International Journal of Multimedia and Ubiquitous Engineering
Vol.11, No.7 (2016), pp.159-168
http://dx.doi.org/10.14257/ijmue.2016.11.7.17

Near Duplicate Document Detection using Document Image

[1]Gaudence Uwamahoro, [1]Zhang Zuping*, [1]Ambele Robert Mtafya, [2]Weiqi Li, [1] and Long Jun

[1]School of Information Science and Engineering, Central South University Changsha, 410083, China
[2]School of Electronic and Information Engineering, Xi'an Jiaotong University Xian, 710049, China
*gauwa2002@yahoo.fr, * zpzhang@csu.edu.cn, kakaambe@gmail.com, liweiqi@stu.xjtu.edu.cn, jlong@csu.edu.cn*

Abstract

With development, access of Internet has allowed storage of huge documents containing information. Identifying near duplicate documents among those documents is a major problem in information retrieval due to their dimensionality which leads to high cost time. We propose an algorithm based on tf-idf method with importance and discriminative power of a term within a single document to speed up search process for detecting how documents are similar in collection. Using only 26.6% of original document size, our method performs well on efficiency and memory usage as we have reduced compare to the original one and that leads to a decreased time in searching process for similar documents in a collection.

***Keywords**: near duplicate document, tf-idf, document image, document relevance, keywords extraction*

1. Introduction

As the number of digital documents continues to grow, the need for an efficient method for getting the useful information needed has become apparently evident. Intervention of information retrieval techniques and text mining has played a big importance to the similar and near duplicate documents detection. The use of features from each document can be used as documents content representation but the extraction of those features must be taken in consideration to get the strong and useful features. There are a lot of words in a document that don't serve much in the comparison task and they consume memory space during searching process. The existence and the weight of terms are the most factors to consider during features selection. Identifying features will help to get a fast way to find similar documents and related ones. There is a proliferation of similar and almost similar text documents containing useful information because digital documents are dynamic and continually changing. It is a challenge to get information needed by the user because of the big volume of data, the search system must retrieve that information efficiently and effectively to satisfy the user request as in [1]. Lack of speed in searching for a query and relevance ranking is an obstacle. Practically the user needs the relevant documents to his query but he/she is interested most on high relevant documents *i.e.* documents returned first. Probably those documents are documents which contain terms of query that have high frequency. Extracting keywords from a text is closely related to ranking terms in the text by their relevance. In keywords extraction, there are different approached for keyword extraction: simple statistic approaches, machine learning approaches, linguistic approaches and combination of different methods. In this paper we proposed *tf.idf* approach in statistic approach where the *tf.idf* weight evaluates the importance of a term to a document in collection.

2. Related Works

The existence of near duplicate documents has the source from some changes made on original document such as delete, insert and substitution as in [2]. There are several approaches to detect those documents like edit distance, fingerprinting, shingling, checksumming and bag of word, WPBADS algorithm and similarity measures also are used as in [3-4]. Being efficient only for small size document is the major drawback for those approaches. The shingling algorithm was proposed by Broder *et al.* [5]. More shingles share two documents more documents are similar. The problem with shingles is that the size of shingles is greater than the size of document itself. That drawback of increasing the space required to store the index leads to slow the time for serving result. To improve shingling method, Fetterly *et al.* in [6] used five-grams as a shingle and sample 84 shingles for each document then 84 shingles are built into six super shingles. Documents having two super shingles in common are considered as near duplicates. Those issues in shingling method incited researchers on the way to deal with the high dimensionality of shingles.

Broder in [7] proposed to reduce the complexity of Shingling for processing large collections, by the use of super shingles with meta-sketches. Charikar in [8] proposed a method based on random projection of words in documents to detect near duplicate documents and improved overall precision and recall. Henzinger combined two algorithms; one proposed by Broder in [7] and algorithm proposed by Charikar in [8] as in [9]. Henzinger improved on precision compared to using the constituent algorithms individually. Most of the methods used have the same shortcoming of being efficient only for small size. Fingerprinting method has been proposed to overcome that drawback. Heintze and Manku invented the fingerprinting technique where every shingle is fingerprinted as in [10] and documents are near duplicates if one of the fingerprints matches. To convert shingles in fingerprints it is used a method proposed by Robin in 1981. Xiao, W., L. and J. in [11] proposed a new filtering technique by exploiting the ordering information in prefix filtering. Lakkaraju, P.Gauch, S., Speretta and M., proposed a method based on conceptual tree where they presented each document as a tree [12]. Near duplicate documents detection methods have main aim of evaluation of similarity score to satisfy the user request. The user wants the system to return first results he/she is interested in *i.e.* documents that contain almost similar to what is looking for.

The most popular technique for ranking used is *tf-idf* and it has been used for retrieving documents in [13]. The *tf-idf* measure combines two aspects of a word: the importance of a word for a document and its discriminative power and it uses the number of documents in which a word is used. The smaller the number, the more distinguishing the word is. Sadakane in [14] proposed a method to compute *tf-idf* scores of each retrieved document. However, what is lacking in his method is the notion of top-*k* documents with the highest *tf-idf* scores. Extracting keywords from a text is closely related to ranking words in the text by their relevance for the text. Keywords extraction is based on the *tf-idf* measure. In [15], *tf-idf* was used for discriminating non-keyphrases and highlights key-phrases which are particularly in given document. The *tf-idf* score of a phrase is considered as a standard metric that measures how specific a phrase is to a given document. Hannaneh and his group used *tf-idf* for detecting email campaigns and conducted an experiment to test the performance degradation when removing some unigrams of the vectors based on their importance, judged by the weighting score as in [16]. In our method we use *tf-idf* to measure the most important terms in each document which help to know how documents in collection are similar to each other.

3. Proposed Method

The dimensionality of data containing the needed information is an obstacle in information retrieval especially for detecting near duplicate documents from that huge data. Several methods have been proposed to speed up search process by reducing their size using different features selection like shingles. The words of document have different importance, some are more important than others and can help to get want we need from documents like knowing how documents are similar to each other and we call them keywords in this paper. Extracting keywords from each document is a good strategy to improve efficiency in near duplicate documents detection. The importance of a word in document is obtained by *tf-idf* method to weight each word. This method is used in vector space model where each document is represented as a vector of words weight; if a word occurs in a document, its value in vector is non zero. We choose to use this method because of its simplicity based on linear algebra, giving word weights besides the commonly used binary values (0 or 1) thus enabling ranking document according to their possible relevance. The aim of our proposed method is to increase efficiency by reducing memory usage.

There are four main stages in our method: the first one is preprocessing where each document is preprocessed by stopwords and punctuations removal, lowering each character and stemming using Porter stemmer. The second is constructing image of each document made of highest important words, and documents comparison and ranking using Jaccard similarity as described in Figure 1.

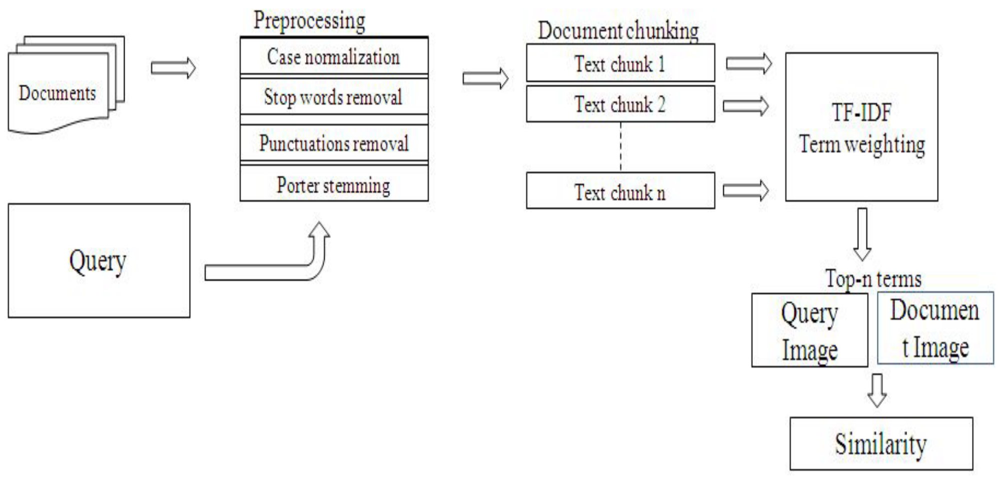

Figure 1. Weighted Terms Filtering Method for Image of Document

3.1. Document Image

We proposed a method based on words(terms) with high weight in each document that will consider the fixed number of the top words with high weight calculated by *tf-idf* method to represent each document instead of using all words in document vector representation and maintain the relationship with other documents we are comparing. The ranking order of a document used as query against other document is remaining the same as the one obtained when a document is compared to other documents using all words of document. The document image (*DI*) allows the reduction the of document vector size which leads to increase of efficiency.

3.2. Algorithm Description

The proposed method takes advantage of the importance of words in document to speed up similarity calculation between pair of documents using Jaccard. We call important words keywords terms in this paper and those keywords terms from a document are combined to make an image of document (*DI*) used in documents similarity calculation. The importance of a word (term) is measured using *tf-idf* technique that calculates the weight of each term in document where $tf_{t,d}$ (term frequency) of term *t* in document *d* is defined as the number of times that *t* occurs in *d* and the weight of *t* is: $w_{t,d} = 1 + log_{10}\ tf_{t,d}$, if $tf_{t,d} > 0$ and 0 otherwise. The *idf* (inverse document frequency) of *t* idf_t is the number of documents that contain *t* and it is given by $idf_t = log_{10}\ N/df_t$. Then the *tf-idf* weight of a term *t* is the product of its *tf* weight and its *idf weight*. The weight of term *t* is: $w_{t,d} = (1+log tf_{t,d}) \times log\ N/df_t)$. As the documents in collection have different lengths *i.e*. the number of terms in each document, that has impact on weights of the words (terms). In our method we give chance to a term to be weighted in documents of equal size. We consider a document to be a set of multi small documents of equal size called sentences *i.e*. document *d*i = $\{s_1, s_2, s_3, \ldots, s_n\}$ where $s_1, s_2, s_3, \ldots, s_n$ are the sentences with equal size, *n* is the number of sentences in document d_i.

The proposed method is described as follows: the collection *A* is a set of *m* documents, therefore $A = \{d_1, d_2, d_3, \ldots, d_m\}$. Each document d_i is split into equal *n* sentences *S* with a fixed minimum threshold *t* length where *t* is the number of terms in every sentence, hence $d_i = \sum_1^n t_i$. After getting sentences in every document, then we calculate *tf-idf* for each term in *S*. Top-*k* terms with high *tf-idf* score from each sentence are picked to represent the sentence in d_i where *k* is the number of terms with high *tf-idf*. Therefore $d_i' = \sum_1^n t_i'$ where t_i' is the length/size of top-k in Si and $t_i' \in t_i$ and we call d_i' document image (DI) and is used to represent di during the comparison between documents in order to know the relationship between documents in collection. Representing a document by its image helps to get a reduced document in size with important terms and that leads to a reduced memory used by document during comparison and leads also to the low cost of times for comparison. The proposed method has 3 parts: to determine the threshold for document image representatives using Empirical-Determining Threshold Algorithm (EDTA), identifying terms for image of document using KIA algorithm, and getting similarity between documents.

To make image of each document in collection when we are looking how documents are similar to the given document using Jaccard similarity, the minimum size of document that can represent that document instead of using whole size of document is needed so that the order of ranked document remains the same as the one using whole size of document called original document. Our method aims also to captures the maximum number shared terms between pairs of documents in comparison. Choosing the threshold size for document image (top-*n*), we need to know the similarity between documents and the ranking order of similarity scores based on the size of original documents and choose reduced size of original document based on selection of high weight of terms that can represent document as image. We found the threshold empirically using the following steps:

Step 1: Get ranked similarity list from the original documents in collection (documents with all terms without terms weighting using *tf-idf*).

Step 2: get ranked similarity list from representative images of documents. We consider the minimum size (in terms) of image made by the union of 5 terms with high weight calculated using *tf-idf* method from each sentence in document where document *D* = $\{sentence_1, \ldots, sentence_2, \ldots, sentence_n\}$, and the gape of 5 is used; *i.e*. first five terms with high weight are used as images, then ten, twenty, thirty, up to the fixed maximum terms.

Step 3: Compare the order of each representative image to the original order.

Step 4: If the order of similarity is similar the order of similarity of original documents, then take that threshold as image. Here the threshold is the number of top terms (first terms) with high weight from each sentence in each document and those terms can represent that document as image

The algorithm for that method is describes as follow:

Empirical Determining-Threshold Algorithm (EDTA)

Input: collection of document $D = \{d_1, d_2, \ldots, d_n\}$

Output: threshold

```
1. Get list(ranked list)
   $L_o$←sort (sim ($Q$, $D_i$)) //where Q is document to compare with other documents, $L_o$ is the list of similarity scores using original documents and i = document1, 2, 3 ...:
2. for $x$ = [5, 10, …, max]
3.     Get($Q$x, $D$x) // where x is the union of terms with high weight from each sentence in each document.
4.     $L_{img}$ ←sort(sim ($Q_x$, $D_x$)) // where $L_{img}$ is the list of similarity scores using images of documents
5.     if True
6. OrderCompare ($L_o$, $L_{img}$)
7.         $x$= threshold
8.     end if
9. end for
```

Keyterms Identification Algorithm (KIA)

Input: D: document, n: number of words in each sentence

Output: list of keywords extracted DI

```
1. DI← empty list
2. Split D in s [ ] sentences of n equal size
3. for each sentences in $s_i$
4.     L [ ] = calculate tf-idf for each word in sentences in $d_i$
5.     Sort (L) ← take top n tf-idf
6.     DI← DI ∪ L
7. end for
8. Return DI
```

Images Similarity Algorithm

Input: d_i, d_j: documents in collection D where D is a set of documents images (DI)

Output: similarity score between two documents

```
1. for all ($d_i$, $d_j$) ∈ D
2.     sim = jaccard($d_i$, $d_j$)
3. end for
4. Return sim
```

Image Query Algorithm

Input: D: set of documents

Output: k document in D

```
1. for each documents $d_i$ in D
2.     simk←sim (k, $d_i$)
3. end for
4. Return sort (simk)
```

3.3. Time Complexity

The operations considered most are the one for identifying document image and documents comparison. The time spent for building image of document depends on the input size of sentences in document. For each iteration, the algorithm calculates *tf-idf* of terms in sentences and that operation time complexity is $O(s)$ where s is the number sentences in a document. Operation of calculating similarity between two documents is $O(t)$ where t is number of terms of documents to compare.

4. Experiment and Analysis

We are based on collection of 22 documents with different sizes, to get the similarity between documents in the same group, by choosing one document as query against documents in the group. Table 1 show the different documents used with their size.

Table 1. Documents Size

D*id*	Tot.size	D.image (% in size)	No.sent	D*id*	Tot.size	D.image (% in size)	No.sent
d_1	733w	211w(28.78%)	7	d_{12}	316w	90w(28.48%)	3
d_2	805w	230w(28.57%)	8	d_{13}	441w	121w(27.43%)	4
d_3	558w	149w(26.78%)	5	d_{14}	472w	120w(25.42%)	4
d_4	533w	150w(28.74%)	5	d_{15}	866w	242w(27.94%)	8
d_5	651w	180w(27.64%)	6	d_{16}	820w	244w(29.75%)	8
d_6	349w	90w(25.78%)	3	d_{17}	546w	151w(27.65%)	5
d_7	662w	181w(27.34%)	6	d_{18}	931w	274w(29.43%)	9
d_8	642w	180w(28.03%)	6	d_{19}	768w	215w(31.71%)	7
d_9	287w	61w(21.26%)	2	d_{20}	874w	242w(27.68%)	8
d_{10}	300w	30w(10%)	3	d_{21}	631w	181w(28.68%)	6
d_{11}	671w	61w(9.09%)	6	d_{22}	736w	216w(29.34%)	7

We have five groups of groups where documents with different sizes are included in documents ranking to know how documents are similar to the query. In Table 1, abbreviations D*id*, Tot.size, D.image and No.sent represent respectively document identification, total size (in terms) of original document after being preprocessed, total size of document after being preprocessed and represented by top- terms selected based on the tf-idf score, number of sentences in each document. Collection L is made of five groups where $L = \{A, B, C, D, E\}$ and each group with different documents, $A = \{d_{15}, d_{22}, d_8, d_7, d_3, d_1, d_{14}, d_4, d_9\}$, $B = \{d_4, d_6, d_8; d_3, d_{20}, d_7, d_9\}$, $C = \{d_5, d_6, d_{22}, d_8, d_{21}, d_3, d_4, d_{14}, d_{16}, d_{13}, d_{10}, d_{12}\}$, $D = \{\mathrm{d}_7, \mathrm{d}_5, \mathrm{d}_8, \mathrm{d}_{15}, \mathrm{d}_{14}, \mathrm{d}_2, \mathrm{d}_3, \mathrm{d}_4, \mathrm{d}_{11}\}$, $E = \{d_8, d_{15}, d_7, d_5, d_4, d_3, d_2, d_9, d_{11}, d_{12}\}$. From the collection L with the size in terms of 13592, each document is split in equal sentences of one hundred terms as minimum threshold and first 30 terms with high weight calculated using *tf-idf* method in each sentence are joined to make image of that document. By using images of documents the size of new collection used in our method is only 3619 terms as 73.4% of the original of collection size has reduced. We found that considering top-30 terms with high weight from each sentence in document as image of document is a suitable minimum threshold of image document that can maintain the same ranking order as original documents during comparison. As we take one document image as the query against the collection to know how documents are similar to each other and know the highest similar to the query, our method show almost the same result in ranking order as the result used when comparing original documents where the similarity is based on Jaccard similarity measure. The documents similarity scores are seen in Table 2.

Table 2. Documents Ranking and Comparison

Q1	**SQ**	**d_{15}**	**d_{22}**	**d_{8}**	**d_{7}**	**d_{3}**	**d_{15}**	**d_{4}**	**d_{9}**				
d_{1}	1 1	0.13 0.06	0.13 0.06	0.12 0.05	0.12 0.04	0.12 0.03	0.11 0.03	0.10 0.02	0.09 0.01				
Q2		**d_{6}**	**d_{8}**	**d_{3}**	**d_{21}**	**d_{7}**	**d_{9}**						
d_{4}	1 1	0.19 0.09	0.13 0.06	0.10 0.05	0.10 0.05	0.107 0.029	0.075 0.024						
Q3		**d_{6}**	**d_{7}**	**d_{22}**	**d_{8}**	**d_{21}**	**d_{3}**	**d_{4}**	**d_{14}**	**d_{16}**	**d_{13}**	**d_{10}**	**d_{12}**
d_{5}	1 1	0.23 0.08	0.16 0.07	0.15 0.06	0.13 0.06	0.127 0.053	0.115 0.053	0.11 0.05	0.113 0.043	0.10 0.04	0.10 0.03	0.07 0.03	0.06 0.023
Q4		**d_{6}**	**d_{5}**	**d_{8}**	**d_{15}**	**d_{14}**	**d_{2}**	**d_{3}**	**d_{4}**	**d_{11}**			
d_{7}	1 1	0.27 0.07	0.16 0.07	0.14 0.06	0.13 0.06	0.116 0.058	0.115 0.038	0.11 0.03	0.107 0.029`	0.098 0.030			
Q5		**d_{16}**	**d_{7}**	**d_{21}**	**d_{5}**	**d_{4}**	**d_{3}**	**d_{2}**	**d_{9}**	**d_{11}**	**d_{12}**		
d_{8}	1 1	0.169 0.078	0.14 0.06	0.14 0.06	0.13 0.06	0.130 0.062	0.114 0.052	0.10 0.04	0.092 0.034	0.088 0.030	0.08 0.02		

The following abbreviations are used in Table 2. Q_1, Q_2, Q_3, Q_4, Q_5 represents the queries which are different documents respectively d_1, d_4, d_5, d_7, d_8. The values 1 in second column *SQ* are the score of each query itself, d_{15}, d_{22}, ..., *etc.* in bold are documents identifications. The similarity scores are from columns 3 where there are two lines in each column. The first line represents the similarity score between query and original document in bold in same column and the second line is the score between query and document image of the original document. To detect near duplicate documents with our method is faster than the method that uses original document dues to the use of small size of each document during the comparison. Both methods calculate similarity by considering intersection of common terms share by both documents on the total terms of both documents using Jaccard. According to the queries in table 2, our method give almost the same ranking order but there are some swaps on queries 2 and 4 where d_9 comes before d_7 with respectively scores 0.0247 and 0.0291 for the query 1. The second swapping was on query 4, where d11 with 0.0309 comes before d4 with score 0.0291. As is seen on Figure 2 there is a small difference between two methods on similarity score of two documents due to the swap of documents in Q_1 and Q_4.

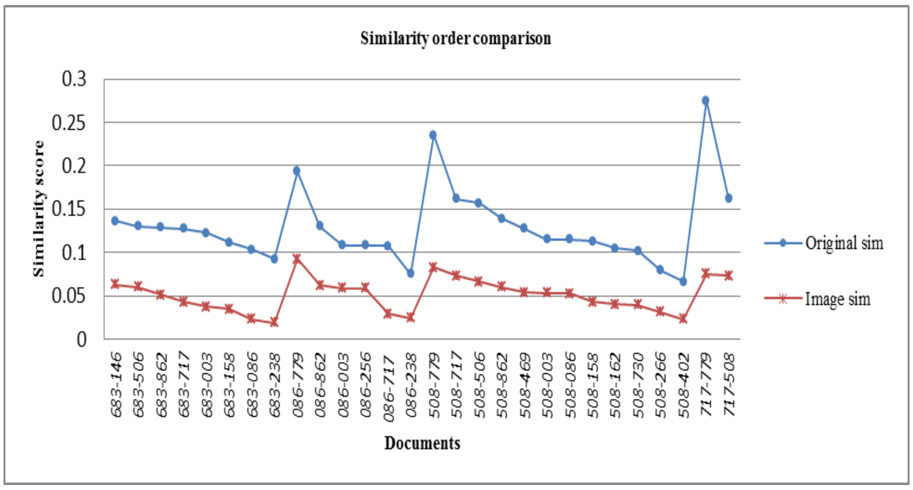

Figure 2. Similarity Comparison Order Using Original and *DI* Documents

Running time of proposed method is low which imply efficiency as is shown on Figure 3.

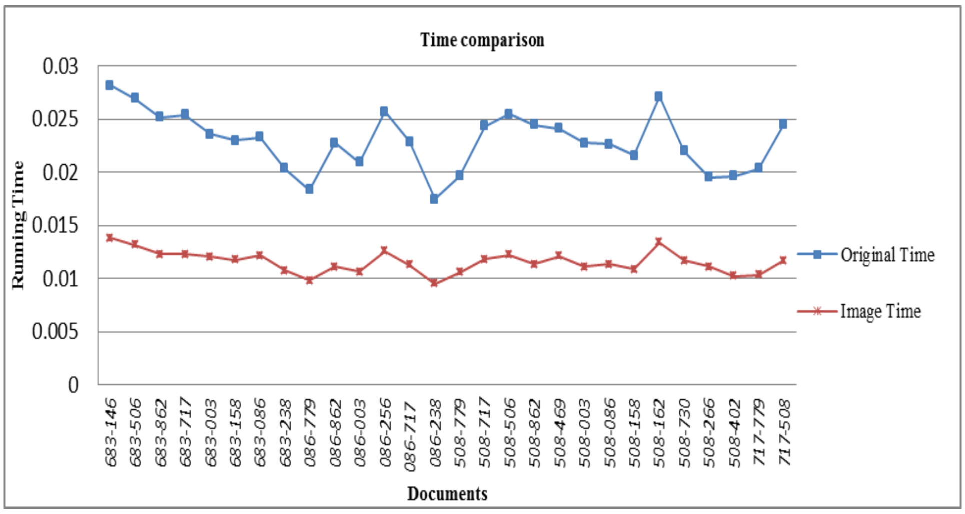

Figure 3. Time Comparison Between Using Original and *DI* Documents

5. Conclusion

Existence and weight of term plays a great role in feature selection and terms have different importance in a document. In this paper, we proposed an efficient method for near duplicate document detection by considering the importance of term based on its weight in documents. With this method we shown that extracting important terms in document plays a great role in document comparison by allowing the use of small size of document in order to know how they are similar each other. In our experiments we show that our method gives the same order of relevance based on same similarity as the order obtained using the method that considers all terms of document. The tf-idf method has been used to identify most important terms from document based on their weight; only terms with high score that fit the fixed threshold are selected to be candidate to be feature. By representing each document by extracted terms with high weight as an image of document facilitates getting similarity efficiently than using traditional method that considers all terms in a document to appear during comparison process. We realized that it is possible to get the suitable minimum threshold size of document *i.e.* image of a document also called documents representation. That can be used during comparison and maintain the same ranking order as original documents during comparison. In our experiments, our method gives the same order of relevance based on same similarity as the order obtained using the traditional method. The results show the high efficiency by reducing document size and leads to the decreased space and computation time.

The future work will be concentrated to the more robust and accurate methods for near duplicate documents detection. To increase the effectiveness and efficiency, the methods for feature extraction will be improved and fingerprinting methods can be also suggested for documents dimensionality reduction. For feature selection, the combination of tf.idf method and other features selection method can be used to maximize the number of features in a document. Not only simple statistic methods can be used but also the methods based on machine learning approach, SVM. With SVM, a set of feature functions that take a document as input and produce feature value will be used for documents representation. Our method should consider the fingerprinting of features extracted from documents for documents dimensionality reduction. The future method must capture the maximum features from documents.

Acknowledgements

Project supported the National Natural Science Foundation of China (Grant No. 61379109, M1321007) and Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20120162110077).We would like to thank the anonymous referees for their helpful comments and suggestions

References

[1] R. Gupta, “Query Based Duplicate Data Detection”, on WWW (IJCSE) International Journal on Computer Science and Engineering vol. 2, no. 4, **(2010)**, pp. 1395-1400.
[2] V. E. and P. Rosso, “Detection of near-duplicate user generated contents: the SMS spam collection”, Proceedings of the 3rd international workshop on search and mining user-generated contents, **(2011)**, pp. 27-34.
[3] Y. S. Lin, T.-Y. Liao and S.-J. Lee, “Detecting near-duplicate documents using sentence-level features and supervised learning. Expert System”, Appl., 2013, vol. 40, no. 5, pp. 1467-1476.
[4] Z. Zuping and U. Gaudence, “Efficient Algorithm for Near Duplicate Documents Detection”, International Journal of Computer Science Issues, vol. 10, no. 2, **(2013)**, pp. 12-18.
[5] A. Z. Broder, S. C. Glassman, M. S. Manasse and G. Zweig, “Syntactic clustering of the Web, Computer Networks”, vol. 29, no. 8, 13, **(1997)**, pp. 1157-1166.
[6] F. D. Manasse and M. Najork, “On the evolution of clusters of near-duplicate web pages”, In Proceedings of the first Latin American Web Congress (LAWeb), **(2003)**, pp. 3745.
[7] A. Z. Broder, “Identifying and filtering near-duplicate documents”, In COM, **(2000)**, pp. 1-10.
[8] M. S.Charikar, “Similarity estimation techniques from rounding algorithms”, In Proceedings of 34th Annual ACM Symposium on Theory of Computing, (Montreal, Quebec, Canada, **(2002)**, pp. 380-388.
[9] M. Henzinger, “Finding near-duplicate Web pages: large-scale evaluation of algorithms”, In SIGIR, **(2006)**, pp. 284-291.
[10] N. Heintze, “Scalable document fingerprinting”, Proceedings of the 2nd USENIX Workshop on Electronic Commerce 216, **(1996)**, pp. 191-200.
[11] C. Xiao, W. Wang, X. Lin and J. X. Yu, “Efficient Similarity Join for Near Duplicate Detection”, Beijing, China, **(2008)**, pp. 131-140.
[12] L. P. S. Gauch and M. Speretta, “Document similarity Based on Concept Tree Distance”, Proceedings of Nineteeth ACM conference on Hypertext and Hypermedia, Pitteburgh, PA, USA, **(2008)**, pp. 127-132.
[13] W. K. Hon, R. Shab and S. B. Wu, “Efficiency Index for Retrieving Top-k. Most frequent Documents”, Journal on Discreet Algorithms, vol. 8, no. 4, **(2010)**, pp. 402-417.
[14] K. Sadakane, “Succinct representations of lcp information and improvements in the compressed suffix arrays”, in: Proceedings of Symposium on Discrete Algorithms, **(2002)**, pp. 225-232.
[15] Y. H. Kerner, Z. Gross and A. Masa, “Automatic extraction and learning of keyphrases from scientific articles”, in Computational Linguistics and Intelligent Text Processing, vol. 3406, **(2005)**, pp. 657-669.

[16] H. Hajishirzi, Y. W. Tau and A. Kocz, “Adaptive Near-Duplicate Detection via Similarity Learning”, Proceedings of SIGIR, vol. 10, **(2010)**, pp. 10-17.