Research on the construction and filter method of stop-word list in text preprocessing

Zhou Yao
Science and Technology on Information System Engineering Laboratory
National University of Defense Technology
Changsha, China
zhou720yao@163.com

Cao Ze-wen
Science and Technology on Information System Engineering Laboratory
National University of Defense Technology
Changsha, China
zwcao1016@hotmail.com

Abstract — **In the text preprocessing of text mining, a stop-word list is constructed to filter the segment results of the text documents so that the dimensionality of the text feature space can be cut down primarily. This paper summarized the definition, extraction principles and method of stop-word, and constructed a customizing Chinese-English stop-word list with the classical stop-word list based on the difference of text documents' domain. Three different filter algorithms were designed and implemented in the process of the stop-word filter and their efficiency was compared emphatically. The experiment indicated that the hash-filter method was the fastest.**

*Key word**: text mining; text preprocessing; stop-word list; stop-word filter; hash algorithm***

I. INTRODUCTION

Text Mining(TM), also called Knowledge Discovery in Texts(KDT), is defined as the nontrivial extraction of implicit, previously unknown, and potentially useful information from given text data. The general process of TM is showed in Fig. 1.

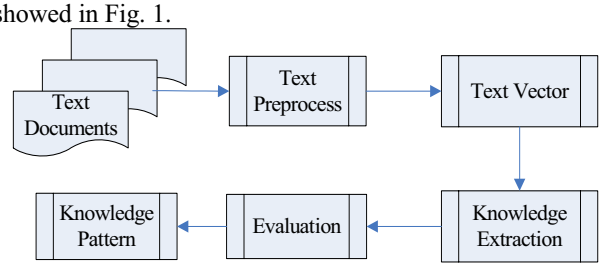

Figure1. Process of Text Mining

Helena et al, who held a viewpoint that the preprocessing phase is crucial to the efficiency of the process, applied the data mining technology in the preprocessed texts. In their paper, punctuation marks and structural information often need to be handled separately. Some of them may be ignored entirely, and some of them may require special treatment. Generally, text preprocessing consists of text input, word segment and stop-word filter, which can require as much as 80 percent of the total effort. After the segment and filter, the dimensionality of the text feature vector can be reduced a lot so that the processing effort needed in the discovery phase can be decreased greatly. Yang and Pedersen[3] hold that there is no negative effect if the text feature space is filtered by the first ten stop-words which are in descending order by document frequency and little negative effect if the text feature space is filtered by the first 100 stop-words. CatarinaSilva[4] has validated that the accuracy of the text categorization based on SVM is improved after the stop-word filter. Accordingly, the construction of stop-word list and the design of filter algorithm are crucial to the results of the text mining.

II. STOP-WORD SURVEY

A. Stop-word

The concept stop-word was first used in the information retrieval (IR) system. Two related facts were noticed in the early days of IR by Luhn[5]. First, a relatively small number of words account for a very significant fraction of all text's size. Words like IT, AND ,THE and To can be found in virtually every sentence in English-based documents. Secondly, these words make very poor index terms, with which users are indeed unlikely to ask for documents. And during Luhn's successive research, he labeled the words present in the region of highest frequency as noise word or common word and finally named them stop-word, which are too commonly used to discriminate between keyword terms and non-keyword terms and even can be ignored[6].

In IR, a stop-word list is used in the automatic indexing to filter out the words having no real purpose in describing document contents and a search using one of these terms is likely to retrieve almost every item in a database regardless of its relevance, so their discrimination value is low[7]. Furthermore, these words make up a large fraction of the text of most documents: the ten most frequently occurring words in English typically account for 20 to 30 percent of the tokens in a document[8], which is quite a large fraction of the document. Eliminating such words from consideration early in automatic indexing speeds processing, saves huge amounts of space in indexes, and does not damage retrieval effectiveness[9]. Tsz-Wai Lo,et al[10] proposed a definition that words in a document that are frequently occurring but meaningless in terms of IR are called stop-word. It is repeatedly claimed in their article that stop-word does not contribute towards the context or information of the documents and they should be removed during indexing as well as before querying by an IR system.

978-0-7695-4353-6/11 $26.00 © 2011 IEEE
DOI 10.1109/ICICTA.2011.64

Generally, stop-word means high frequency and low discrimination and should be filtered out in the IR system. In the same way, the concept of stop-word in the text mining is similar, but the ability to characterize a document is attached much more importance to judge a word is a stop-word or not. As the extraction of a document's feature vector is the preliminary condition for knowledge discovery, it is necessary to delete all the stop-words. When coming to the processing of stop-word in IR, it is not the best to extend the stop-word list as large as possible, but on the opposite, commercial services tend to be more conservative and limit the size of the stop-word list to increase the recall rate. As to the text mining, all the word identified as stop-word should be filtered out to improve the efficiency and accuracy.

B. Principle and method of stop-word identification

Stop-word can not characterize the text. So theoretically, the words meeting this condition are judged as stop-word. But we can not directly put the condition as the judgment principle, and especially in the automatic identification of stop-word, the computer programs can not tell whether one word characterizes the text intelligently. So the only principle is that the efficiency and accuracy of the text mining is improved or not after the stop-words are filtered out, which mainly contains two aspects:

- The accuracy of text mining should not be decreased if the stop-words were deleted.
- The dimensionality of the text feature space should be reduced if the stop-words were deleted.

There are two ways to construct a stop-word list including artificial pattern and automatic extraction which obey the above two principles. In Luhn's research[6], he tried to resolve the problem of stop-word using the notion of the 'word's resolving power' – the ability to distinguish one article from the rest of articles in a data set. In fact, he defined a threshold below or above which could be labeled as stop-word to determine the set of significant words in his document collection. In 1979, Van Rijsbergen[7] published the second edition of his book- Information Retrieval where he suggests a classic list of 250wtop-words in English which is often used by default or as a baseline in test database. Fox[13] formed a stop-word list of 421 words derived from the Brown corpus of 1,014,000 words drawn from a broad range of literature in English and then he expanded it to 425 in [9]. These two stop-word lists from Luhn and Fox are so classic and are used abroad. In the construction of Chinese artificial stop-word list, the Information Retrieval Laboratory of HIT in China suggested a stop-word list based on their *《Thesaurus-Extended Edition》*. But as a whole, there is no such a Chinese stop-word list admitted and used abroad until now.

In the automatic extraction of stop-word list, the most commonly used method is to calculate the term frequency, where high frequency is correlative with high noise. Besides, there are some other traditional methods based on the document frequency, the entropy calculation and so on. Gu et al proposed a new method which is to calculate the probability that the word occurs in each sentence of corpus, and calculate the probability that the sentence include the word occurring in corpus, then calculate the entropy of these probabilities, and select stop-words according to the entropy. And the method accords with people's cognizance and is better than the traditional methods validated by experiments. Tsz-Wai Lo et al [10]presented another approach called term-based random sampling based on the Kullback-Leibler divergence measure. Zhou et al[15] applied programme flows control to eliminate the single Chinese word, pure English words, number and Chinese words containing English letter or maths symbol from the original text vector. Automated stop-word list construction is undoubtedly the future of stop-word list[16] and has many advantages, including the fact that it can work on any language-which neither the analyst nor the preset stop-word list can.

Artificial construction of stop-word list is easy and not current comparatively, and it does not have the pertinency to the certain text documents when it works . Oppositely, the automatic extraction of stop-word list is potential and flexible, and it can construct an appropriate stop-word list for different domain of the text documents, but how to design an efficient and accurate algorithm is still a complicated problem.

III. CONSTRUCTION OF STOP-WORD LIST AND THE FILTER METHOD

A. Construction of Chinese-English stop-word list

The Chinese-English stop-word list in the paper is artificial based on the preset stop-word list. The user-defined stop-word list can be obtained by merging the classical stop-word list with the stop-words depended on the different domain of the text document corpus. The concrete customizing principles are displayed as follow:

- Fox's classic stop-word list and the stop-word list used in SMART;
- Chinese stop-word list（From Harbin Institute of Technology in China）
- Punctuations(DBC case and SBC case);
- Arabic numerals 0-9, the capital of 0-9 , hundred and thousand written in Chinese;
- Partial Greece letters commonly used;
- Partial math symbols;
- Words too commonly used in the specific domain(Depend on the difference of the domain);

From the above steps, a customizing Chinese-English stop-word list containing 1289 words is constructed.

B. stop-word filter algorithm

It is required to filter all the terms generated by the word segment during the text preprocessing, so it is significant to

design an efficient algorithm to improve the performance of text mining.

If the stop-word list is saved in the disk, it needs reading from the disk for the filter of the segment results of each text document, which costs lots of time. But the stop-word list containing 1289 words is only 7.90KB, which is minor to the computer memory 2-4G nowadays. It is acceptable to read the stop-word list into the memory once and all the operations are done in the memory which will improve the efficiency.

Besides this, some other measures are taken into consideration to improve the efficiency further. Here are three filter algorithms:

Sequence-Filter: the basic idea is to get a term from the word segment linked list *TERMS* and compare it with each stop-word in the stop-word linked list *STOPLIST*. As is showed in Fig. 2:

Input: stop-word linked list *STOPLIST*, Term linked list *TERMS={T_i}*
Output: *TERMSFILTER={TF_i}*

SeqFilter（STOPLIST,TERMS）：

（1） Get one term T_i from *TERMS*;
（2） Get one stop-word $Stop_j$ from *STOPLIST*;
（3） T_i== $Stop_j$？：
If yes, T_i is stop-word, and delete it. Switch to (5);
（4） *STOPLIST* is empty?；
If yes, T_i is not stop-word, and save it in *TERMSFILTER*;
If no, switch to (2);
（5） *TERMS* is empty?：
If no, switch to (1);
（6） Return *TERMSFILTER*. Filter ends.

Figure 2 Sequence-Filter Algorithm

The Most Recently Used-Filter(MRU-Filter): the basic idea is to get a term from the word segment linked list *TERMS* and compare it with each stop-word in the stop-word linked list *STOPLIST*. If it is a stop-word, insert it into the head of the *STOPLIST* and delete it from its foregoing position. As is showed in Fig. 3:

Input: stop-word linked list *STOPLIST*, Term linked list *TERMS={T_i}*
Output: results linked list *TERMSFILTER={TF_i}*

MRUFilter（STOPLIST,TERMS）：

（1） Get a T_i from *TERMS*;
（2） Get a $Stop_j$ from *STOPLIST*;
（3） T_i == $Stop_j$?：
If yes, T_i is stop-word. Delete T_i, insert $Stop_j$ into the head of *STOPLIST* and delete $Stop_j$ in its foregoing position. Switch to(5);
（4） *STOPLIST* is empty?:
If yes, T_i is not stop-word and save it in the *TERMSFILTER*;
If no, switch to(2);
（5） *TERMS* is empty?:
If no, switch to(1);
（6） Return *TERMSFILTER*. Filter ends.

Figure 3 MRU-Filter Algorithm

Hash-Filter: the basic idea is to choose a hash function F which is used to calculate the hash code for each stop-word in the list and return the remainder which is the index of the stop-word in the hash-table modulo the hash-table size. A solution to the hash collision is also set up. When it comes to filter, the index of a term is calculated by hash function F. The search is performed directly using the term's index in the Hash-Filter. If it finds the term already present, the term is a stop-word; if not, it returns NULL. Here the hash function F is BKDR Hash Function[17] and the stop-word will be inserted into the head of the linked list when there is hash collision. As is showed in Fig. 4 and Fig. 5:

Input: stop-word linked list *STOPLIST, SEED, SIZE* of the Hashtable
Output: *HASHTABLE*

HashStoplist（*STOPLIST, SEED, SIZE*）：

（1）Get a $Stop_j$ from *STOPLIST*;
（2）Calculate the *Hashcode*: $Stop_j$[0]*$SEED$^(n-1) + $Stop_j$[1]*$SEED$^(n-2) + ... + $Stop_j$[n-1]=*Hashcode*, and put a value *V* mapping the $Stop_j$;（^ is exponential operation ）
（3）Calculate *Hashcode mod Size*, and get *Index*;
（4）Check the *Hashtable* at *Index*:
If NULL, save *($Stop_j$, V, Hashcode, Next=null)* at *HASHTABLE (Index);*
If FULL, save *($Stop_j$, V, Hashcode，Next=previousNode)* at the linked list started from *HASHTABLE (Index);*
（5）*STOPLIST* is empty?:
If no, switch to（1）;
（6）Return *HASHTABLE*.

Fig 4 HASHTABLE Construction Algorithm

Input: Term linked list *TERMS={T_i}, HASHTABLE*
Output: results linked list *TERMSLIST={TF_i}*

HashFilter（*TERMS,HASHTABLE*）：

（1）Get T_i from *TERMS*;
（2）Calculate the hashcode: T_i*[0]*$Seed$^(n-1) + T_i*[1]*$Seed$^(n-2) + ... + T_i*[n-1]*=hashcode;（^ is exponential operation）
（3） Calculate *Hashcode mod Size*, and get *Index*;
（4）Search the linked list in *HASHTABLE* at *Index*:
If existed, T_i is stop-word, and delete it;
If not existed, save T_i in *TERMSFILTER*;
（5）*TERMS* is empty?:
If no, switch to(2);
（6）Return *TERMSFILTER*.

Fig 5 Hash-Filter Algorithm

The flow diagram of the Hash-Filter is showed as follow in Fig. 6:

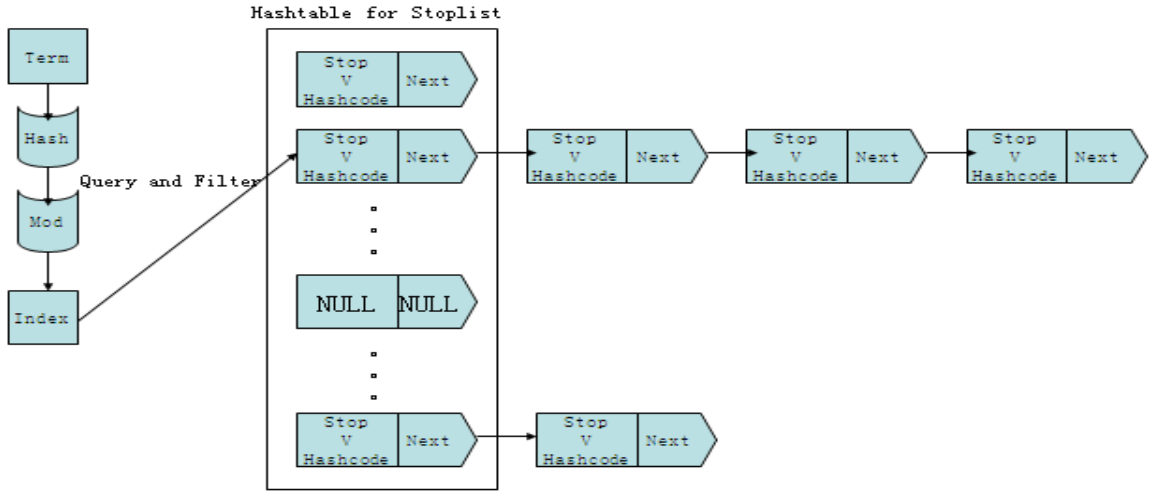

Fig 6 Hash-Filter Diagram

IV. ALGORITHM ANALYSIS AND EXPERIMENT RESULT

A. *Algorithm Analysis*

Assumed that there are N terms to be filtered, N1 of which are nonstop-words and N2 are stop-words. The stop-word list is made up of M stop-words.

Compare times of Sequence-Filter: Comp1=N1*M+ N2*M/2.（M/2 is the average compare time for stop-words）

Compare times of MRU-Filter: Comp2=N1*M+ N2*M'.（M' is the average compare time for stop-words when applying MRU-Filter for filter, and M'<M/2）

Compare times of Hash-Filter: Comp3=N*1=N.

From the analysis above, it is obvious that Comp3<Comp2<Comp1. So the Hash-Filter has the lowest complexity and the MRU-Filter does more compares than Sequence-Filter, but at the same time, the insertion and deletion operations in MRU-Filter should also be considered.

B. *Experiment Results*

In this article, the 204 text documents in Chinese with English abstract are from the training and test text corpora of FUDAN University. The hardware for our experiment is CPU Pentium（R） Dual E2200 2.20GHz, memory DDR2 2.0G and HD 150G. The results are displayed as follows:

(a) Single text's filter performance compare

Fig.7 shows that when filtering every single text document with three different algorithms, the time consuming varies a lot from each other, where the hash-filter algorithm does best, the MRU-filter algorithm is secondary and the sequence-filter algorithm behaves worst. Fig.8 shows that the compare times by using MRU-filter algorithm are apparently less than the compare times of sequence-filter algorithm, which interprets the MRU-filter's better performance.

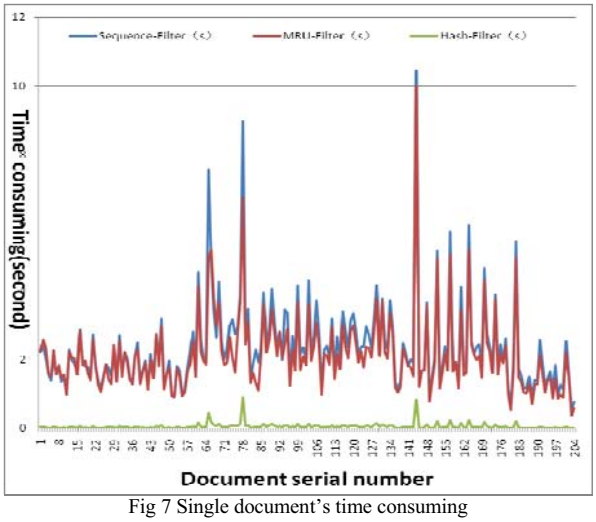

Fig 7 Single document's time consuming

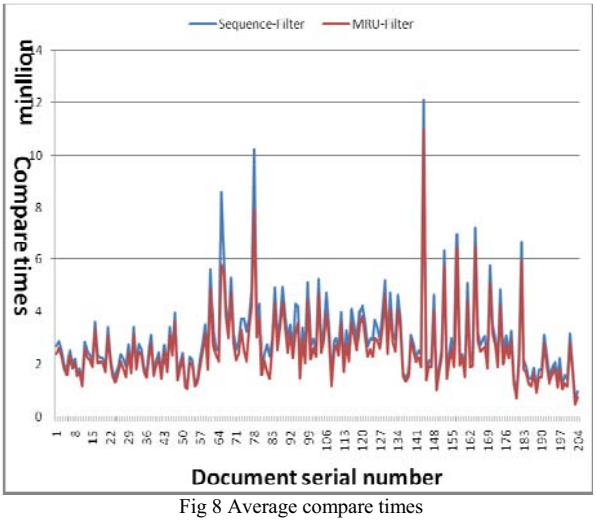

Fig 8 Average compare times

(b) Average efficiency of stop-word filter algorithms

From table 1, we can know that when the hash-filter algorithm is applied, the efficiency can be improved a lot, which is in accord with its low complexity. And at the meantime, although the average compare times decrease 371,656 between suquence-filter and MRU-filter, only 0.229second is saved in the average time consuming. It lies in that since the compare is done in the memory consuming little time, the insertion and deletion operations cost some extra time, which has a negative effect on MRU-filter algorithm's performance improvement.

TABLE 1 Performance compare over the three algorithms

	Sequence-Filter	**MRU-Filter**	**Difference**	**Hash-Filter**
Entire time consuming	477.863s	431.859s	46.004s	11.705s
Average time consuming	2.342s	2.113s	0.229s	0.057s
Entire compare times	594,877,231	519,059,591	75,817,640	//
Average compare times	2,916,065	2,544,409	371,656	//

V. CONCLUSION

Stop-word filter is a very important step to decrease the dimensionality of the text vector in the text preprocessing, which has a great effect on the efficiency of the text mining. As the stop-word list changes more or less with the different domain of the text documents to deal with and the universal stop-word list is generally unacquirable, in the context of this article, we propose a customizing stop-word list which consists of classic stop-word lists and some other newly constructed lists and can be modified based on the domain of the text documents. It is much more efficient to use the hash algorithm to filter the stop-words and the superiority over time will be disclosed apparently especially when it deals with the massive text documents.

ACKNOWLEDGMENT

We thank Doc. D.F. Hou for insightful discussions and comments on earlier draft. We are grateful to Master X. Chen from HIT for his generosity in supplying us the Chinese stop-word list.

REFERENCES

[1] Feldman R, Dagan I. Knowledge Discovery in Textual Databases(KDT)[C].In: Proceedings of the 1st Annual Conference on Knowledge Discovery and Data Mining,1995:112—117.

[2] Helena A., Oskari H., Mika K. and Inkeri V.A. Applying Data Mining Techniques in Text Analysis[J],1997(2):4-8.

[3] Yang Y., Pedersen J O. A Comparative Study on Feature Selection in Text Categorization[A].Proceedings of ICML-97,14th International Conference on Machine Learning[C]. San Francisco: Morgan Kaufmann Publishers Inc.,1997.412-420.

[4] Silva C, Ribeiro B. The Importance of Stop Word Removal on Recall Values in Text Categorization[J].Neural NetWorks,2003,3:20-24.

[5] Luhn H.P. A Statistical Approach to Mechanized Encoding and Searching of Literary Information. IBM Journal of Research and Development, 1957, 1(4):309-317.

[6] [6]Luhn,H.P. The Automatic Creation of Literature Abstracts.IBM Journal, April, presented at IRE National Convention, New York, 1958, March 24.

[7] VAN RIJSBERGEN, C. J. Information Retrieval[M]. London: Butterworths. 1975

[8] FRANCIS, W., and KUCERA, H. Frequency Analysis of English Usage. Journal of English Linguistic. 1982,18(1):64-70.

[9] Fox C. Lexical analysis and stop-lists. Information retrieval: Data structures and algorithms[M].Upper Saddle River, New Jersey: Prentice Hall,1992.

[10] Tsz-Wai Lo,et al. Automatically Building a stopwords list for an information retrieval system. Proceeding of the 5th Duth-belgian Information Retrieval Workshop. Utrecht, the Neterlands.2005,141-148.

[11] Ljiljana D and Jacques S. When Stopword Lists Make the Difference. JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE AND TECHNOLOGY, 2010, 61(1):200–203.

[12] C.X.Cui. Research on the Effect of Stop Words Selection onText CategOrization[J].JOURNAL OF TAIYUAN NORMAL UNIVERSITY(Natural Science Edition),DEC,2008,VOL.7(4).pp.91-93(In Chinese)

[13] Fox C. A Stop List for General Text. ACM-SIGIR Forum, 1990, VOL.24.pp. 19–35.

[14] Y.J.Gu,X.Z.Fan,J.H.Wang,T.Wang,and W.J.Huang.Automatic Selection of Chinese Stoplist[J].Transaction of Beijing Institute of Technology.2005.VOL.25(04).pp.337-340.(In Chinese)

[15] Q.Q.Zhou,B.D.Sun and Y. Wang. Study on New Pretreatment Method for Chinese Text Classofication System[J].APPLICATION RESEARCH OF COMPUTER.2005.VOL.22(2).pp.85-86(In Chinese)

[16] Blanchard A. Understanding and customizing stopword lists for enhanced patent mapping. World Patent Information, 2007, 29:308-316.

[17] Brian W . Kernighan and Dennis M. Ritchie. The C Programming Language. Prentice-Hall. 1988: Chapter 6-Structures.