice, we will end the algorithm execution as long as the result is stable. After multiple experiments, we find that the average iteration times when the result can reach stable is 5. So we set the iteration times to 5 directly to improve the efficiency.

B. Key topics extraction

The text clusters generated by Kmeans algorithm is our final result. Topics are extracted from those clusters. We compute each cluster's center point, and calculate the similarities between the centroid and every document point. The most similar document point is the topic we find, the title of the document is the key topic title. We call those kinds of documents as "centroid document".

Definition 3 (Centroid document). A document in a cluster which is closest to the centroid of the cluster. The document is the key topic in the news set, and the title of the document is the key topic's title.

The key topic can represent the meaning of the whole data news cluster in large extent through our experiment before [17]. We use the centroid documents to represent all the cluster and the result is acceptable. So in this paper, we still follow our previous method, treat the centroid document as the representative of the centain cluster.

V. PARALLELIZATION OF JS-IDF ALGORITHM

Due to the news report dataset in practical use is huge and complicated, parallelization execution is introduced to improve the efficiency of processing data. As we mentioned in part II, MapReduce is a parallelized calculation model, it can allocate several reduce() function to process different key-value pairs.

Implementation on our algorithm JS-IDF consists of three parts, Jaccard Similarity coefficient calculation, inverse dimension frequency calculation and text vectors construction.

The main function of mapper class is mapping key and value, and reducer class is used to do distributed computation. For example, In Table II, we can use reducer class to process all the document at same time. In stand-alone version, we can only process the documents one by one. Inverse dimension frequency calculation and text vectors construction are operated in a similar way.

Pseudo-code of the three parts are list below.

TABLE II. IMPLEMENTATION ON JACCARD SIMILARITY COEFFICIENT

	Mapper Class:
1	**Input:** *M*, where *M* contains all documents in document set.
2	**Output**: *M,* which is in sequence file format. Key is name of document, value are the relative words.
3	For every $M_j \in M$, do

4	Split d into two parts, key is document name and value are words
	Reducer Class:
5	**Input:** *M*, which is from the mapper class. *D*, which contains all word clusters (dimensions), including cluster number and relative words.
6	**Output**: *JS,* which is Jaccard similarity coefficient of each document-dimension pair
8	For every $M_j \in M$, do
9	Calculate *t, t* is the total number of words occurred in both document *j* and cluster *i*
10	Calculate *c,* c is the number of words co-occurred in both document *j* and cluster *i*.
11	JS ← js_{ij}, $js_{ij} = c/t$

TABLE III. IMPLEMENTAION ON INVERSE DIMENSION FREQUENCY

	Mapper Class:
1	**Input:** *D*, which contains all word clusters (dimensions), including cluster number and relative words.
2	**Output**: *D,* which is in sequence file format. Key is cluster number, value are the relative words.
3	For every $D_i \in D$, do
4	Split d into two parts, key is cluster number and value are words
	Reducer Class:
5	**Input:** *D*, which is from the mapper class. *M*, where *M* contains all documents in document set.
6	**Output**: *IDF,* which is inverse dimension frequency of every dimension
7	For every $D_i \in D$, do
8	Calculate *t, t* is the total number of documents which contain words from dimension *i*
9	IDF ← idf_i, $idf_i = log(size\ of\ M/t)$

TABLE IV. IMPLEMENTAION ON TEXT VECTOR CONSTRUCTION

	Mapper Class:
1	**Input:** *JS* ← js_{ij}, where *js* is the jaccard similarity coefficient for every document i-dimension j pair
2	**Output**: *JS*, which is in sequence file format. Key is name of document *j*, value are Jaccard similarity coefficient js_{ij}, for dimension i and document j.
3	For every $js_{ij} \in JS$, do
4	Split js_{ij} into two parts, key is name of document *j*, value are Jaccard similarity coefficient js_{ij}, for dimension i and document j.
	Reducer Class:
5	**Input:** *JS* from mapper class
6	*IDF* ← idf_i, where idf_i is the inverse dimension frequency for every dimension.
7	**Output**: *V*, which are the collection of all the text vectors.
8	For every *js* ∈JS_Map, do
9	$jsidf_{ij} = js_{ij} * idf_i$
10	*JSIDF_Map* ←$jsidf_{ij}$, where key is document-dimension pair, value is corresponding *jsidf* value.
11	For every $j \in$ D, do
12	Find corresponding *jsidf* value of *j* in JSIDF_Map
13	*V* ← v_j, where v_j is vector of document *j*.

VI. EVALUATION AND RESULT

A. Evaluation data

Eight datasets are used to assess the model, which are listed in table V

TABLE V. EVALUATION DATASETS DESCRIPTION

No.	Name	Number	Size
1	Labeled news set 1	2816	11.2 MB
2	Labeled news set 2	1418	8.17 MB
3	Park Geun-hye Event	1449	6.58 MB
4	Test news set 1	21801	77 MB
5	Test news set 2	29821	112 MB
6	Test news set 3	39038	140 MB
7	Test news set 4	45361	161 MB
8	Test news set 5	56504	198 MB

Dataset 1, and database 2 are labeled news set. Each of the labeled dataset is classified into 10 classes, such as environment, politics, computer science and so on. Files in every class start with the same number. For example, documents in environment class start with '5', so the file names in this class are: 51.txt, 52.txt. 53.txt, … In other word, '5' can represent the class 'environment', so when we see a file name starts with '5', we will know it is from class 'environment.

Dataset 3 is news about Part Feun-hye event collected from Internet. Key topics extracted from dataset 3 will be compared with news topics of Part Feun-hye event, which selected by editors in netease.com. The four dataset are used to evaluate the accuracy of our method.

Dataset 4 to Dataset 8 are the test news set on penalization. We use 5 different size of news set to compare the time efficiency of stand-alone version and parallelization version. The result will be shown in Part C.

B. Evaluation Methodology

1) JS-IDF algorithm

The purpose that we put forward the JS-IDF algorithm is that we want to find an efficient way to extract topics from the event news set. Based on this requirement, we use two kinds of dataset to test our method. One is labelled news set, the news set is classified into 10 news classifications, so we

cluster the labeled news set into ten classes and calculate the centroids of the classes and extract the centroid document from each class. In this case, we can extract 10 topics from 10 classes. In the ideal condition, the 10 topics should exactly belong to the 10 news kinds respectively. We can use this standard to compare our method with others. The parameter, accuracy R is calculated to reflect the accuracy of different models. The method identifies N topics and the total correct topics is M.

$$R = \frac{N}{M} \tag{5}$$

We also use a dataset of Park Geun-hye event to evaluate our method's effect on event extraction in reality. We find a news special called 'Impeach on Park Geun-hye' on netease.com. [18] In the news special, editors sort out 20 topics of Park Geun-hye event, which can be regarded the standard to evaluate our result.

2) Parallelization of JS-IDF algorithm

In this paper, we implement our JS-IDF algorithm on MapReduce and serial version respectively. We improve the performance of our algorithm through parallelization. Various datasets with different size are processed to test execution time of the whole process from calculating JS-IDF value to constructing document vectors.

C. Evaluation result and analysis

1) JS-IDF algorithm

The method *JS-IDF* is compared to Ming Tang's [13] method, which is efficient in file classification. Ming Tang et al. use TFIDF values to weight word embedding. Firstly, they obtain word embedding through Word2Vec model. Then they calculate TFIDF value of each word in data set. Finally, for each document, all the vectors of words in the document is weighted by their TFIDF values and are added together. The final result is the text vector. We call the method as *TFIDF_WordVec* method. After generating text vectors, we utilize the text vectors for text clustering and extract the document which is closest to centroid as the topic.

In our method, the number of word cluster is determined by the average number of words in news articles, as we mentioned before. The numbers of dimensions are the result of total words in the dataset divided by the average words in news article.

After we observation, the average number of words in two labeled news set is around 30. So As a result, the numbers of dimension of labeled database1 and database2 are 517 and 377 respectively.

Table VI shows the results of *TFIDF_WordVec* and *JS-IDF* respectively. The bolded part in every file name is the class number, as we mentioned in the first section of this part. The documents in Table VI are organized in ascending order based on class number. Table VII shows the accuracy of the two methods.

TABLE VI. RESULT OF TWO METHODS

Labeled news set 1		**Labeled news set 2**	
TFIDF_WordVec	**JS-IDF**	**TFIDF_WordVec**	**JS-IDF**
1135.txt	**1**95.txt	**5**108.txt	**1**102.txt
4121.txt	**2**197.txt	**6**193.txt	**2**131.txt
563.txt	**4**196.txt	**7**143.txt	**4**144.txt
7185.txt	**5**72.txt	**7**115.txt	**5**147.txt
847.txt	**6**174.txt	**8**176.txt	**6**222.txt
853.txt	**8**10.txt	**9**401.txt	**6**273.txt
8135.txt	**9**97.txt	**10**385.txt	**6**257.txt
9409.txt	**9**14.txt	**10**443.txt	**8**198.txt
9385.txt	**9**129.txt	**10**286.txt	**8**157.txt
955.txt	**10**389.txt	**10**367.txt	**9**332.txt

TABLE VII. ACCURANCY OF TWO METHODS

No.	**Name**	**R*(TFIDF_WordVec)***	**R*(JS-IDF)***
1	Labeled news set 1	0.6	0.8
2	Labeled news set 2	0.6	0.7

The dataset is processed by *TFIDF_WordVec* and *JS-IDF* separately, the results of those two methods are text vectors from news sets. Then we cluster documents through the text vectors and extract the document which is closest to cluster centroid as the final result.

We list the final results, which are the key topics of both two labeled news sets in Table VI. As we mentioned before, the labeled news set consists of 10 classes, so we should have ten topics start with 1, 2, 3, … , 10 in the final result in ideal condition after clustering. In fact, the accuracy of the two methods cannot reach 100%, so the results contain some repeated classes and some missing classes. For example, in the result of JS-IDF method in labeled news set 1, class '9' occurred three times but class '3' and class '7' are missing.

As a result, table VII shows the accuracy of the two methods in different data sets.

We also want to figure out how our method working in the actual news key topics extraction, so we grab 1449 news articles about Park Geun-Hye event from People's Daily Online as the test data set.

Table VIII shows the comparison of the results of *TFIDF_WordVec* and *JS-IDF* in Park Geun-Hye event. The column 'title' is 20 titles selected by editors in *news.163.com* [17]. Because of the fuzziness of description, we use the way as table VIII shows to demonstrate the matching degree between the key topics we extract and the standard key topics, instead of listing all the result in the paper.

TABLE VIII. COMPARISON OF TWO METHOD IN PARK GEUN-HYE EVENT

	Title	TFIDF_Word Vec	JS-IDF
1	Encrypted files found in Choe Sum Sil's computer and people call for impeach to president Park	√	
2	Park's constitutional amendment may fail due to the impeach		√
3	House of Park's secretary is searched	√	√
4	Choe Sum Sil, who is Park Geun-hye's best friend, is arrested		√
5	Procurators begin to investigate Park's voice memo	√	√
6	Park Geun-hye apologize to the public		√
7	Park Geun-hye nominates the procurators to investigate the event.		√
8	Eight consortiums in South Korean are under investigation due to Park's event	√	√
9	Choe Sum Sil's another computer is found.		
10	Choe Sum Sil refuse the calling, court summon Choe compulsorily		√
11	Chong Wa Dea refuse being searched by law enforcement officer	√	
12	Park Geun-hye becomes criminal suspect		√
13	Chong Wa Dea refuse being searched by law enforcement officer again	√	
14	Samsung is under investigation due to Park's event		√
15	Law enforcement officers fail to search Chong Wa Dea again	√	
16	Park Geun-hye denies all the accuse	√	√
17	The special investigation group prosecute 17 suspects	√	√
18	Park is confirmed taken bribes from Samsung		√
19	Park Geun-hye is recalled	√	√
20	Park Geun-hye is arrested and under investigation	√	√
Accuracy		**0.55**	**0.75**

In table VIII, we consider both title and content of topics, if the topic we extract has similar meaning to the standard topic, we treat it as a correct one. Because the descriptions of the same topic may be different, we cannot judge whether a extracted topic is correct only through title comparison.

In the table, *TFIDF_WordVec* method collects 10 matching key topics and *JS-IDF* method collects 13 matching key topics under same document clustering algorithm. The accuracies are 0.55 and 0.75 respectively.

2) Parallelization

We implement our JS-IDF algorithm in MapReduce to improve the efficiency, the hardware and software information of the cluster is list in Table IX.

TABLE IX. INFORMATION ABOUT CLUSTER

Hardware Information			
CPU	**Memory**	**Exchanger**	**Number of machines**
Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.1GHz*4	64GB RAM	1000M	5
Software Information			
Operating System	**JDK**	**Hadoop**	
CentOS release 6.5	1.70.51	2.6.0	

Details on how to implement the JS-IDF algorithm is mentioned in part V. Figure 4 shows the comparison of two versions.

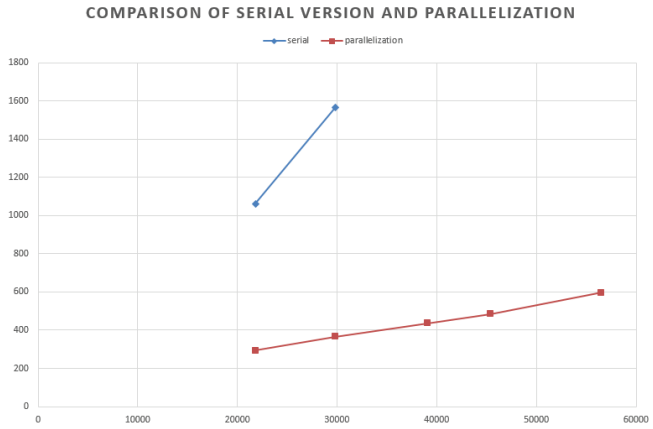

Fig. 4. Comparison of serial version and parallelization version

In figure 4, the X axis represents the number of news in test news set and the Y axis represents the elapsed time of each experiment. The unit of Y axis is second. Serial version is only support test news set 1 and 2, which contains 21801 and 29821 news separately. The test news set 3, which consists of 39038 news, cannot be executed successfully in serial version because of the out of memory error. So the time cost of serial version only contains two valid data.

The conclusion is obviously in figure 4, the parallelization version which is deployed in MapReduce is more efficiency than the serial version. As the number of news increases, the executed time of penalization version grows up slowly and steadily.

VII. CONCLUSION

People are exposed to enormous amount of news events nowadays. This paper dedicates to identify key topics in huge amount of news articles accurately and efficiently, a novel algorithm is proposed to transfer word embedding to text

vectors because the accurate text vector is one of the most important part of text clustering and key topic extraction. The algorithm is also implemented on MapReduce to improve its efficiency. It can be further used in event evolution analysis and hot topics extraction.

ACKNOWLEDGMENT

This work was supported by National Basic Research (973) Program of China (No. 2013CB329606) and Special Fund for Beijing Common Construction Project.

REFERENCES

[1] T. Mikolov, K. Chen, G. Corrado, et al., "Efficient Estimation of Word Representations in Vector Space". Computer Science, 2013.

[2] Xiaoming Zhang,Xiaoming Chen,Yan Chen, Senzhang Wang,Zhoujun Li,Jiali Xia: Event detection and popularity prediction in microblogging.Neurocomputing 149: 1469-1480 (2015)

[3] Y. Yuan, L. He, L. Peng, et al. "A new study based on Word2vec and cluster for document categorization". Journal of Computational Information Systems, 2014, 10(21), pp.9301-9308.

[4] GE. Hinton, "Learning distributed representations of concept", In: proceedings of the Eighth Annual Conference of the Cognitive Science Society. Amherst, Mass. 1986, pp.1-12..

[5] T. Mikolov, S W. Yih, G. Zweig, "Linguistic Regularities in Continuous Space Word Representations", 2013, vol. 2, pp.296-301.

[6] W. Zheng, P. Xu. "Research on Chinese words clustering based on word2vec model", Software, 2013, vol. 12, pp.160-162.

[7] L. Ma, Y. Zhang. "Using Word2Vec to process big text data". In: Proceeding of IEEE International Conference on Big Data. IEEE, 2015, pp.2895-2897.

[8] R. Ju, P. Zhou, C. H. Li, et al. "An Efficient Method for Document Categorization Based on Word2vec and Latent Semantic Analysis", In: proceedings of IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. IEEE, 2015, pp.2276-2283.

[9] W. Dong, "Research of recommendation algorithm based on LDA and Word2Vec model", Doctoral dissertation, Beijing University of Posts and telecommunications, 2015.

[10] SK. Sien, "Adapting word2vec to Named Entity Recognition", In: proceedings of Nordic Conference of Computational Linguistics, 2015.

[11] P. Wang, Y. Gao, X. Chen, "Research on text clustering based on LDA model", Information Science, 2015, pp.63-68.

[12] R. Fu, B. Qin, T. Liu. "Open-categorical text classification based on multi-LDA models", Soft Computing, 2015, 19(1), pp.29-38.

[13] Q. V. Le, T. Mikolov, "Distributed Representations of Sentences and Documents", Computer Science, 2014, vol. 4, pp.1188-1196.

[14] M. Tang, L. Zhu, X. Zhou, "A text vector representation based on Word2Vec model", Computer Science, 2016, 43(6), pp.214-217.

[15] J. Lilleberg, Y. Zhu, Y. Zhang, "Support vector machines and Word2vec for text classification with semantic features", In: proceeding of IEEE, International Conference on Cognitive Informatics & Cognitive Computing. IEEE, 2015, pp.136-140.

[16] J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters." In: proceeding of Conference on Symposium on Opearting Systems Design & Implementation. DBLP, 2004, pp.137-150.

[17] C. Wu, B. Wu, B. Wang, "Event Evolution Model Based on Random Walk Model with Hot Topic Extraction", In: proceeding of Advanced Data Mining and Applications. Springer International Publishing, 2016.

[18] https://c.m.163.com/news/s/S1477710860903.html?spss=newsapp&spsw=1 (News special of Park Geun-Hye)