Progress Report #1 for Comparing performance of Text classification algorithms on various fraction of sequential text of each document in the data sets

By Yen Hoang, Anbang Xu, Taewoo Kim

1. Topic change Announcement [Anbang]

We have changed our focusing problem from unsupervised “Clustering” to supervised “Classification” problem by considering the Professor’s comments. For more information, please refer to the Appendix A. In this progress report, we will discuss what we have found about the data set so far and about future plans that we have on our mind.

2. About the Data Set [Table 1 - Yen, Figure 1 - Anbang, Table 2 and Figure 2 - Taewoo]

We are dealing with one of text categorization corpora - Reuters-21578 collection. For 115 categories, it has separate training set and test set. Here are details what we found about the data set. It is interesting fact to note that there is at least one training file for each category. However, there are no files in 25 categories of test set.

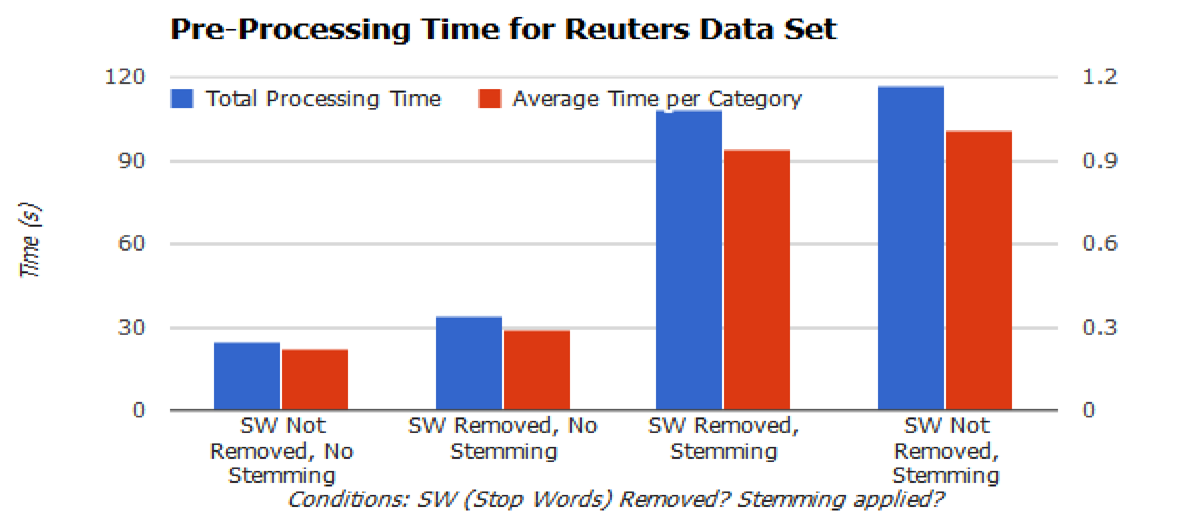
Table 1. [By Yen] - Overview of the Reuters Data Set

	Training Set	Test Set
Number of Category	115	115
Number of Total Files	9,646	3,744
Total Size	9,072.6KB (=8.86MB)	3,655.7KB (=3.57MB)
Average File Size	0.94KB	0.97KB
Average Number of Files per Category	83.88	32.56
Maximum Number of Files in Category	2,877 (Category: earn)	1,087 (Category: earn)
Minimum Number of Files in Category	1 (Only 1 file in 25 Categories)	0 (No test files in 25 Categories)

For this Reuters data set, to maintain quality of text data, we have removed non Alphanumeric characters and stop words and tokenized by using whitespace. We have also applied Porter Stemming. In detail, we have used Python functions from Natural Language ToolKit (NLTK). NLTK includes 127

English stop words corpus such as i, me, myself. It also provides Porter Stemmer. These pre-processing steps took from 25 seconds to 117 seconds based on certain conditions. As we can expect, removing stop words and applying stemming takes time. If we haven't removed stop words and not applied stemming, it only took 25 seconds to pre-process. On the contrary, if we have removed stop words and applied stemming, it took 117 seconds to pre-process the data set. Figure 1 clearly shows this. Even though it takes about four times longer, we have decided to remove stop words and apply stemming since it is better way to maintain quality of text in the data set.

Figure 1. [By Anbang] - Pre-Processing Time for the data set under certain conditions



* left Axis (0~120): Total Processing Time. Right Axis (0~1.2): Average Processing Time

Finally, after pre-processing, we have analyzed terms and their frequencies. Interestingly, most frequent term in both data sets is 'said'. Maybe this is due to the fact that the Reuters data set is some collection of articles. Also, when we check most frequent TOP 1,000 terms in the training set, 843 terms are also found in the TOP 1,000 frequent terms in the test set. This shows us that test set contains similar contents as the training sets.

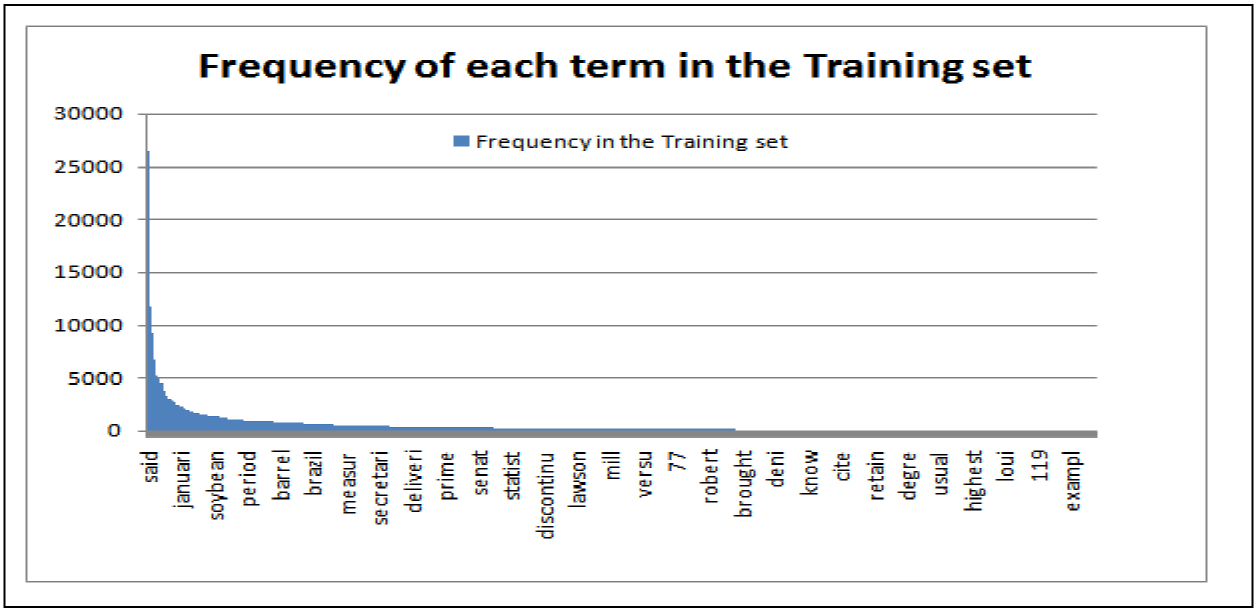
Table 2. [By Taewoo] - Term related Statistics

	Training Set	Test Set
Number of Unique Terms	20,275	13,077
Total Frequencies	1,061,742	460,673
Average occurrences of each	52.4	35.2

unique term		
Average length of Unique Terms	5.8	5.6
Number of Total Files	9,646	3,744
Average number of terms per each file	110.1	123.0
Most frequent term and it's frequency	said (26,443)	said (9,199)
Sum of frequencies for TOP 1,000 frequent terms when sorted by descending order	822,975 (77.5% of total frequencies)	361,744 (78.5% of total frequencies)

Figure 2 shows that distribution of frequency per term in the training set follows the Zipfian distribution when we order terms by descending order based on their frequency.

Figure 2. [By Taewoo] Frequency of each term in the Training set



3. Other things and Future plan [Taewoo]

As we become familiar with the data set, we are discussing about three classification algorithms Vector Space model using TF-IDF and cosine similarity, Decision Tree, and Naive Bayes. For future plan, we will fully understand these algorithms and finish coding and try existing packages provided by libraries such as scikit-learn before submitting the Progress Report 2. We also hope that we can finish executing these algorithms on the Reuters data set. After submitting the Progress Report 2, we will begin evaluate the impact of various fraction of sequential text extracted from original document when it is used as input instead of each original document by using fraction number parameter which ranges 10% to 100% to extract some fraction of sequential text in the original document.

Appendix A. Changes from our Proposal

1) Topic change [Anbang]

We have changed our topic from ‘clustering’ to ‘classification’ since we agree the feedback from the Professor and we think that it will be more easy to evaluate the output of algorithms since there is training set and test set. Now, the goal of our project is to compare performance of three text classification algorithms on the data sets which have separate training and test data. Also, we would like to evaluate the impact of document size on these classification algorithms by comparing results between using whole document and using just some fraction of sequential text in the original document.

2) Algorithms [Anbang]

Classification algorithms - Vector Space model using TF-IDF and cosine similarity, Decision Tree, and Naive Bayes - will be implemented.

3) Data Set [Yen]

We will use the text categorization data sets - Reuters-21578 collection, and Ohsumed collection since they have well sorted text categorization information and each of collection has training and test data set.

4) Software [Taewoo]

Python and some libraries needed to show plots will be used. Also, for proper processing, pre-processing of each document such as removing unnecessary letters and stop words, and applying stemming are needed. We will use libraries such as Natural Language Toolkit (NLTK).

5) Evaluation [Taewoo]

Since each document in the test sets has its own label, we can easily check the result whether each algorithm was successful or not for given test set.