Effective Dimension Reduction Techniques for Text Documents

P. Ponmuthuramalingam[1] and T. Devi[2]

[1]Department of Computer Science, Government Arts College, Coimbatore, India
[2]School of Computer Science and Engineering, Bharathiar University, Coimbatore, India

Summary
Frequent term based text clustering is a text clustering technique, which uses frequent term set and dramatically decreases the dimensionality of the document vector space, thus especially addressing: very high dimensionality of the data and very large size of the databases. Frequent Term based Clustering algorithm (FTC) has shown significant efficiency comparing to some well known text clustering methods, but the quality of clustering still needs further enhancement. In this paper, the morphological variant words, stopwords and grammatical words are identified and removed for further dimension reduction. Two effective dimension reduction algorithms, improved stemming and frequent term generation algorithms have been presented. An experiment on classical text documents as well as on web documents demonstrates that the developed algorithms yield good dimension reduction.

Key words:
Dimension reduction, Latent semantic, Information retrieval, Text representation, Text documents.

1. Introduction

Every day, people encounter a huge amount of information and store or represent it as data, for further analysis and management. As more text documents are stored in large database, it becomes a huge challenge to understand hidden patterns or relations in the data [3][5][11]. As text data is not in numerical format, it cannot be analysed with statistical methods [8].

Text mining is the process of finding interesting patterns in text data and often involves datasets with large number of terms. Dimension reduction selects frequent terms in the dataset prior to perform Text mining and it is important for the accuracy of further analysis as well as for the performance. As the redundant and infrequent terms could mislead the analysis, it not only increases the complexity of the analysis, and also degrades the accuracy of the result [2][4][6]. For instance, clustering techniques, which partition entities into groups with a minimum level of homogeneity within a cluster, may produce inaccurate results. Dimension reduction improves the performance of clustering techniques by reducing dimensions so that text mining procedures process data with a reduced number of terms [10][14].

Manuscript received July 5, 2010
Manuscript revised July 20, 2010

The conventional dimension reduction techniques are not easily applied to text mining application directly (i.e., in a manner that enables automatic reduction) because they often require 'a priori' domain knowledge and/or arcane analysis methodologies that are not well understood by end users [4][7].To overcome these limitations an improved stemming with frequent term generation approach has been adopted in this paper.

2. Review of Literature

Fung B.C.M. et al. (2003) addressed the problem of poor clustering accuracy due to the incorrect estimation of the number of clusters. Frequent Item set based Hierarchical Clustering (FIHC) algorithm has been proposed by Fung B.C.M. and it makes use of frequent item set and construction of a hierarchical topic tree from the clusters. A frequent item set is being used as preliminary step and the dimension of each document is drastically reduced, which in turn increases efficiency and scalability [6][8][15]. The author performed the experiment on a Pentium III 667 MHz PC with largest datasets (Reuters) and the presented algorithm is more scalable because the experiment with 10000 documents shows that FIHC algorithm completes its whole process within two minutes while Unweighted Pair Group Method with Arithmetic Mean (UPGMA) and Hierarchical Frequent Term based Clustering (HFTC) could not even produce a clustering solution [8][13].

Bi-secting k-means generates relatively deep hierarchies and hence not suitable for browsing [4][13]. The other frequent item set based algorithm HFTC [4] provides a relatively flat hierarchy but its different branches of hierarchy decrease the accuracy. FIHC uses sibling merging method and overcomes the problem and it gets higher accuracy in browsing [7].

3. Stemming

3.1 Description

Information Retrieval (IR) is essentially a matter of deciding which document in a collection should be

retrieved to satisfy a user's need for information [11]. The user's information need is represented by a query or profile, and contains one or more search terms, plus perhaps some additional information. The words that appear in documents and in queries often have many morphological variants. Thus, pairs of terms such as "computing" and "computation" will not be recognised as equivalent without some form of *Natural Language Processing* (NLP) [9].

In most cases, morphological variants of words have similar semantic interpretations and can be considered as equivalent for the purpose of IR applications. For this reason, a number of so-called stemming algorithms or stemmers have been developed, which attempt to reduce a word to its stem or root form [12]. *Stemmer* is a program or algorithm which determines the morphological root of a given inflected word form, involves the suffix removal. An algorithm which attempts to convert a word to its linguistically correct root is sometimes called a *lemmatizer*.

The 'rules' for removing a suffix will be given in the form

(condition) S1 → S2

where S1 denotes suffix string of a word before stemming and S2 denotes the replacement string of word after stemming. The above rule implies that if a word ends with the suffix S1 and the stem before S1 satisfies the given condition, S1 is replaced by S2.

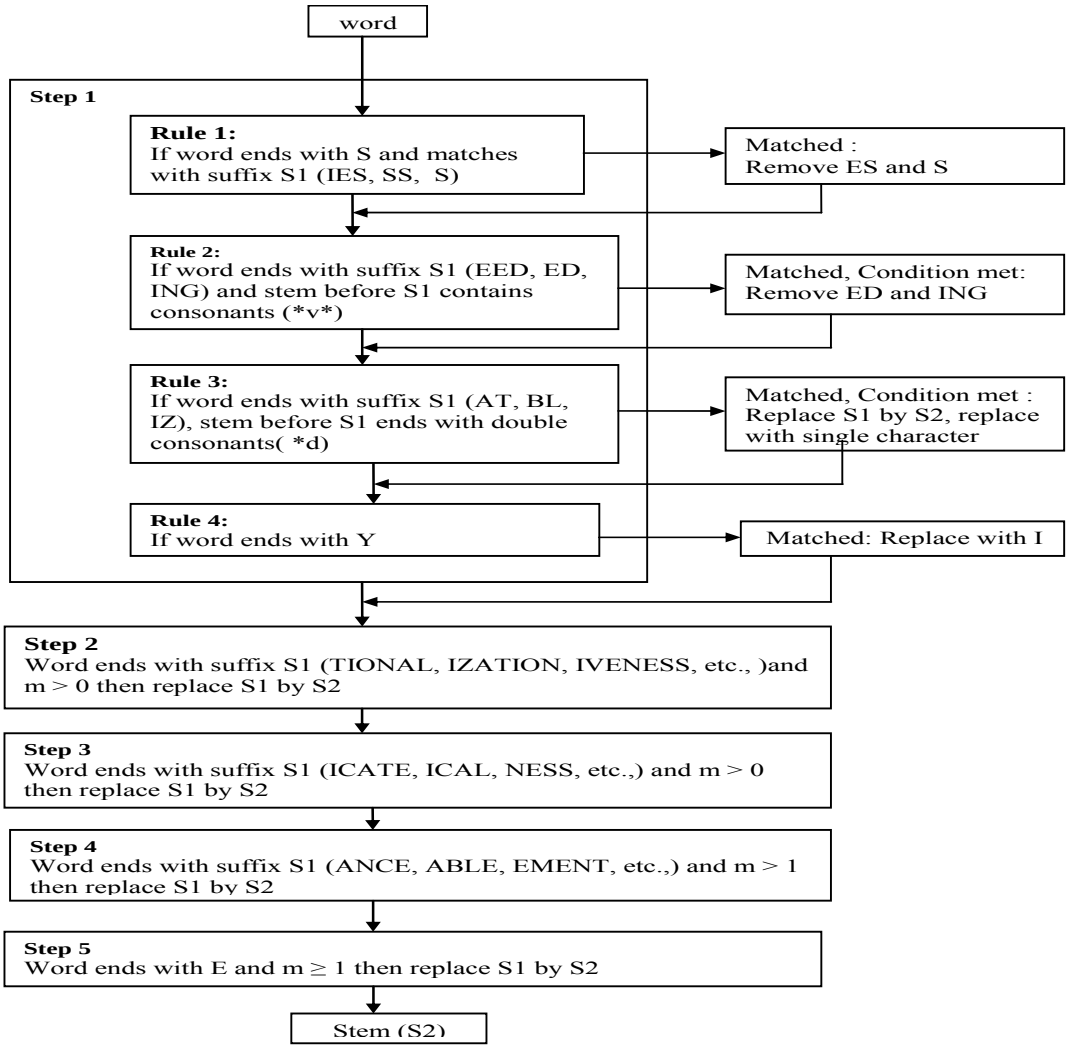

Figure 1 Steps involved in stemming algorithm

*S1 – suffix string to be tested

*S2 – replacement string after satisfying the condition
A word is represented in the form of vowel (V) and consonant(C), a measure m denotes the number of times the vowel and consonant sequence (VC) is repeated in a word. The condition part may also contain the following:
*S - the stem ends with S (and similarly for the other letters).
v - the stem contain a vowel.
*d - the stem ends with a double consonant (E.g. –TT, -SS).
*o - the stem ends CVC, where the second c is not W, X or Y (E. g. –WIO, -HOP).
and the condition part may also contain expressions with \and\, \or\ and \not\, so that
(m>1 and *S or *T)
tests for a stem with m>1 ending in S or T, while
(*d and not *L or *S or *Z)
tests for a stem ending with a double consonant other than L, S or Z.

3.2 Algorithm

The Porter Stemmer is a conflation stemmer developed by Martin Porter at the University of Cambridge in 1980. The stemmer is a linear step stemmer and it has five steps applying rules within each step [12]. Within each step, if a suffix rule matched a word, then the conditions attached to that rule are tested on what would be the resulting stem, if that suffix is removed, in the way defined by the rule.

The first step of the algorithm is designed to deal with past participles and plurals. This step is the most complex and is separated into four parts in the original definition. The first part deals with plurals and removes ES and S in the word. The second part removes ED and ING. The third part continues only if ED or ING is removed and transforms the remaining stem to ensure that certain suffixes are recognised and if ends with double consonant then replace with single consonant. The fourth part simply transforms a terminal "y" to an "i" and this part is inserted as step 2 in the flowchart (figure 1).

The remaining steps are relatively straightforward and contain rules to deal with different order classes of suffixes, initially transforming double suffixes to a single suffix and then removing suffixes, provided the relevant conditions are met.

3.3 Assumptions

In general, documents are clustered, based on context matching or similarity. Mostly, the contexts of documents are represented by nouns. Based on this, the following assumptions have been made in document dimension reduction:-

- Elimination of words which possess less than 3 characters
- Elimination of general words (stopwords)
- Elimination of adverbs and adjectives
- Elimination of non-noun verbs

Stopwords, words which are used frequently in the database and are not searchable. Most search engines do not consider common words in order to save disk space or to speed up the search result. These filtered words are known as stopwords. A list of 429 common words is eliminated as stopwords by using the improved stemming algorithm. The stopwords list is obtained from the following link
http://www.lextek.com/manuals/onix/stopwords.html
An adverb modifies a verb and it helps to tell "how", "when", or "where", the action took place. A list of 8854 words is eliminated as
adverb by the improved stemming algorithm. The links to obtain adverb and verbs are
http://www.esldesk.com/vocabulary/adverbs
http://www.englishclub.com/vocabulary/regular-verbs-list.htm
An adjective is a word that describes or modifies a noun or pronoun. There are 1814 adjectives in the list. The link to obtain adjectives is
http://www.esldesk.com/vocabulary/adjectives
The following assumptions have been made to achieve frequent term generation:

- For small document, each line is treated as a record
- For large document, each paragraph is treated as a record.

3.4 An Illustrative Sample

Consider a small sample text document for illustration, which consists of 4 lines having 44 words.

Source File Content:

> *The purpose of system study is to maximize our profit. Our profits are maximized by selling more products. In order to sell more products, the product prize must be less and reliable. The net profit is decided by the net sales over a period.*

Total number of words: 44
Number of Stopwords: 26
Number of Adverb, Adjectives and Verbs: 18

The stemmed and improved stemming algorithms have been tested for the following three test cases.

Stemming Output:
Test Case 1: Stemming alone

In this process the suffixes are alone removed to form root word (suffix stripping). The output of the execution of the improved stemming algorithm is as follows:-

The purpose of system studi is to maxim our profit our profit ar maxim by sell more product in order to sell more product the product prize must be less and reliabl the net profit is decid by the net sale over a period

Total number of words: 44

First the basic Porter's algorithm is chosen, which was downloaded from http://tartarus.org/~martin/PorterStemmer/index-old.html. The corresponding stemmed output consists of 44 words with only suffixes are stripped off for the conflated words.

Improved Stemming Output:
Test Case 2: Stemming with Stopword

Along with suffix stripping the word which are used frequently and carry no useful information about the context are removed as stopwords. The corresponding output is shown below.

Purpose system studi profit profit sell product order sell product product prize reliabl net profit decid net period

Total number of words: 18

In the above output, the Porter's algorithm has been modified to eliminate the stopwords like "I", "am", "will", "do", "you", etc.,. The stemmed output is reduced to 18 words from the original 44 words, giving 59.09% ((44-18)/44*100) dimension reduction.

Test Case 3: Stemming with Stopword, Adverbs, Adjectives,Verbs

In this study, along with test cases 1 and 2, the grammatical words, adverbs, adjectives and verbs are removed. Then the output consists of the following

product product product reliabl period

Total number of Words: 5

In addition, the algorithm is modified to eliminate stopwords as well as the grammatical words of verbs, adverbs and adjectives listed by the standard linguistic list (link shown in 3.3) and the number of stemmed words now becomes 5, giving a reduction of 88.64% ((44-5)/44*100).

4. Intermediate Term Generation

The aim of Intermediate Term Generation (ITG) is to generate index code for each stemmed document. The stemmed terms are arranged in the form of records or lines as in the given document are the seed for this procedure and are called Stemmed Record List. The procedure includes the following steps:

- Preparation of term list – the terms are arranged in ascending order to assign an index number for every term.
- Preparation of term index code – the terms in the record list are replaced by the corresponding index number for frequent term generation.

ITG Procedures

The objective of the procedure GenerateTermList is to prepare TermList which are stored in ascending order to assign index value. In step 3, the stemmed terms are added from StemmedRecordList to a list called TermListArray, where the redundant terms are stored only once. In step 6, the terms are sorted in ascending order of *terms*.

```
Input: StemmedRecordList
Output: TermListArray
GenerateTermList( )
1. Initialise no_of_terms = 0
2. For each term in StemmedRecordList
3.   Add term to TermListArray
4.   Increment no_of_terms by 1
5.  Next loop (term)
6. Sort TermListArray in ascending order of terms
7. return (TermListArray)
```

Figure 2 Procedure for generation of TermList

The main function of generation of term index procedure is to assign index value for each record term list. In step 3 and 4, for each term in the StemmedRecordList, the corresponding index value is obtained from TermListArray and stored in TermIndexrecord file in row major order. Each TermIndexrecord is sorted in index order in step 6.

```
Input: TermListArray, StemmedRecordList
Output : TermIndexrecord
TermIndexGeneration( )
1.  For each record r in StemmedRecordList
2.    For each term t in record
3.       Get the index of the term in TermListArray
4.       Write index into TermIndexrecord
5.     Next loop (t)
6.   Sort the TermIndexrecord based on index
7.  Next loop (r)
8. return (TermIndexrecord)
```

Figure 3 Procedure for generation of TermIndex

An Illustration

Stemmed File:

purpos system studi profit profit sell product order sell product product prize reliabl net profit decid net period

StemmedRecord List:

1. *purpos system studi profit*
2. *profit sell product*
3. *order sell product product prize reliabl*
4. *net profit decid net period*

TermListArray:

0	1	2	3	4
decid	*net*	*order*	*period*	*prize*

5	6	7	8	9
product	*profit*	*purpose*	*reliabl*	*sell*

10	11
studi	*system*

TermIndexrecord:

6 7 10 11
5 6 9
2 4 5 8 9
0 1 3 6

5. Binary Code Conversion

One-to-one Mapping Procedure

A one-to-one mapping between Index value and binary code has been performed. It matches and replaces a binary value of either 1 or 0 (text present or absent) for every indexed term in the TermIndexrecord. The procedure insert a binary 1, where the occurrence of the term in the index code value position, otherwise a binary 0, and the previous index code values have been filled with binary zero.

```
Input: TermIndexrecord
Output: Binary Code
BinaryCodeGeneration( )
1.  For each record r in termIndex_file
2.    Sort record r in ascending order.
3.    Initialise prevIndex=0
4.    For each Index in TermIndexrecord
5.        Insert (Index-prevIndex-1) 0's to binarycoded file
6.        Insert 1's to binarycode
7.    Assign prevIndex=Index
8.    Next loop (Index)
9.  Next loop (r)
```

Figure 4 Procedure for Binary Code Conversion

The index terms are stored in ascending order of index value in step 2. In step 5, binary zeros are inserted upto the previous index position of the current index. Then, a binary 1 is inserted in the current index position.

An Illustration

Consider the following term index code as input, arranged in the record forms.

TermIndexrecord:
6 7 10 11
5 6 9
2 4 5 8 9
0 1 3 6

The corresponding binary codes are mapped as follows:

Binary Code:
0 0 0 0 0 0 1 1 0 0 1 1
0 0 0 0 0 1 1 0 0 1 0 0
0 0 1 0 1 1 0 0 1 1 0 0
1 1 0 1 0 0 1 0 0 0 0 0

6. Frequent Term Set Generations Descriptions

Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent item sets. Apriori employs an iterative approach known as level-wise search, where k-item sets are used to explore (k+1)-item sets. First, the set of frequent 1-item sets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted L_1. Next, L_1 is used to find L_2, the set of frequent 2-itemsets, which is used to find L_3, and so on, until no more frequent k-item sets, is found. The finding of each L_k requires one full scan of the database.

To improve the efficiency of the level-wise generation of frequent item sets, an important property called the Apriori property, is used to reduce the search space [7].

Apriori Property: All non-empty subsets of a frequent item set must also be frequent. This property belongs to a special category of properties called antimonotone, in the sense that if a set cannot pass a test all of its supersets will fail the same test as well. A two step process is followed, consisting of join and prune actions.

Algorithm for discovering large item sets make multiple passes over the data. In the first pass, it count the support of individual items and determine which of them are large, i.e., have minimum supports. Each subsequent pass starts with a seed set of item sets found to be large in the previous pass. This seed set is used for generating new potentially large item sets, called candidate item sets, and count the actual support for these candidate item sets during the pass over the data. At the end of the pass, it determines which of the candidate item sets are actually large, and they become the seed for the next pass. This process continues until no new large item sets are found.

The Apriori algorithms [1][7] generate the candidate item sets to be counted in a pass by using only the item sets found large in the previous pass, without considering the documents in the database. The basic intuition is that any subset of a large item set must be large. Therefore, the candidate item sets having k items can be generated by joining large item sets having k-1 items, and deleting those that contain any subset that is not large. This procedure results in generation of a much smaller number of candidate item sets.

An Illustration

Consider the above binary coded file as input. The corresponding output of frequent term set generation algorithm for 20% and 60% minimum support is given below.

Frequent Item Set (Minimum Support: 20%):

[*decid, net, order, period, prize, product, profit, purpos, reliabl, sell, studi, system*]

Frequent Item Set (Minimum Support: 60%):

[product, profit, sell]

When the minimum support is 20%, the number of terms is reduced from 18 to 12 and thus giving 33.33% ((18-12)/18*100) reduction. When the minimum support is 60%, the number of terms are reduced to 3 and gives a reduction of 83.33% ((18-3)18*100).

7. Results and Discussions

Table 1 represents the memory reduction in terms of size of stemming algorithm. The corpus data samples S1 to S4 of size 250 KB to 20000 KB are used to test the developed stemming algorithm for the test cases 1 to 3. The size reduction of case 3 outperforms actual size and other cases. Case 1 represents stemming (suffix stripping), case 2 represents Stemming with general and stopwords and case 3 represents Stemming with general and grammar words. It is observed that case 3, stemming with general and grammar words, would perform better than other cases.

Table 1 Memory reduction for different data sets

Data Sets	Actual Size (KB)	After Stemming			Frequent term sets		
		Case 1	Case 2	Case 3	Case 11	Case 21	Case 31
S1	249	186	135	78	107	81	41
S2	2999	2110	1527	819	64.6	47	24.4
S3	4567	3287	2416	1666	1303	793	570
S4	20275	14851	11300	7081	141	1418	1161

S1- Reuters Transcribed Subset - 200 files- 249 KB
S2- PDF-54 files-2999 KB
S3- Mini-News group-2000 files- 4567 KB
S4- Reuters-21578-22 files- 20275 KB

The reduction chart Figure 5 shows that case 3 algorithm performs better than case1 and case 2.

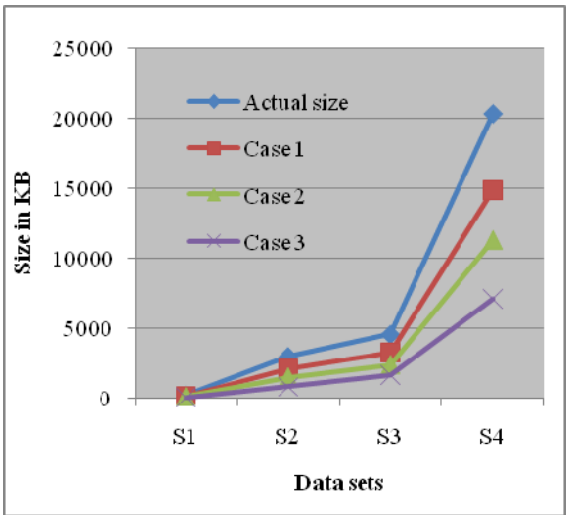

Figure 5 Size reduction chart of improved stemming algorithm

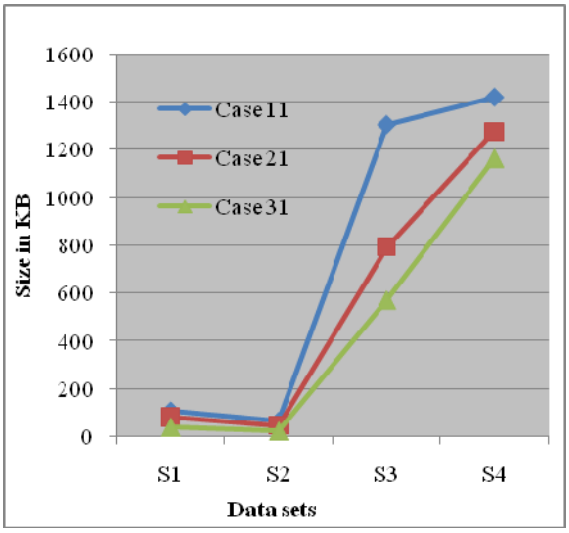

Figure 6 Size reduction chart of frequent term set generation algorithm

The test cases 11, 21 and 31 with a uniform minimum support of 10% are applied to stemmed output of case 1, 2 and 3 respectively of the original samples S1 to S4. Case 11 represents frequent term set for case 1, case 21 represents frequent term set for case 2 and case 31 represents frequent term set for case 3.The reduction chart in Figure 6 shows that case 31 performs better than case 11 and 21. Similarly, Table 2 represents the words reduction of stemming algorithm. The words reduction of case 3 outperforms actual size and other cases.

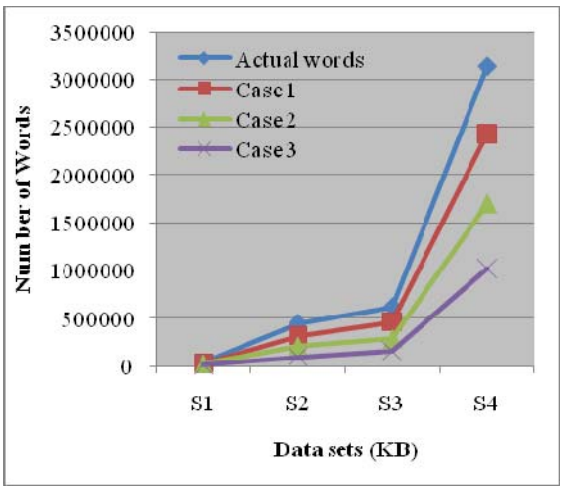

Figure 7 Words reduction chart of improved stemming algorithm

Table 2 Word Reduction for different data sets

Data Sets	Actual Number of Words	After Stemming			Frequent term sets		
		Case 1	Case 2	Case 3	Case 11	Case 21	Case 31
S1	40489	32076	21672	12367	14888	10520	5042
S2	448489	320661	205098	97648	7775	5196	2470
S3	620546	463183	283960	160209	137872	1892	41793
S4	3133193	2427765	1703670	1018807	144432	123158	108770

The words reduction chart of three cases is shown in Figure 7. The reduction chart shows that case 3 algorithm performs better than case 1 and case 2. The words reduction chart of three cases is shown in Figure 8. The reduction chart shows that case 31 performs better than case 11 and 21.

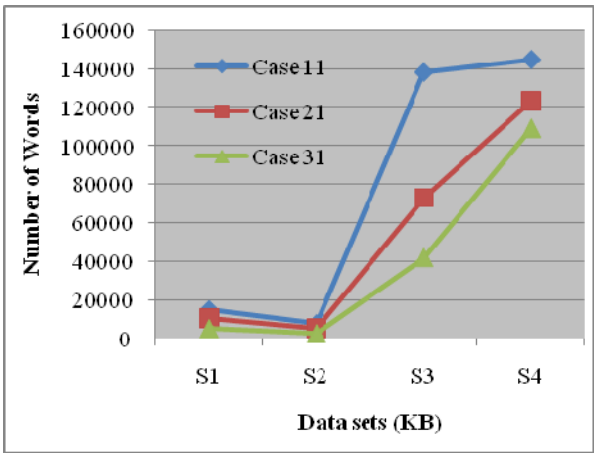

Figure 8 Words reduction chart of frequent term set generation algorithm

8. Conclusions

Dimension reduction improves the performance of text mining techniques to process the data with a reduced number of terms. In this work, two improved dimension reduction algorithms namely stemming and frequent term generations are developed and tested. The morphological variant in the given document, stopping words and grammatical words are identified and removed by stemming algorithms. The algorithms have been tested for three test cases and the improved stemming algorithm yields good results. The procedure TermList and TermIndexGeneration are used to generate sorted term list and its index value, which are used to generate the binary code value for the reduced terms. The frequent term set generation algorithm when applied on the output generated by the modified stemming algorithm yields good results.

References

[1]Agrawal R., and Srikant R., Fast algorithm for mining association rules, Proceedings of 20th International Conference on Very Large Data Bases, VLDB 94, Santiago de Chile, Chile, 1994, pp. 487-499.

[2]Ahonen-Myka H., Mining all Maximal Frequent Word Sequences in a Set of Sentences, Proceedings of the 14th ACM International Conference on Information and Knowledge Management, 2005, pp. 255-256.

[3]Allan J., HARD Track Overview in TREC High Accuracy Retrieval from Documents, Proceedings of the 12th Text Retrieval Conference, 2003, pp. 24-37.

[4]Beil F., Ester M. and Xu X., Frequent Term-based Text Clustering, Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 436-442.

[5]Frakes B. and Baeza-Yates R., Information Retrieval: Data Structures and Algorithms. Englewood Cliffs, N.J.: Prentice Hall, 1992.

[6]Fung B. C. M., Wang K. and Ester M., Hierarchical Document Clustering using Frequent Item sets, Proceedings of SIAM International Conference on Data Mining, 2003, pp. 59-70.

[7]Han J., Kamber M., Data Mining: Concepts and Techniques, Morgan Kaufmann (Elsevier), 2006.

[8]Han J., Pei J. and Yin Y., Mining frequent patterns without candidate generation, Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'00), Dallas, Texa, USA, May 2000.

[9]Lin D. and Pantel P., Discovering Word Senses from Text , Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 615-619.

[10]Liu X. and He P., A study on text clustering algorithms based on frequent term sets, Proceedings of the first international conference on advanced data mining and applications, 2005, pp.347-354.

[11]Manning C.D., Raghavan P. and Schutze H., Introduction to Information Retrieval, Cambridge University Press, Cambridge, UK, 2008.

[12]Porter M.F., An Algorithm for Suffix Stripping, Program, Vol. 14, no. 3, pp. 130-137, 1980.

[13]Steinbach M., Karypsis G. and Kumar V., A Comparison of Document Clustering Techniques, KDD-2000 Workshop on Text Mining, 2000, pp. 203-215.

[14]Thammi Reddy K., Shashi M. and Pratap Reddy L., Hybrid Clustering Approach for Concept Generation, International Journal of Computer Science and Network Security (IJCSNS), VOL. 7 NO.4, April 2007.
[15]Yanjun Li, Congnan Luo, Soon M. Chung, Text clustering with feature selection by using Statistical Data, IEEE Transactions on Knowledge and Data Engineering, 2008. Vol.20, pp. 641-652.

P.Ponmuthuramalingam received his Masters Degree in Computer Science from Alagappa University,Karaikudi in 1988 and the M.Phil in Computer Science from Bharathidasan University, Tiruchirapalli. He is working as Associate Professor in Computer Science, Government Arts College, Coimbatore since 1989. His research interest includes Text mining, Semantic Web, Network Security and Parallel Algorithms .

T.Devi received the Master of Computer Applications from P.S.G. College of Technology, Coimbatore in 1987 and Ph.D from the University of Warwick, United Kingdom in 1998. She is presently heading Department of Computer Application, School of Computer Science and Engineering, Bharathiar University, Coimbatore. Prior to joining Bharathiar University, she was an Associate Professor in Indian Institute of Foreign Trade, New Delhi. Her current research centered on the Software Engineering, Product Introduction, Technical Process Management and Concurrent Engineering. She has contributed more than 60 papers in various National / International / conference/ Seminars /Symposia.