Final Report

1 Dataset Reuters-21578

- Access to Reuters-21578 Distribution 1.0: https://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection
- Introduction of Reuters-21578

The Reuters-21578 dataset is one of the standard benchmarks for text categorization (TC) task. It includes 21578 news instances that appeared in the Reuters newswire in 1987, which are classified into 135 topics of categories mostly regarding business and economy. Reuters-21578 collection has several characteristics that make it widely used for TC experiment:

- ➢ It is multi-labeled. Each document may belong to more than one category.
- ➢ Some documents in the dataset do not belong to any category.
- ➢ The distribution of documents across the categories is highly skewed. Some categories have very few positive examples while others have thousands.
- ➢ There are several semantic relations among the categories but there is no explicit hierarchy or description defined on the categories.

A previous version of the Reuters collection is Reuters-22173. A revision of the collection resulted in the removal of 595 duplicates from the original set of 22173 documents, thus leaving the 21578 documents that now make Reuters-21578. There are actually 12902 documents which can be used in experiments because others have not been considered for labeling initially. In general, TC researchers use the ModApte split, in which 9603 documents are selected for training and other 3299 constitute the test set. Some of the 12902 documents have no categories attached to them. Among the 135 categories, 20 have no positive training documents in the ModApte split. In other words, 115 remaining categories have at least one positive training example each and can all theoretically be used in experiments. Three subsets of them are most popular among global researchers. The first subset is the set of 10 categories with the highest number of positive training examples, named R(10). The second subset is the set of 90 categories with at least one positive training example and one test example, named R(90). The third subset is the set of 115 categories with at least one positive training example, named R(115).

2 Literature Review

Article 1

Yang, Y., & Pedersen, J. O. (1997, July). A comparative study on feature selection in text categorization. In *ICML* (Vol. 97, pp. 412-420).

A paper of Yang & Pederson conducts a comparative study of five feature selection methods and two classifiers (1997) in text categorization. Yang & Pederson (1997) made evaluations of **term selections based on document frequency (DF), information gain (IG), mutual information (MI), an x^2-test (CHI) and term strength (TS)**. Two m-ary classifiers: **k-nearest-neighbor classifier (kNN) and linear least squares fit mapping (LLSF)** are adopted separately to assess the effectiveness of the five feature selection methods. The experiments use two data collections: Reuters-22173 which is an old version of Reuters news dataset, and OHSUMED collection. It is helpful for me to learn mathematical introduction and comparison of the typical five feature selection methods from the paper. And its experiment procedure using original

Reuters dataset can be regarded as a good reference to my project on Reuters-21578 dataset. Yang & Pederson (1997) draw a conclusion that **using IG thresholding with kNN classifier** can achieve the highest accuracy rate of text classification based on Reuters corpus, which is an essential point for verification of Reuters-21578 dataset of my project. When using Reuters-22173, only documents having at least one topic were utilized and they were divided into a training set of 9610 and a test set of 3662 documents. The mean length of words in each instance and the whole standard deviation were also computed in the experiment.

Article 2

Debole, F., & Sebastiani, F. (2005). An Analysis of the Relative Hardness of Reuters-21578 Subsets. *Journal of the American Society for Information Science & Technology*, *56*(6), 584–596. http://doi.org/10.1002/asi.20147

The paper written by Debole & Sebastiani (2005) is the most useful and practical one for my project among all the literatures. It is for beginners in text classification and has an elaborate representation and comparison between different usages of Reuters-21578 dataset. Firstly, it introduces the field of Text Categorization (TC) comprehensively, which systematically brings me preliminary knowledge of TC. In the entire TC system, there are several sections: document indexing, classifier leaning and classifier evaluation. In document indexing, feature or term selection, term weights, reduction factors are several main issues to be concerned. In classifier learning, documents are classified into the training set to build the classifier and the test set to evaluate the effectiveness of the classifier. In classifier evaluation, training efficiency (average time required to build a classifier), classification efficiency (average time required to classify a document), and effectiveness (average correctness of classification behavior) are considered measures of success for a TC system. In binary TC, effectiveness is always measured by a combination of precision and recall.

Debole & Sebastiani (2005) also introduces the history and characteristics of Reuters-21578 collection and its subsets. There are actually 12902 documents which can be used in experiments because others have not been considered for labeling initially. In general, TC researchers use the ModApte split, in which 9603 documents are selected for training and other 3299 constitute the test set. TC is conducted based on the TOPICS group including 135 categories which are primarily business and economy aspects. Some of the 12902 documents have no categories attached to them. Among the 135 categories, 20 have no positive training documents in the ModApte split. In other words, 115 remaining categories have at least one positive training example each and can all theoretically be used in experiments. However, three subsets of them are most popular among global researchers. The first subset is the set of 10 categories with the highest number of positive training examples, named R(10). The second subset is the set of 90 categories with at least one positive training example and one test example, named R(90). The third subset is the set of 115 categories with at least one positive training example, named R(115).

Debole & Sebastiani (2005) "test the relative hardness of these subsets in a variety of experimental TC contexts, generated by two different term weighting policies, three different feature selection functions, three different 'reduction factors' for feature selection, three different learning methods, and two different experimental measures, in all possible combinations" (p. 585). The descriptions of those common algorithms and models, and the whole procedure of the experiment give me a comprehensive understanding of popular notions in TC. Though I still could not comprehend the detailed operation actions and the results, it strengthened my initial impressions in Text Categorization indeed. In the paper, there is a lot to be retested and emulated for the implement of my own project.

Article 3

Rogati, M., & Yang, Y. (2002). High-performing Feature Selection for Text Classification. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management* (pp. 659–661). New York, NY, USA: ACM. http://doi.org/10.1145/584792.584911

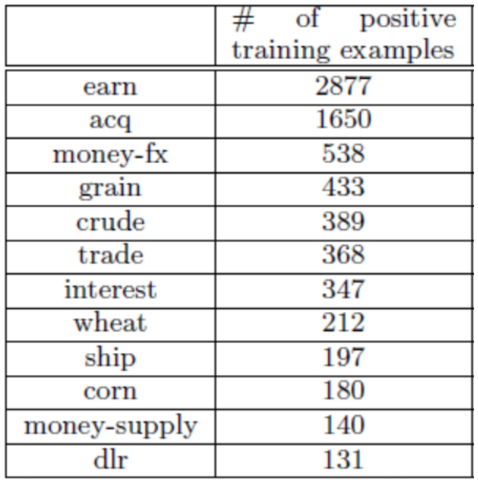
Rogati & Yang (2002) presented a study of the performance of over 100 variants of **five filter feature selection methods (document frequency (DF), information gain (IG), x^2 (CHI), mutual information, and binary version of information gain (IG2))** using two data collections (Reuters-21578 and part of RCV1) and **four classifiers (NB, Rocchio, kNN and SVM)**. They also conducted experiments on **combining** high-performing but uncorrelated feature selection methods to improve text categorization results. As a beginner with inadequate knowledge in text classification, I do not know the specific operating procedure using different feature selection methods since Rogati & Yang (2002) did not clarify it in the paper. They merely made a detailed discussion of the experiment results instead of experiment methods. Maybe the authors assume that readers already know the popular feature selection methods and classifiers. Nevertheless, the idea of implementing combination of well performed feature selection methods enlightened me to have a try on my own project.

Article 4

Zhang, J., & Yang, Y. (2003). Robustness of Regularized Linear Classification Methods in Text Categorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 190–197). New York, NY, USA: ACM. http://doi.org/10.1145/860435.860471

Zhang & Yang (2003) used Reuters-21578 dataset to make an evaluation of **robustness** of regularized linear classification methods in text categorization. In this dataset, each instance may belong to more than one category, so they split the classification problem into **multiple binary classification problems** with respect to each category, which may also be adopted for my own project. All numbers and stop words are removed, and words are converted into lowercase without stemming. Infrequent features (appear less than three times) are filtered out. Zhang & Yang (2003) chose **information gain (IG)** as the feature selection method and **binary term weighting** is applied. I am not sure what this step means and how to achieve it. Besides, words in document titles are treated distinctly to words in document bodies. Precision and recall are used to evaluate models in their combined form called F_1 (I omit the mathematical expression here).

The three linear classification methods are **SVM, ridge regression and logic regression**. I have never heard of the last two classification methods before. The paper aims to emulate real-world applications under situations of small number of features, mix-labeled documents and rare positive examples, and concludes that ridge regression may be the most promising candidate for rare class problems. This situation is not the same as the situation of my project but the rule for pulling out categories from Reuters-21578 collection can be a good reference. Zhang & Yang (2003) selected top 12 categories with the most positive training examples and set the **feature size to be 3000** for Reuters-21578.



	# of positive training examples
earn	2877
acq	1650
money-fx	538
grain	433
crude	389
trade	368
interest	347
wheat	212
ship	197
corn	180
money-supply	140
dlr	131

The top 12 categories of Reuters-21578

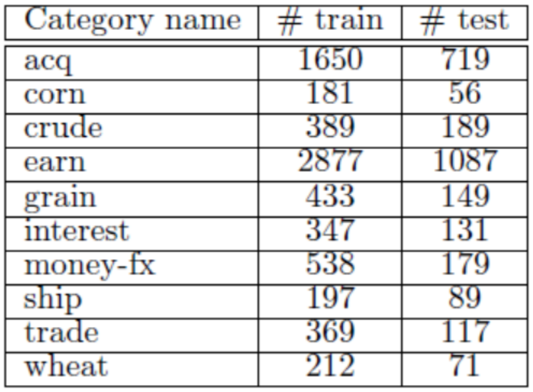
Article 5

Kassab, R., & Lamirel, J.-C. (2006). A New Approach to Intelligent Text Filtering Based on Novelty Detection. In *Proceedings of the 17th Australasian Database Conference - Volume 49* (pp. 149–156). Darlinghurst, Australia, Australia: Australian Computer Society, Inc. Retrieved from http://dl.acm.org/citation.cfm?id=1151736.1151752

Kassab & Lamirel (2006) proposed a new approach relying on a specific novelty detection model which allows both accurate learning of user's profile and evaluation of the coherency of **user's behavior during his interaction with the system**. And this approach has been successfully tested on Reuters-21578 corpus. The experimental results prove that this approach significantly outperforms the well-known Rocchio's learning algorithm.

The experimental results are reported for the set of the **top 10 categories** with the most positive training documents in Reuters-21578 dataset. Due to the high quantity of negative examples in Reuters-21578 collection, the authors have selected negative examples using the query zoning method, which I am not familiar with. They used the highly discriminating words selected by CHI feature selection method to accomplish the experiment. The **top 50 features were chosen for each category and used for forming a global description space of 379 features**. These later were used for representing both training and test documents by vectors of feature-weights which were calculated using TF weighting algorithm.

This a screenshot of the table of the top 10 categories with the most positive training instances.



Category name	# train	# test
acq	1650	719
corn	181	56
crude	389	189
earn	2877	1087
grain	433	149
interest	347	131
money-fx	538	179
ship	197	89
trade	369	117
wheat	212	71

Training and test examples of top 10 categories of Reuters-21578

Compared to the results in the paper of Zhang & Yang (2003), I find that category “trade” and “corn” both have one more training example in the paper of Kassab & Lamirel (2006). For this situation, I need to figure out on my own when preprocessing the original dataset.

Article 6

Crammer, K., & Singer, Y. (2003). A Family of Additive Online Algorithms for Category Ranking. *J. Mach. Learn. Res.*, *3*, 1025–1058.

Crammer & Singer (2003) devised a new family of topic-ranking algorithms for **multi-labeled documents**. The motivation for the algorithms stem from recent advances in online learning algorithms. In the paper, the authors provide a unified analysis of the family of algorithms in the mistake bound model, then discuss experiments with the proposed family of topic-ranking algorithms on the Reuters-21578 corpus and the new corpus released by Reuters in 2000. On both corpora, the presented algorithms achieve state-of-the-art results and outperforms topic-ranking adaptations of Rocchio’s algorithm and of the Perceptron algorithm. Likely, the algorithm design in this paper is arcane to me. So I mainly focus on the data preprocessing section and the application methods of Reuters-21578 collection in their experiments.

Crammer & Singer (2003) used the ModApte pre-processing of the corpus. All words were converted to lower-case, digits were mapped to a single token designating it is a digit, and non-alpha-numeric characters were discarded. During the experiment, the authors also used a stop-list to remove very frequent words. The number of different words left after pre-processing is 27747. After the pre-processing, the corpus contains 10,789 documents each of which is associated with one or more topics. The number of different topics in the ModApte version of Reuters-21578 is 91. Since this corpus is of comparatively small size, they used **5-fold cross validation** in the experiments and did not use the original partition into a training set and a test set. Over 90% of the documents of Reuters-21578 are associated with a single topic. This preprocessing and split method is uncommon and may not be used for my own project.

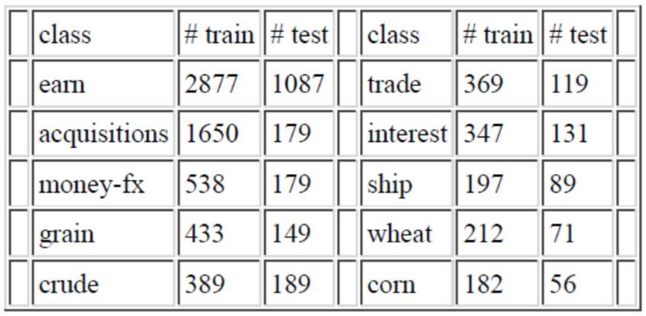
In addition, in terms of feature selection method, they used the weights of the prototypes generated by the adaptation of Rocchio’s algorithm.

Article 7

Evaluation of text classification. (n.d.). Retrieved from http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-text-classification-1.html

In this paper, 10 largest classes in Reuters-21578 collection with number of documents in training and test also have been listed.

Table 13.7: The ten largest classes in the Reuters-21578 collection with number of documents in training and test sets.



class	# train	# test	class	# train	# test
earn	2877	1087	trade	369	119
acquisitions	1650	179	interest	347	131
money-fx	538	179	ship	197	89
grain	433	149	wheat	212	71
crude	389	189	corn	182	56

The specific numbers of the listed classes are almost the same as the quantities of the above two tables mentioned. But there is still minute disparity between the three tables; and I am still wondering that.

For each of these classifiers, this paper points that researchers can measure recall, precision, and accuracy. ModApte split which comprises 9,603 training documents and 3,299 test documents is the dominant method. The distribution of documents in classes is very uneven in Reuters-21578 dataset, and some work evaluates systems on only documents in the ten largest classes. I have learned from the article that accuracy is not a good measure for small classes because always saying no, a strategy that defeats the purpose of building a classifier, will achieve high accuracy. The always-no classifier is 99% accurate for a class with relative frequency 1%. For **small classes, precision, recall and F_1 are better measures**.

I also learn that for Reuters-21578 collection, we use several binary classifiers and need to compute a single aggregate measure that **combines the measures for individual classifiers**. There are two methods for doing this. "***Macroaveraging*** computes a simple average over classes. ***Microaveraging*** pools per-document decisions across classes, and then computes an effectiveness measure on the pooled contingency table" ("Evaluation of text classification," n.d., p. 2). Therefore, macroaveraging gives equal weight to each class, whereas microaveraging gives equal weight to each per-document classification decision. In other words, **microaveraged results** are therefore a measure of effectiveness on the **large classes** in a test collection; to get a sense of effectiveness on **small classes**, researchers should compute **macroaveraged** results.

For different classifiers evaluation, the paper compared NB, Rocchio, decision trees, kNN and SVM several methods. In general, the ranking of classifiers ultimately depends on the class, the document collection, and the experimental setup.

Article 8

Appendix A, REUTERS-21578

It is an appendix of an article, but I cannot find its original intact article. The appendix is a condensed and derivative version of README file in the original Reuters-21578 dataset. The Reuters-21578 collection is distributed in 22 files. Each of the first 21 files (reut2-000.sgm through reut2-020.sgm) contain 1000 documents, while the last (reut2-021.sgm) contains 578 documents. The files are in SGML format. Rather than going into the details of the SGML language, it is described how the SGML tags are used to divide each file, and each document, into sections. The <REUTERS> and </REUTERS> tags serve to delimit documents within a file, other tags are used to delimit elements within a document. Specific format descriptions can be acquired in the README file.

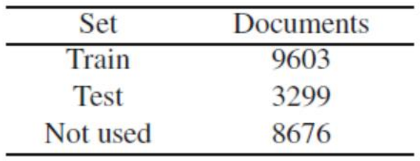
In the ModApte split:

Training Set (9,603 documents): LEWISSPLIT="TRAIN"; TOPICS="YES"

Test Set (3,299 documents): LEWISSPLIT="TEST"; TOPICS="YES"

The TOPICS="YES" stories with no topics can reasonably be considered negative examples for all 135 valid TOPICS categories. The total 12902 documents are instances that will be practically used for text categorization.

Table A.5 Used documents on ModApte split.



Set	Documents
Train	9603
Test	3299
Not used	8676

For text categorization general research, **effectiveness results can only be compared between studies that the same training and test set**. Therefore, different researchers can compare their consequences and evaluate their models only within the same distribution of training and test sets. So it is defined exactly which articles are in each of the recommended training sets and test sets by specifying the values those articles will have on the TOPICS, LEWISSPLIT, and CGISPLIT attributes of the REUTERS tags.

It is lucky that this appendix also list the stop words in the Reuters-21578 dataset. It may not be comprehensive, but it is a fairly good guidance.

Article 9

Ghamrawi, N., & McCallum, A. (2005). Collective Multi-label Classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management* (pp. 195–200). New York, NY, USA: ACM. http://doi.org/10.1145/1099554.1099591

As Ghamrawi & McCallum (2005) said, “common approaches to multi-label classification learn independent classifiers for each category, and employ ranking or thresholding schemes for classification” (p. 195). Because they do not exploit **dependencies between labels**, such techniques are only well-suited to problems in which categories are independent. However, in many circumstances, labels are highly interdependent. This paper explores **multilabel conditional random field (CRF) classification models** that directly parameterize label co-occurrences in multi-label classification. Experiments show that the models outperform their singlelabel counterparts on standard text corpora. Their experiments used two data sets that, although sparsely multi-labeled, have become standard for multi-label classification experiments: the Reuters-21578 and OHSU-Med text corpora. In the two datasets, labels are not hierarchical, and each document has at least one label from the entire label set. Experiments use corpus Reuters10 based on ModApte split. Documents belong to the 10 largest classes, which label 84% of the documents and form 39 distinct combinations of labels in the training data. Features are ranked according to their **mutual information (MI)**, so that the classifiers may select a proportion of features having the highest rank. Still, I could not truly understand its implementing methods but its idea is worth studying. Results shows that collective classifiers perform better than the traditional binary model, and that directly parameterizing these dependencies is advantageous.

Article 10

Yang, Y., & Liu, X. (1999). A Re-examination of Text Categorization Methods. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 42–49). New York, NY, USA: ACM. http://doi.org/10.1145/312624.312647

Yang & Liu (1999) reported a controlled study with statistical significance tests on five text categorization methods: the Support Vector Machines (SVM), a k-Nearest Neighbor (kNN) classifier, a neural network (NNet) approach, the Linear Least-squares Fit (LLSF) mapping and a Naive Bayes (NB) classifier. The results show that "SVM, kNN and LLSF significantly outperform NNet and NB when the number of positive training instances per category are small (less than ten), and that all the methods perform comparably when the categories are sufficiently common (over 300 instances)" (p. 42).

They also used the ApteMod version of Reuters-21578 collection. They selected the categories which have at least one document in the training set and one in the test set. This process resulted in 90 categories in both the training and test sets. After eliminating documents which do not belong to any of these **90 categories, they obtained a training set of 7769 documents, a test set of 3019 documents**, and a vocabulary 24240 unique words after stemming and stop word removal. The number of categories per document is 1.3 on average. This split method may be used for my own project.

Synthesis of Articles

Most of the selected articles use Reuters-21578 dataset as one of their experimental subjects. Debole & Sebastiani (2005) elaborated the preliminary knowledge and standard procedure for text categorization, which makes me have a clearly cognition of the basic concepts.

In the pre-processing section, almost all the articles referred tokenization, stop word removal, stemming, lowercase transformation, etc. Then during the document indexing section, there are lots of choices of feature selection, term weights and other factors. Yang & Pederson (1997) and Rogati & Yang (2002) made comprehensive comparisons between several common feature selection methods including DF, IG, MI, CHI, IG2 and TS. They demonstrated the mathematical expressions of different methods and explained both their advantages and disadvantages towards different experiment situations. On the whole, IG and CHI have comparatively good effectiveness based on Reuters-21578 dataset.

In terms of classifier learning section, different articles used different training and test set of Reuters-21578. Generally, however, ModApte split with R(10) and R(90) have been used most frequently. Since text classification of Reuters-21578 collection is a multiple binary classification problem, a variety of classification methods and algorithms are proposed aiming to solve individual aspects of the problem. On the Stanford website titled Evaluation of Text Classification, traditional classifiers like Naïve Bayes, Rocchio, decision trees, kNN and SVM are listed and compared between each other. Yang & Liu (1999) also discussed the neural network method for text categorization. Besides, Zhang & Yang (2003) made an analysis of robustness of regularized linear classification methods including SVM, ridge regression and logic regression. Kassab & Lamirel (2006) and Crammer & Singer (2003) respectively adopted advanced interacting intelligent model and a new family of algorithms to execute classification and their performances outweigh traditional single classifiers. The dependencies between labels were also considered by Ghamrawi & McCallum (2005) and they devised a novel classification model named CRF.

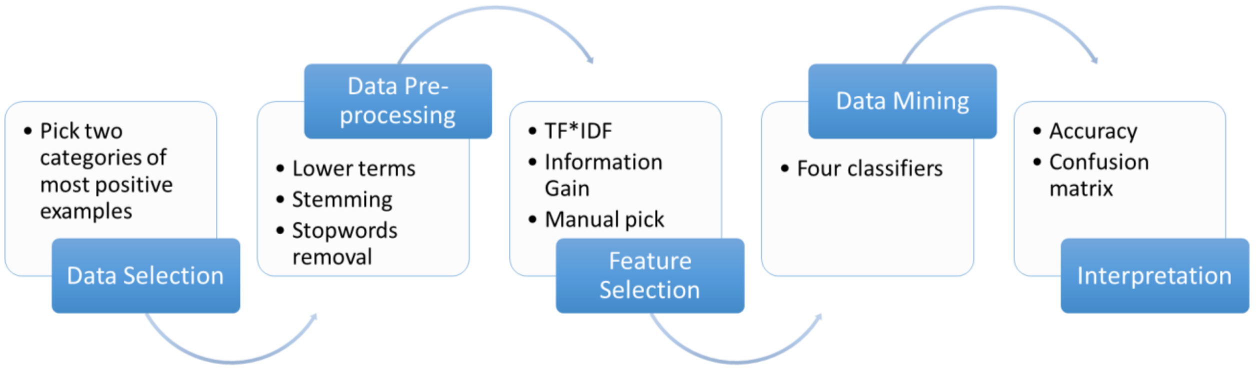
In the classifier evaluation section, training efficiency (average time for building a classifier), classification efficiency (average time for classifying a document) and effectiveness (accuracy, combination of precision and recall) are three assessment criteria. The third indicator has many detailed choices towards multi-classifiers experiment. Several selected articles mentioned their operating methods, such as macroaveraging and microaveraging.

These articles can largely help me to implement text categorization for my own project in different facets.

3 Workflow and Decisions

3.1 Workflow Overview

The workflow diagram of my news classification project is shown below. At first, it is inevitable to pick some categories as a subset from the whole 135 categories Reuters-21578 dataset, because a multi-labeled text classification project may be significantly sophisticated with too many categories. And a classification model should be built separately for each category. Then, after making the data selection, data pre-processing is the next step including tokenization, lower manipulation, stemming and stopwords removal. With well-prepared data, feature selection based on classical algorithms and manual filter can be executed to choose top effective terms for the next step. The next step is data mining using existing four classifiers (Naïve Bayes, Support Vector Machine, Decision Tree and Generalized Linear Model) in Oracle Data Miner to automatically build training and test set for the selected terms. The last step is to develop interpretation and evaluation of the performances of different feature selection methods and different classifiers. The indexes for evaluation are the overall accuracy and the appearance of the confusion matrix. It is also necessary to find which terms actually play important roles in the classification, and to explore their meanings to discover new knowledge.



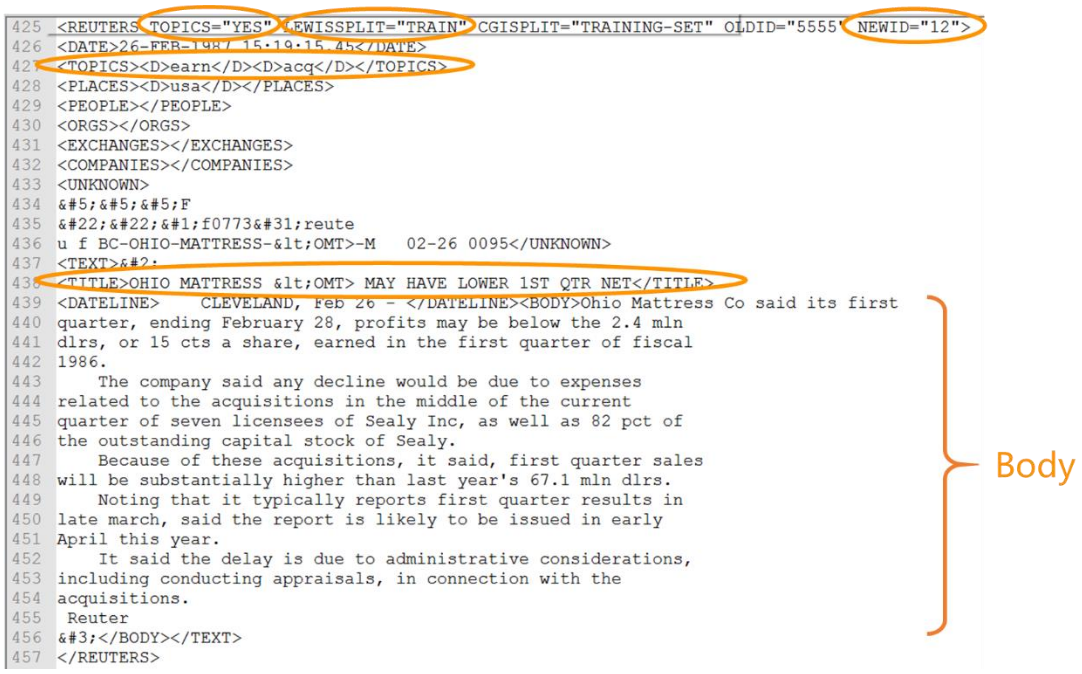
3.2 Data Selection

To simplify the classification model, two categories of news abstracts with the most positive examples are selected as my dataset for this project. One category is Mergers/Acquisitions (ACQ), and the other is Earnings and Earnings Forecasts (EARN). Instances selected could belong to either one category or both categories.

Mergers and acquisitions is a general term that refers to the consolidation of companies or assets. “A merger means a combination of two companies to form a new company, while an acquisition is the purchase of one company by another in which no new company is formed.” Terms related to ACQ could be Acquiree or Target Company, Amalgamation, Business Cycle, Circular merger, Consolidation, etc. (“Giddy: Mergers & Acquisitions Glossary,” n.d.).

Earnings "are the amount of profit that a company produces during a specific period, which is usually defined as a quarter (three calendar months) or a year. Earnings typically refer to after-tax net income." An earnings estimate "is an analyst's estimate for a company's future quarterly or annual earnings." Terms related to EARN could be Earnings Before Interest, Taxes, Depreciation, and Amortization (EBT), Exchange Rate, Growth, Industry Relative, Long Term Growth, Mean, Median, Yield, etc. ("Earnings Estimates Glossary," n.d.).

To select the news instances that satisfy the needs, I adopted some restrictions applied to Java codes. This is a screen shot of the original data format quick look and the extracted elements. The "NEWID" is regarded as instance unique ID, and the topic names have been restricted to earn or acq. I also merged the title and body part together to form the bag of words.



```
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="5555" NEWID="12">
<DATE>26-FEB-1987 15:19:15.45</DATE>
<TOPICS><D>earn</D><D>acq</D></TOPICS>
<PLACES><D>usa</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>
&#5;&#5;&#5;F
&#22;&#22;&#1;f0773&#31;reute
u f BC-OHIO-MATTRESS-&lt;OMT>-M   02-26 0095</UNKNOWN>
<TEXT>&#2;
<TITLE>OHIO MATTRESS &lt;OMT> MAY HAVE LOWER 1ST QTR NET</TITLE>
<DATELINE>    CLEVELAND, Feb 26 - </DATELINE><BODY>Ohio Mattress Co said its first
quarter, ending February 28, profits may be below the 2.4 mln
dlrs, or 15 cts a share, earned in the first quarter of fiscal
1986.
    The company said any decline would be due to expenses
related to the acquisitions in the middle of the current
quarter of seven licensees of Sealy Inc, as well as 82 pct of
the outstanding capital stock of Sealy.
    Because of these acquisitions, it said, first quarter sales
will be substantially higher than last year's 67.1 mln dlrs.
    Noting that it typically reports first quarter results in
late march, said the report is likely to be issued in early
April this year.
    It said the delay is due to administrative considerations,
including conducting appraisals, in connection with the
acquisitions.
 Reuter
&#3;</BODY></TEXT>
</REUTERS>
```

After selecting the news abstracts that actually have terms, the gold standard has been produced as follows.

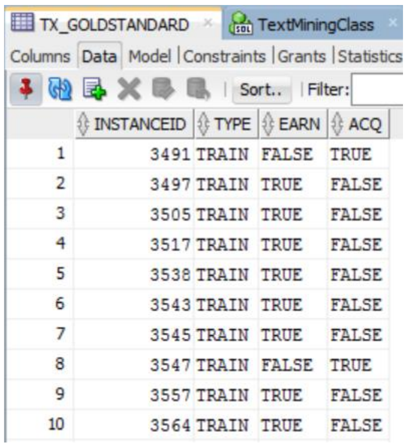
Number of instances belonging to ACQ: 2210

Number of instances belonging to EARN: 3775

Number of instances belonging to both ACQ and EARN: 18

Total number of instances: 5967

I used Java language to extract the required information for gold standard from the original data and saved as a CSV file, and then imported the CSV file to Oracle SQL Developer. The attribute Type means the original assigned type based on ModApte split. The value could be TRAIN or TEST. The attributes Earn and Acq reflect which category the instance belongs to. The value could be TRUE or FALSE. The created table TX_GOLDSTANDARD is shown below.


TX_GOLDSTANDARD TextMiningClass

Columns | Data | Model | Constraints | Grants | Statistics

Sort.. | Filter:

	INSTANCEID	TYPE	EARN	ACQ
1	3491	TRAIN	FALSE	TRUE
2	3497	TRAIN	TRUE	FALSE
3	3505	TRAIN	TRUE	FALSE
4	3517	TRAIN	TRUE	FALSE
5	3538	TRAIN	TRUE	FALSE
6	3543	TRAIN	TRUE	FALSE
7	3545	TRAIN	TRUE	FALSE
8	3547	TRAIN	FALSE	TRUE
9	3557	TRAIN	TRUE	FALSE
10	3564	TRAIN	TRUE	FALSE

3.3 Data Pre-Processing

Documents in the corpus should be subdivided into training and testing datasets. For text categorization general research, effectiveness results can only be compared between studies with the same training and test set, helping different researchers to compare their consequences and evaluate their models. So, initially I planned to choose ModApte split in which 9603 instances are selected for training and the other 3299 form the test set. And I once hoped to use the R(10) subset including 10 categories with the highest number of positive training examples for research.

However, since I have decided to choose the only top two categories for my classification project, there is no need to follow the traditional ModApte split. The split should be randomly allocated by Oracle classification model of different classifiers.

After extracting the needed instances, I did the tokenization, lower terms, stop words removal and manual stemming using Java Stanford NLP API library for each instance during the pre-processing procedure.

1) Extract text in both title and body part of each selected instance
2) Remove word “Reuter” because it appears in all the instances at the end
3) Restrict to get terms that contain at least one letter
4) Tokenization text and create table TX_TERMS
5) Lower all the terms and create table TX_TERM_LOWER
6) Create table TX_A2_STOPWORDS based on list built by Gerard Salton and Chris Buckley for the experimental SMART information retrieval system at Cornell University
7) Remove the stopwords. The total amount of words in the original list is 571, and there are 459 words in the instances matched.
8) Create table TX_TERM_LOWER_DISTINCT_NOSTOP which consists of distinct lowercase terms apart from stopwords
9) Manual stemming and the total number of terms is 20912, which is still too large
10) Remove terms that appear only in one instance and get 9835 distinct lowercase terms

3.4 Feature Selection and Data Mining

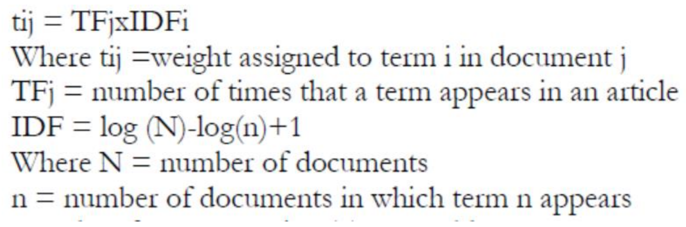
An indexing method contains two steps. First, there is a selection of what feature or term can be chosen to identify a specific category. Second, there is an algorithm to compute term weights of different terms for classification purpose. Different feature selection methods will be utilized, including information gain (IG), document frequency (DF), x^2 statistics (CHI) and mutual information (MI). For my project, I adopted TF/IDF and IG the two main methods.

For data mining, text classification of the dataset is a multi-label classification task. So it can be decomposed into two binary classification issues using binary classifiers. I built one model to predict whether it is ACQ or not, and another model to predict whether it is EARN or not. I adopted the existing four classifiers Naïve Bayes, Support Vector Machine, Decision Tree and Generalized Linear Model to make classification using Oracle Data Miner. When using the classifiers, training sets will be used to build classifiers, and test sets will be applied to evaluate the effectiveness of classifiers.

1) First try

I firstly adopted TF/IDF feature selection method. TF stands for the number of times that a term appears in each abstract. DF means the number of instances in which a term appears. TF/IDF aims to compute the value of Term frequency * Inverse document frequency.

- Create table TX_DF for the selected 9835 terms
- Create table TX_TF for the 9835 terms among the 5967 instances
- Create table TX_TFIDF for the computational results of all the terms and ordered by its value in descending sequence. The attributes of the table are Distinctterm, InstanceID and TFIDF.


tij = TFjxIDFi
Where tij =weight assigned to term i in document j
TFj = number of times that a term appears in an article
IDF = log (N)-log(n)+1
Where N = number of documents
n = number of documents in which term n appears

- Create table TX_TFIDF_TOP to identify distinct top terms that have the highest values of tf*idf and ranked in descending order. The attributes of the table are Distinctterm and MaxTFIDF.
- Select top 100 terms and create table TX_FEATURES1 to convert rows to columns
- Insert the number of appearance of terms in the instances to the table TX_FEATURES1
- Create table TX_C1 to document InstanceID, appearing times of top terms and its category

Findings of the table TX_C1:

The values of the cells are almost zero. There are unexpected words such as HK, BP, TEXAS which are not ideal for classification of ACQ and EARN categories.

Findings of the classification results:

The accuracy is around 67.66% for Acq model and around 65.97% for Earn model.

2) Second try

I adopted Information Gain feature selection method. I divided the instances into three categories:

- Category 1: only belong to ACQ, 2192 instances
- Category 2: only belong to EARN, 3757 instances
- Category 3: belong to both ACQ and EARN, 18 instances

Create several tables to compute the values of different probabilities based on the formula.

$$Information\ Gain\ (t) = -\sum_{i=1}^{\overset{m=3}{m}} \Pr(c_i)\log\Pr(c_i) + \Pr(t)\sum_{i=1}^{m}\Pr(c_i|t)\log\Pr(c_i|t) + \Pr(\bar{t})\ \sum_{i=1}^{m} Pr(c_i|\bar{t})\log\Pr(c_i|\bar{t})$$

The relevant tables are:

- TX_DF_PR1 -> Pr(t)
- TX_DF_ACQNUMBER, TX_DF_NUM_PR2, TX_DF_PR2 -> Pr(c_1|t)
- TX_DF_EARNNUMBER, TX_DF_NUM_PR3, TX_DF_PR3 -> Pr(c_2|t)
- TX_DF_BOTHNUMBER, TX_DF_NUM_PR3_5, TX_DF_PR3_5 -> Pr(c_3|t)
- TX_DF_PR4 -> Pr($\bar{t}$)
- TX_DF_PR5 -> Pr($c_1|\bar{t}$)
- TX_DF_PR6 -> Pr($c_2|\bar{t}$)
- TX_DF_PR6_5 -> Pr($c_3|\bar{t}$)

Finally, create table TX_IG to compute the value of IG for each term.

- Select top 100 terms and create table TX_FEATURES2 to convert rows to columns
- Insert the number of appearance of terms in the instances to the table TX_FEATURES2
- Create table TX_C2 to document InstanceID, appearing times of top terms and its category

Findings of the table TX_C2:

The table still looks sparse but acceptable. There are meaningless words which have been selected. They are “u.s., 1st, 2nd, 3rd, 4th”. These terms could not help make classification of ACQ and EARN categories.

Findings of the classification results:

The accuracy is around 95.12% for Acq model and around 94.42% for Earn model. It is obvious that there is a huge improvement compared to TF/TDF method.

3) Third try

- Select top 105 terms which have the highest value of IG
- Remove the 5 terms “u.s., 1st, 2nd, 3rd, 4th” manually
- Delete the five terms in the table TX_FEATURES2 and TX_C2

Findings of the modified table TX_C2:

The table still looks sparse but acceptable.

Findings of the classification results:

The accuracy for Acq and Earn almost remain the same compared with the second try.

4) Fourth try

- Select top 205 terms which have the highest value of IG
- Remove the 5 terms “u.s., 1st, 2nd, 3rd, 4th” manually
- Create table TX_FEATURES3 to convert rows to columns
- Insert the number of appearance of terms in the instances to the table TX_FEATURES3
- Create table TX_C3 to document InstanceID, appearing times of top terms and its category

Findings of the table TX_C3:

The table is too sparse for terms at the behind.

Findings of the classification results:

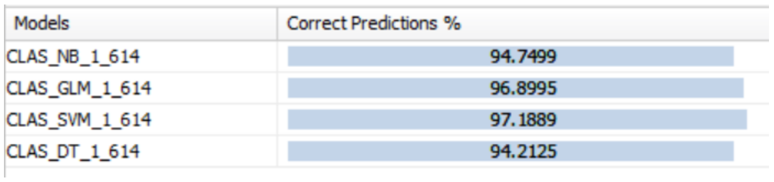
The accuracy raised 0.64% to 95.76% for Acq averagely compared to the third try

The accuracy raised 0.77% to 95.19% for Earn averagely compared to the third try.

3.5 Interpretation

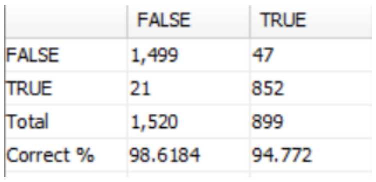
1) Best result: Fourth try

- ACQ classification



Models	Correct Predictions %
CLAS_NB_1_614	94.7499
CLAS_GLM_1_614	96.8995
CLAS_SVM_1_614	97.1889
CLAS_DT_1_614	94.2125

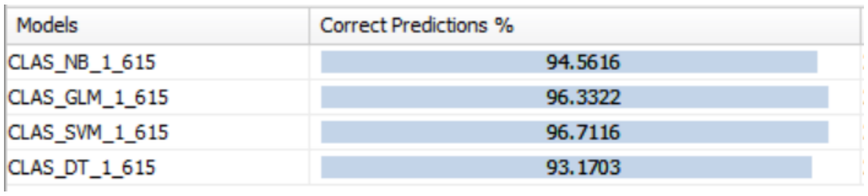
The best average accuracy for ACQ classification is 95.76%. And the best classifier is SVM.



	FALSE	TRUE
FALSE	1,499	47
TRUE	21	852
Total	1,520	899
Correct %	98.6184	94.772

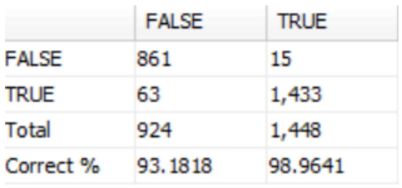
From the confusion matrix of SVM displayed above, it can be found that the accuracy of FALSE category is higher than that of TRUE category. And it is true for all the four classifiers.

- EARN classification



Models	Correct Predictions %
CLAS_NB_1_615	94.5616
CLAS_GLM_1_615	96.3322
CLAS_SVM_1_615	96.7116
CLAS_DT_1_615	93.1703

The best average accuracy for EARN classification is 95.19%. And the best classifier is SVM.



	FALSE	TRUE
FALSE	861	15
TRUE	63	1,433
Total	924	1,448
Correct %	93.1818	98.9641

From the confusion matrix of SVM displayed above, it can be found that the accuracy of TRUE category is higher than that of FALSE category. And it is true for all the four classifiers.

2) Discussion

I. **Phenomenon:** Accuracy of Acq is slightly higher than that of Earn
Explanation: Total case count of Acq (2419) is slightly higher than that of Earn (2372)

II. **Phenomenon:** SVM classifier is the best classifier for both classification models
Explanation: SVM is better for small amount of instances because small number of support vectors decide the result

III. **Phenomenon:** For confusion matrix, the accuracy of TRUE/FALSE category is higher than another for all classifiers
Explanation: Total number of instances belonging to TRUE/FALSE is higher than another

4 Knowledge Discovery

After completing the good classification models with relatively high accuracy, the analysis of the terms that actually play important roles in the models has been conducted. The comparison between words selected automatically by the classification models and words of the glossary of ACQ or EARN category is meaningful to acquire some knowledge.

4.1 Significant words in ACQ classification model

From the SVM model, words that have the highest positive coefficients for ACQ category are:

"Acquisition, merger, stake, buy, bid, sees, offer, corp, results, takeover, buys, gain, note, sell, completes, acquired, valued, held, sells, reached, acquire, income, commission…"

Among them, "corp" stands for "corporation".

Most words selected automatically are absolutely in the glossary of terms for Mergers and Acquisitions. "Acquisition" and "merger" especially have the highest positive coefficients. While, some words may not be specific for ACQ category but are also widely used in business news, such as "sell", "complete", "reach" and "held".

4.2 Significant words in EARN classification model

From the SVM model, words that have the highest positive coefficients for EARN category are:

"net, profit, earnings, dividend, cts, shr, loss, qtr, mln, gain, quarter, sell, charge, buyout, stock, note, acquire, pct, results, sells, today, completes, acquired, div, tendered, sales, revs, payable…"

Among them, "net" stands for "net income", it is defined as a corporation's after-tax. "Dividend" may refer to Dividends Per Share (DPS), a corporation's common stock dividends on an annualized basis, divided by the weighted average number of common shares outstanding for the year. "Cts" stands for "Crude oil Transshipment Station", and "shr" stands for "share". "Qtr" stands for "quarterly", and "pct" stands for "per cent".

Most words selected automatically are absolutely in the glossary of terms for Earnings and Earnings Estimates. "Net", "profit" and "earnings" especially have the highest positive coefficients. Some words are relevant to quantity and unit, which is a common issue in the EARN category news. While, some words may not be specific for EARN category but are also widely used in business news, such as "sell", "complete", "gain", "note", "results", "today" and "acquired".

4.3 Significant words for both categories

Some common words are widely used in business news and they usually contribute to the positive coefficients for classification of certain topic news. There are words appear in both ACQ and EARN categories for high positive coefficients. They are “result”, “note”, “sell”, “gain”, “acquire” and “complete”. They are all positive terms for the TRUE judgment of ACQ or EARN in respective models.

5 Reflections and Lessons

From the above analysis, I have learned some terms belonging to certain topic of news, especially for the Earnings and Mergers/Acquisition topics. And I also find some terms that widely appear in most business news. That’s an interesting discovery based on statistical results of data mining.

For the knowledge discovery framework, there are Data Selection, Pre-processing, Transformation, Data Mining and Interpretation, several steps. I spent most of time to the transformation section since there were huge amounts of computation work to create many intermediate tables using SQL queries. Especially to compute Information Gain for all the selected terms, I needed to consider the null values of some attributes and to join tables together to generate right outcomes. I also looked for some commands and codes on the Internet when manipulating the tables. Simultaneously, the transformation section is also the most important section that decides the final performance of classification to a large extent. The selected features significantly contribute to the execution of different classifiers.

6 Future Work

I would have done differently for the category setting when computing Information Gain of all the terms. I will compute IG separately for two classification models. So there would be two different tables of top terms based on IG.

- c_1 = earn, c_2 = not earn
- c_1 = acq, c_2 = not acq

However, there are only 18 instances that belong to both ACQ and EARN categories, so there may not be any big difference. The probability of instances belonging to both categories is so small that it could be overlooked. And the total number of instances is around 6000, which is not enough for more effective data mining. So, more pieces of news with more overlapping could be added to the dataset to make data mining based on larger dataset.

The report of my project is open to the public.

7 Appendices

7.1 Java codes

- GoldStandard.java

```
import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.regex.Matcher;

import java.util.regex.Pattern;
import java.io.BufferedReader;
public class GoldStandard {

        public GoldStandard() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) throws IOException{
                // TODO Auto-generated method stub

                File directory = new File("C:/Documents/GSLIS/590 Text Mining/Dataset/News");
                File[] listOfFiles = directory.listFiles();

     BufferedReader reader = null;
          String line = "";
          PrintWriter outputStream = null;
          String outputFilename = "GoldStandard.csv";
          String allText = "";
          String instanceID = "";
          String type = "";
          String title = "";
          boolean isearn = false;
          boolean isacq = false;

          Pattern patID = Pattern.compile("\\<REUTERS.+NEWID=\"([0-9]+)\"\\>");
                Matcher matID;

                Pattern patType = Pattern.compile("\\<REUTERS.+LEWISSPLIT=\"([a-zA-Z]+)\"");
                Matcher matType;
```

```
            Pattern patTitle = Pattern.compile("\\<TITLE\\>(.+?)\\</TITLE\\>");
            Matcher matTitle;

            Pattern patBody = Pattern.compile(".+\\<BODY\\>([\\S\\s]+)");
            Matcher matBody;

        try
        {
            outputStream = new PrintWriter(outputFilename);
        }
        catch (FileNotFoundException e)
        {
            System.out.println("Error creating the file.");
            System.exit(0);
        }

        outputStream.println("InstanceID" + "," + "Type" + "," + "earn" + "," + "acq" + "," + "Title" +
"," + "Body");

            for (File file : listOfFiles)
            {
                if (file.isFile())
                {
                    try
                  {
                    reader =  new BufferedReader(new FileReader(file));
                  }
                  catch (FileNotFoundException e)
                  {
                    System.out.println("Error opening the file " + file.getName());
                    System.exit(0);
                  }
```

```
				while ((line = reader.readLine())!= null)
			{
					if (line.startsWith("<REUTERS TOPICS=\"YES\""))
					{
						matType = patType.matcher(line);
						if (matType.find())
						{
							type = matType.group(1);
						}
						matID = patID.matcher(line);
						if (matID.find())
						{
							instanceID = matID.group(1);
						}

						while ((line = reader.readLine())!= null
&& !line.contains("</REUTERS>"))
						{
							if (line.startsWith("<TOPICS>"))
							{
								if (line.contains("earn"))
									isearn = true;
								if (line.contains("acq"))
									isacq = true;
							}
							if (line.startsWith("<TITLE>"))
							{
								matTitle = patTitle.matcher(line);
								if (matTitle.find())
								{
									title = matTitle.group(1);
```

```
                                                title = title.replaceAll(",", " ");
                                        }
                                }
                                if (line.contains("<BODY>"))
                                {
                                        matBody = patBody.matcher(line);
                                        if (matBody.find())
                                        {
                                                allText = matBody.group(1) + "
";
                                        }
                                        while ((line = reader.readLine())!= null
&& !line.contains("</BODY>"))
                                        {
                                                allText = allText + " " + line;
                                                allText = allText.replaceAll(",",
" ");
                                                allText                        =
allText.replaceAll("[\\s]{2,}", " ");
                                        }
                                }
                        }

                        if (isearn == true || isacq == true)
                        {
                                outputStream.println(instanceID + "," + type +
"," + isearn + "," + isacq + "," + title + "," + allText);
                        }
                        allText = "";
                        isearn = false;
                        isacq = false;
                }
        }
```

```
                    }
            }

//                      Properties props = new Properties();
//
//                      props.setProperty("annotators","tokenize, ssplit, pos, lemma, ner, parse,
dcoref, sentiment");
//
//                      StanfordCoreNLP pipeline = new StanfordCoreNLP(props);
//                      Annotation annotation = new Annotation(allText);
//                      pipeline.annotate(annotation);
//                      List<CoreMap>               sentences                   =
annotation.get(CoreAnnotations.SentencesAnnotation.class);
//                  int countSentence = 0;
//                      for (CoreMap sentence : sentences)
//                      {
//                          countSentence++;
//                          outputStream.println(abstractID[countFile] + "|" + countSentence
+ "|" + sentence);
//                      }

        outputStream.flush();
            outputStream.close();


    }


}
```

- Tokens,java

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import java.util.Properties;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import edu.stanford.nlp.ling.CoreAnnotations;
import edu.stanford.nlp.ling.CoreAnnotations.TextAnnotation;
import edu.stanford.nlp.pipeline.Annotation;
import edu.stanford.nlp.pipeline.StanfordCoreNLP;
import edu.stanford.nlp.util.CoreMap;

import java.io.BufferedReader;

public class Tokens {

	public Tokens() {
		// TODO Auto-generated constructor stub
	}

	public static void main(String[] args) throws IOException{
		// TODO Auto-generated method stub
		File directory = new File("C:/Documents/GSLIS/590 Text Mining/Dataset/News");
		File[] listOfFiles = directory.listFiles();

    BufferedReader reader = null;
         String line = "";
         PrintWriter outputStream = null;
         String outputFilename = "TokensSub.csv";
         String allText = "";
         String instanceID = "";
String title = "";
boolean isearn = false;
boolean isacq = false;

Pattern patID = Pattern.compile("\\<REUTERS.+NEWID=\"([0-9]+)\"\\>");
        Matcher matID;

        Pattern patTitle = Pattern.compile("\\<TITLE\\>(.+?)\\</TITLE\\>");
        Matcher matTitle;

        Pattern patBody = Pattern.compile(".+\\<BODY\\>([\\S\\s]+)");
        Matcher matBody;

try
{
        outputStream = new PrintWriter(outputFilename);
}
catch (FileNotFoundException e)
{
        System.out.println("Error creating the file.");
        System.exit(0);
}

outputStream.println("InstanceID" + "," + "TermID" + "," + "Term");

        for (File file : listOfFiles)
        {
                if (file.isFile())
                {
                        try
                  {
                    reader =  new BufferedReader(new FileReader(file));
                }
                catch (FileNotFoundException e)
                {
                    System.out.println("Error opening the file " + file.getName());
                    System.exit(0);
                }

                    while ((line = reader.readLine())!= null)
                {
                        if (line.startsWith("<REUTERS TOPICS=\"YES\""))
                        {
                            matID = patID.matcher(line);
                            if (matID.find())
                            {
                                instanceID = matID.group(1);
                            }

                            while ((line = reader.readLine())!= null
&& !line.contains("</REUTERS>"))
                            {
                                if (line.startsWith("<TOPICS>"))
                                {
                                    if (line.contains("earn"))
                                        isearn = true;
                                    if (line.contains("acq"))
                                        isacq = true;
                                }
                                if (line.startsWith("<TITLE>"))
                                {
                                    matTitle = patTitle.matcher(line);
                                    if (matTitle.find())
```

```
                                        {
                                                title = matTitle.group(1);
                                                title = title.replaceAll(",", " ");
                                        }
                                }
                                if (line.contains("<BODY>"))
                                {
                                        matBody = patBody.matcher(line);
                                        if (matBody.find())
                                        {
                                                allText = matBody.group(1) + "
";
                                        }
                                        while ((line = reader.readLine())!= null
&& !line.contains("</BODY>"))
                                        {
                                                allText = allText + " " + line;
                                                allText = allText.replaceAll(",",
" ");
                                                allText                          =
allText.replaceAll("[\\s]{2,}", " ");
                                        }
                                }
                        }

                        if (isearn == true || isacq == true)
                        {
                                Properties props = new Properties();

                                props.setProperty("annotators","tokenize, ssplit,
pos, lemma");
```

```
                                        StanfordCoreNLP pipeline = new
StanfordCoreNLP(props);
                                        Annotation annotation = new Annotation(title + "
" + allText);
                                        pipeline.annotate(annotation);
                                        List<CoreMap> sentences =
annotation.get(CoreAnnotations.SentencesAnnotation.class);
                                    int countTerm = 0;
                                        for (CoreMap sentence : sentences)
                                        {
                                             for (CoreMap token :
sentence.get(CoreAnnotations.TokensAnnotation.class))
                                             {
                                                  String word =
token.get(TextAnnotation.class);
                                                  if
(!word.equalsIgnoreCase("Reuter") && word.matches(".*[a-zA-Z].*"))
                                                  {
                                                       countTerm++;

     outputStream.println(instanceID + "," + countTerm + "," + word);
                                                  }
                                             }
                                        }

                                   }
                                   allText = "";
                                   title = "";
                                   isearn = false;
                                   isacq = false;
                              }
                      }
                    }
                    }

            outputStream.flush();
                    outputStream.close();
        }

}
```

7.2 SQL queries

```
SELECT COUNT(DISTINCT LOWER(TERM))
FROM TX_TERMS;

SELECT COUNT(DISTINCT LOWER(TERM))
FROM TX_TERMS, TX_A2_STOPWORDS
WHERE LOWER(TERM) = TX_A2_STOPWORDS.STOPWORD;

CREATE TABLE TX_TERM_LOWER AS
(
SELECT INSTANCEID, TERMID, LOWER(TERM) AS LOWERCASETERM
FROM TX_TERMS
);

CREATE TABLE TX_TERM_LOWER_DISTINCT AS
SELECT DISTINCT LOWERCASETERM AS DISTINCTTERM
FROM TX_TERM_LOWER;

DELETE FROM TX_TERM_LOWER_DISTINCT
WHERE DISTINCTTERM IN (SELECT DISTINCTTERM
                FROM TX_TERM_LOWER_DISTINCT, TX_A2_STOPWORDS
                WHERE DISTINCTTERM = STOPWORD);
SELECT COUNT(*)
FROM TX_TERM_LOWER_DISTINCT_NOSTOP;
```

```
CREATE TABLE TX_DF AS
SELECT DISTINCTTERM, COUNT(DISTINCT TX_TERM_LOWER.INSTANCEID) AS INSTANCENUMBER
FROM TX_TERM_LOWER_DISTINCT_NOSTOP, TX_TERM_LOWER
WHERE TX_TERM_LOWER_DISTINCT_NOSTOP.DISTINCTTERM = TX_TERM_LOWER.LOWERCASETERM
GROUP BY DISTINCTTERM
HAVING COUNT(DISTINCT TX_TERM_LOWER.INSTANCEID)>1
ORDER BY COUNT(DISTINCT TX_TERM_LOWER.INSTANCEID) DESC;

SELECT COUNT(*)
FROM TX_DF;

CREATE TABLE TX_TF AS
SELECT DISTINCTTERM, INSTANCEID, COUNT(TX_TERM_LOWER.LOWERCASETERM) AS TERMNUMBER
FROM TX_TERM_LOWER, TX_DF
WHERE TX_TERM_LOWER.LOWERCASETERM = TX_DF.DISTINCTTERM
GROUP BY DISTINCTTERM, INSTANCEID
ORDER BY COUNT(TX_TERM_LOWER.LOWERCASETERM) DESC;

SELECT COUNT(DISTINCT DISTINCTTERM)
FROM TX_TF;

CREATE TABLE TX_TFIDF AS
SELECT TX_TF.DISTINCTTERM, INSTANCEID, TERMNUMBER * (LOG(2,6415)-LOG(2,INSTANCENUMBER)+1) AS TFIDF
FROM TX_DF, TX_TF
WHERE TX_TF.DISTINCTTERM = TX_DF.DISTINCTTERM
ORDER BY TERMNUMBER * (LOG(2,6415)-LOG(2,INSTANCENUMBER)+1) DESC;

SELECT COUNT(*)
```

```
FROM TX_TFIDF;

CREATE TABLE TX_TFIDF_TOP AS
SELECT DISTINCT DISTINCTTERM, MAX(TFIDF) AS MAXTFIDF
FROM TX_TFIDF
GROUP BY DISTINCTTERM
ORDER BY MAX(TFIDF) DESC;

SELECT DISTINCTTERM
FROM TX_TFIDF_TOP
WHERE ROWNUM<=100;

CREATE TABLE TX_FEATURES1 (
   INSTANCEID INTEGER,
burlington INTEGER,
saudi INTEGER,
honeywell INTEGER,
gm INTEGER,
sgl INTEGER,
dome INTEGER,
wendy INTEGER,
pesch INTEGER,
csr INTEGER,
heineken INTEGER,
squibb INTEGER,
avana INTEGER,
dart INTEGER,
gillette INTEGER,
encor INTEGER,
monier INTEGER,
reebok INTEGER,
resorts INTEGER,
smc INTEGER,
gaf INTEGER,
hughes INTEGER,
uccel INTEGER,
redstone INTEGER,
sealy INTEGER,
japanese INTEGER,
jardine INTEGER,
fe INTEGER,
express INTEGER,
fleet INTEGER,
medical INTEGER,
sprint INTEGER,
etl INTEGER,
borgwarner INTEGER,
harcourt INTEGER,
renault INTEGER,
bally INTEGER,
southern INTEGER,
sumitomo INTEGER,
equatorial INTEGER,
hk INTEGER,
campeau INTEGER,
usair INTEGER,
bp INTEGER,
ge INTEGER,
proxmire INTEGER,
xerox INTEGER,
southland INTEGER,
alcan INTEGER,
marks INTEGER,
avia INTEGER,
cra INTEGER,
texas INTEGER,
purolator INTEGER,
chemical INTEGER,
emery INTEGER,
eddie INTEGER,
vw INTEGER,
wickes INTEGER,
tmoc INTEGER,
quantum INTEGER,
norcros INTEGER,
bil INTEGER,
ibm INTEGER,
sante INTEGER,
pinola INTEGER,
pantera INTEGER,
alliedlyons INTEGER,
tenneco INTEGER,
ici INTEGER,
idc INTEGER,
lucky INTEGER,
fairchild INTEGER,
santa INTEGER,
allied INTEGER,
bhp INTEGER,
calmat INTEGER,
icahn INTEGER,
keycorp INTEGER,
trailways INTEGER,
agl INTEGER,
mart INTEGER,
caesars INTEGER,
merrill INTEGER,
gencorp INTEGER,
shearson INTEGER,
twa INTEGER,
sosnoff INTEGER,
redland INTEGER,
piedmont INTEGER,
air INTEGER,
crazy INTEGER,
chrysler INTEGER,
perelman INTEGER,
royex INTEGER,
preussag INTEGER,
holdenbrown INTEGER,
schwab INTEGER,
banks INTEGER,
progressive INTEGER,
liberty INTEGER);

INSERT INTO TX_FEATURES1
SELECT *
FROM (SELECT INSTANCEID, LOWERCASETERM
    FROM TX_TERM_LOWER)
    PIVOT
    (COUNT(LOWERCASETERM)
        FOR LOWERCASETERM IN ('burlington',
'saudi',
'honeywell',
'gm',
'sgl',
'dome',
'wendy',
'pesch',
'csr',
'heineken',
'squibb',
'avana',
'dart',
'gillette',
'encor',
'monier',
'reebok',
'resorts',
'smc',
'gaf',
'hughes',
'uccel',
'redstone',
'sealy',
'japanese',
'jardine',
'fe',
'express',
'fleet',
'medical',
'sprint',
'etl',
'borg-warner',
'harcourt',
'renault',
'bally',
'southern',
'sumitomo',
'equatorial',
'hk',
'campeau',
'usair',
'bp',
'ge',
'proxmire',
'xerox',
'southland',
'alcan',
'marks',
'avia',
'cra',
'texas',
'purolator',
'chemical',
'emery',
'eddie',
'vw',
'wickes',
'tmoc',
'quantum',
'norcros',
'bil',
'ibm',
'sante',
'pinola',
```

'pantera',

'allied-lyons',

'tenneco',

'ici',

'idc',

'lucky',

'fairchild',

'santa',

'allied',

'bhp',

'calmat',

'icahn',

'keycorp',

'trailways',

'agl',

'mart',

'caesars',

'merrill',

'gencorp',

'shearson',

'twa',

'sosnoff',

'redland',

'piedmont',

'air',

'crazy',

'chrysler',

'perelman',

'royex',

'preussag',

'holden-brown',

```
'schwab',
'banks',
'progressive',
'liberty'));

select count(*)
from TX_FEATURES1;

select count(distinct TX_terms.INSTANCEID)
from TX_terms, TX_GOLDSTANDARD
where TX_TERMS.INSTANCEID = TX_GOLDSTANDARD.INSTANCEID
  AND TX_GOLDSTANDARD.EARN = 'TRUE' AND TX_GOLDSTANDARD.ACQ = 'TRUE';

CREATE TABLE TX_C1 AS
SELECT TX_FEATURES1.*, TX_GOLDSTANDARD.EARN, TX_GOLDSTANDARD.ACQ
FROM TX_FEATURES1
JOIN TX_GOLDSTANDARD ON TX_FEATURES1.INSTANCEID = TX_GOLDSTANDARD.INSTANCEID;

SELECT COUNT (*)
FROM TX_C1;

CREATE TABLE TX_DF_PR1 AS
SELECT DISTINCTTERM, INSTANCENUMBER/5967 AS PR1
FROM TX_DF;

CREATE TABLE TX_DF_ACQNUMBER AS
SELECT TX_DF.DISTINCTTERM, COUNT(DISTINCT TX_TERM_LOWER.INSTANCEID) AS ACQNUMBER
FROM TX_DF, TX_TERM_LOWER, TX_GOLDSTANDARD
WHERE TX_DF.DISTINCTTERM = TX_TERM_LOWER.LOWERCASETERM AND TX_TERM_LOWER.INSTANCEID = TX_GOLDSTANDARD.INSTANCEID
```

```
    AND TX_GOLDSTANDARD.ACQ = 'TRUE' AND TX_GOLDSTANDARD.EARN = 'FALSE'
GROUP BY TX_DF.DISTINCTTERM
ORDER BY ACQNUMBER DESC;

CREATE TABLE TX_DF_NUM_PR2 AS
SELECT TX_DF.DISTINCTTERM, TX_DF_ACQNUMBER.ACQNUMBER AS ACQNUMBER
FROM TX_DF_ACQNUMBER
RIGHT OUTER JOIN TX_DF
ON TX_DF.DISTINCTTERM = TX_DF_ACQNUMBER.DISTINCTTERM;

SELECT COUNT (*)
FROM TX_DF_ACQNUMBER;

SELECT COUNT(*)
FROM TX_DF_NUM_PR2;

CREATE TABLE TX_DF_PR2 AS
SELECT TX_DF.DISTINCTTERM, ACQNUMBER/INSTANCENUMBER AS PR2
FROM TX_DF, TX_DF_NUM_PR2
WHERE TX_DF.DISTINCTTERM = TX_DF_NUM_PR2.DISTINCTTERM;

SELECT COUNT(*)
FROM TX_DF_PR1;

SELECT COUNT(*)
FROM TX_DF_PR2;

CREATE TABLE TX_DF_EARNNUMBER AS
SELECT TX_DF.DISTINCTTERM, COUNT(DISTINCT TX_TERM_LOWER.INSTANCEID) AS
EARNNUMBER
FROM TX_DF, TX_TERM_LOWER, TX_GOLDSTANDARD
```

```
WHERE TX_DF.DISTINCTTERM = TX_TERM_LOWER.LOWERCASETERM AND
TX_TERM_LOWER.INSTANCEID = TX_GOLDSTANDARD.INSTANCEID
    AND TX_GOLDSTANDARD.ACQ = 'FALSE' AND TX_GOLDSTANDARD.EARN = 'TRUE'
GROUP BY TX_DF.DISTINCTTERM
ORDER BY EARNNUMBER DESC;

SELECT COUNT(*)
FROM TX_DF_EARNNUMBER;

CREATE TABLE TX_DF_NUM_PR3 AS
SELECT TX_DF.DISTINCTTERM, TX_DF_EARNNUMBER.EARNNUMBER AS EARNNUMBER
FROM TX_DF_EARNNUMBER
RIGHT OUTER JOIN TX_DF
ON TX_DF.DISTINCTTERM = TX_DF_EARNNUMBER.DISTINCTTERM;

SELECT COUNT(*)
FROM TX_DF_NUM_PR3;

CREATE TABLE TX_DF_PR3 AS
SELECT TX_DF.DISTINCTTERM, EARNNUMBER/INSTANCENUMBER AS PR3
FROM TX_DF, TX_DF_NUM_PR3
WHERE TX_DF.DISTINCTTERM = TX_DF_NUM_PR3.DISTINCTTERM;

SELECT COUNT(*)
FROM TX_DF_PR3;

CREATE TABLE TX_DF_BOTHNUMBER AS
SELECT TX_DF.DISTINCTTERM, COUNT(DISTINCT TX_TERM_LOWER.INSTANCEID) AS
BOTHNUMBER
FROM TX_DF, TX_TERM_LOWER, TX_GOLDSTANDARD
WHERE TX_DF.DISTINCTTERM = TX_TERM_LOWER.LOWERCASETERM AND
TX_TERM_LOWER.INSTANCEID = TX_GOLDSTANDARD.INSTANCEID
```

```
    AND TX_GOLDSTANDARD.ACQ = 'TRUE' AND TX_GOLDSTANDARD.EARN = 'TRUE'
GROUP BY TX_DF.DISTINCTTERM
ORDER BY BOTHNUMBER DESC;

SELECT COUNT(*)
FROM TX_DF_BOTHNUMBER;

CREATE TABLE TX_DF_NUM_PR3_5 AS
SELECT TX_DF.DISTINCTTERM, TX_DF_BOTHNUMBER.BOTHNUMBER AS BOTHNUMBER
FROM TX_DF_BOTHNUMBER
RIGHT OUTER JOIN TX_DF
ON TX_DF.DISTINCTTERM = TX_DF_BOTHNUMBER.DISTINCTTERM;

SELECT COUNT(*)
FROM TX_DF_NUM_PR3_5;

CREATE TABLE TX_DF_PR3_5 AS
SELECT TX_DF.DISTINCTTERM, BOTHNUMBER/INSTANCENUMBER AS PR3_5
FROM TX_DF, TX_DF_NUM_PR3_5
WHERE TX_DF.DISTINCTTERM = TX_DF_NUM_PR3_5.DISTINCTTERM;

SELECT COUNT(*)
FROM TX_DF_PR3_5;

CREATE TABLE TX_DF_PR4 AS
SELECT DISTINCTTERM, 1 - INSTANCENUMBER/5967 AS PR4
FROM TX_DF;

SELECT COUNT(*)
FROM TX_DF_PR4;
```

```
CREATE TABLE TX_DF_PR5 AS
SELECT TX_DF.DISTINCTTERM, (2192-NVL(ACQNUMBER,0))/(5967-INSTANCENUMBER) AS PR5
FROM TX_DF, TX_DF_NUM_PR2
WHERE TX_DF.DISTINCTTERM = TX_DF_NUM_PR2.DISTINCTTERM;

SELECT COUNT(*)
FROM TX_DF_PR5;

DROP TABLE TX_DF_PR6;
CREATE TABLE TX_DF_PR6 AS
SELECT TX_DF.DISTINCTTERM, (3757-NVL(EARNNUMBER,0))/(5967-INSTANCENUMBER) AS PR6
FROM TX_DF, TX_DF_NUM_PR3
WHERE TX_DF.DISTINCTTERM = TX_DF_NUM_PR3.DISTINCTTERM;

SELECT COUNT(*)
FROM TX_DF_PR6;

CREATE TABLE TX_DF_PR6_5 AS
SELECT TX_DF.DISTINCTTERM, (18-NVL(BOTHNUMBER,0))/(5967-INSTANCENUMBER) AS PR6_5
FROM TX_DF, TX_DF_NUM_PR3_5
WHERE TX_DF.DISTINCTTERM = TX_DF_NUM_PR3_5.DISTINCTTERM;

SELECT COUNT(*)
FROM TX_DF_PR6_5;

CREATE TABLE TX_IG AS
SELECT TX_DF_PR1.DISTINCTTERM, (-2192/5967*LOG(2,2192/5967)-3757/5967*LOG(2,3757/5967)-18/5967*LOG(2,18/5967) + PR1*(NVL(PR2,0)*LOG(2,NVL(PR2,1)) + NVL(PR3,0)*LOG(2,NVL(PR3,1)) + NVL(PR3_5,0)*LOG(2,NVL(PR3_5,1))) + PR4*(PR5*LOG(2,PR5) + PR6*LOG(2,PR6) + PR6_5*LOG(2,PR6_5))) AS IG
```

```
FROM TX_DF_PR1, TX_DF_PR2, TX_DF_PR3, TX_DF_PR3_5, TX_DF_PR4, TX_DF_PR5,
TX_DF_PR6, TX_DF_PR6_5
WHERE TX_DF_PR1.DISTINCTTERM = TX_DF_PR2.DISTINCTTERM AND
TX_DF_PR2.DISTINCTTERM = TX_DF_PR3.DISTINCTTERM
    AND TX_DF_PR3.DISTINCTTERM = TX_DF_PR4.DISTINCTTERM AND
TX_DF_PR4.DISTINCTTERM = TX_DF_PR5.DISTINCTTERM
    AND TX_DF_PR5.DISTINCTTERM = TX_DF_PR6.DISTINCTTERM
    AND TX_DF_PR6.DISTINCTTERM = TX_DF_PR6_5.DISTINCTTERM
    AND TX_DF_PR6_5.DISTINCTTERM = TX_DF_PR3_5.DISTINCTTERM
ORDER BY IG DESC;

SELECT COUNT(*)
FROM TX_IG;

SELECT DISTINCTTERM
FROM TX_IG
WHERE ROWNUM<=105;

CREATE TABLE TX_FEATURES2 (
   INSTANCEID INTEGER,
   cts INTEGER,
shr INTEGER,
net INTEGER,
qtr INTEGER,
revs INTEGER,
note INTEGER,
loss INTEGER,
shares INTEGER,
acquire INTEGER,
acquisition INTEGER,
offer INTEGER,
buy INTEGER,
stake INTEGER,
pct INTEGER,
mths INTEGER,
agreement INTEGER,
agreed INTEGER,
avg_ INTEGER,
shrs INTEGER,
terms INTEGER,
company INTEGER,
year_ INTEGER,
profit INTEGER,
div INTEGER,
record_ INTEGER,
merger INTEGER,
sell INTEGER,
purchase INTEGER,
bid INTEGER,
unit INTEGER,
common INTEGER,
exchange_ INTEGER,
transaction_ INTEGER,
acquired INTEGER,
group_ INTEGER,
qtly INTEGER,
commission INTEGER,
prior_ INTEGER,
dividend INTEGER,
tender INTEGER,
completed INTEGER,
outstanding INTEGER,
undisclosed INTEGER,
cash INTEGER,
price INTEGER,
firm INTEGER,
securities INTEGER,
subsidiary INTEGER,
bought INTEGER,
stock INTEGER,
buys INTEGER,
subject INTEGER,
takeover INTEGER,
disclosed INTEGER,
oper INTEGER,
completes INTEGER,
investor INTEGER,
deal INTEGER,
signed INTEGER,
quarter INTEGER,
companies INTEGER,
letter INTEGER,
approval INTEGER,
proposed INTEGER,
control INTEGER,
filing INTEGER,
management INTEGER,
tax INTEGER,
held INTEGER,
sells INTEGER,
sold INTEGER,
payout INTEGER,
corp INTEGER,
definitive INTEGER,
owned INTEGER,
excludes INTEGER,
results INTEGER,
intent INTEGER,
quarterly INTEGER,
announced INTEGER,
owns INTEGER,
board INTEGER,
total INTEGER,
shareholders INTEGER,
jan INTEGER,
part INTEGER,
talks INTEGER,
includes INTEGER,
comment_ INTEGER,
proposal INTEGER,
buyout INTEGER,
made INTEGER,
holds INTEGER,
split_ INTEGER,
gain INTEGER,
principle INTEGER,
pay INTEGER,
spokesman INTEGER,
investment INTEGER,
mln INTEGER);

INSERT INTO TX_FEATURES2
SELECT *
FROM (SELECT INSTANCEID, LOWERCASETERM
    FROM TX_TERM_LOWER)
    PIVOT
    (COUNT(LOWERCASETERM)
        FOR LOWERCASETERM IN ('cts',
'shr',
'net',
'qtr',
'revs',
'note',
'loss',
'shares',
'acquire',
'acquisition',
'offer',
'buy',
'stake',
'pct',
'mths',
'agreement',
'agreed',
'avg_',
'shrs',
'terms',
'company',
'year_',
'profit',
'div',
'record_',
'merger',
'sell',
'purchase',
'bid',
'unit',
'common',
'exchange_',
'transaction_',
'acquired',
'group_',
'qtly',
'commission',
'prior_',
'dividend',
'tender',
'completed',
'outstanding',
'undisclosed',
'cash',
'price',
'firm',
'securities',
'subsidiary',
'bought',
'stock',
'buys',
'subject',
'takeover',
'disclosed',
'oper',
'completes',
'investor',
'deal',
'signed',
'quarter',
'companies',
'letter',
'approval',
'proposed',
'control',
'filing',
'management',
'tax',
'held',
'sells',
'sold',
'payout',
'corp',
'definitive',
'owned',
'excludes',
'results',
'intent',
'quarterly',
'announced',
'owns',
'board',
'total',
'shareholders',
'jan',
'part',
'talks',
'includes',
'comment_',
'proposal',
'buyout',
'made',
'holds',
'split_',
'gain',
'principle',
'pay',
'spokesman',
'investment',
'mln'));

select count(*)
from TX_FEATURES2;

CREATE TABLE TX_C2 AS
SELECT TX_FEATURES2.*, TX_GOLDSTANDARD.EARN, TX_GOLDSTANDARD.ACQ
FROM TX_FEATURES2
JOIN TX_GOLDSTANDARD ON TX_FEATURES2.INSTANCEID = TX_GOLDSTANDARD.INSTANCEID;

SELECT COUNT (*)
FROM TX_C2;

SELECT DISTINCTTERM
FROM TX_IG
WHERE ROWNUM <= 205;

CREATE TABLE TX_FEATURES3 (
   INSTANCEID INTEGER,
   cts INTEGER,
shr INTEGER,
net INTEGER,
qtr INTEGER,
revs INTEGER,
note INTEGER,
loss INTEGER,
shares INTEGER,
acquire INTEGER,
acquisition INTEGER,
offer INTEGER,
buy INTEGER,
stake INTEGER,
pct INTEGER,
mths INTEGER,
agreement INTEGER,
agreed INTEGER,
avg_ INTEGER,
shrs INTEGER,
terms INTEGER,
company INTEGER,
year_ INTEGER,
profit INTEGER,
div INTEGER,
record_ INTEGER,
merger INTEGER,
sell INTEGER,
purchase INTEGER,
bid INTEGER,
unit INTEGER,
common INTEGER,
exchange_ INTEGER,
transaction_ INTEGER,
acquired INTEGER,
group_ INTEGER,
qtly INTEGER,
commission INTEGER,
prior_ INTEGER,
dividend INTEGER,
tender INTEGER,
completed INTEGER,
outstanding INTEGER,
undisclosed INTEGER,
cash INTEGER,
price INTEGER,
firm INTEGER,
securities INTEGER,
subsidiary INTEGER,
bought INTEGER,
stock INTEGER,
buys INTEGER,
subject INTEGER,
takeover INTEGER,
disclosed INTEGER,
oper INTEGER,
completes INTEGER,
investor INTEGER,
deal INTEGER,
signed INTEGER,
quarter INTEGER,
companies INTEGER,
letter INTEGER,
approval INTEGER,
proposed INTEGER,
control INTEGER,
filing INTEGER,
management INTEGER,
tax INTEGER,
held INTEGER,
sells INTEGER,
sold INTEGER,
payout INTEGER,
corp INTEGER,
definitive INTEGER,
owned INTEGER,
excludes INTEGER,
results INTEGER,
intent INTEGER,
quarterly INTEGER,
announced INTEGER,
owns INTEGER,
board INTEGER,
total INTEGER,
shareholders INTEGER,
jan INTEGER,
part INTEGER,
talks INTEGER,
includes INTEGER,
comment_ INTEGER,
proposal INTEGER,
buyout INTEGER,
made INTEGER,
holds INTEGER,
split_ INTEGER,
gain INTEGER,
principle INTEGER,
pay INTEGER,
spokesman INTEGER,
investment INTEGER,
mln INTEGER,
discontinued INTEGER,
offered INTEGER,
extraordinary INTEGER,
led INTEGER,
market INTEGER,
seeking INTEGER,
makes INTEGER,
shareholder INTEGER,
make INTEGER,
plans INTEGER,
receive INTEGER,
holding INTEGER,
acquires INTEGER,
statement_ INTEGER,
holdings INTEGER,
week INTEGER,
reached INTEGER,
seek INTEGER,
payable INTEGER,
directors INTEGER,
worth INTEGER,
york INTEGER,
earnings INTEGER,
received INTEGER,
told INTEGER,
chairman INTEGER,
merge_ INTEGER,
income INTEGER,
international INTEGER,
ct INTEGER,
sale INTEGER,
financing INTEGER,
purchased INTEGER,
business INTEGER,
tendered INTEGER,
plc INTEGER,
sets INTEGER,
yesterday INTEGER,
periods INTEGER,
controlled INTEGER,
april INTEGER,
details INTEGER,
sees INTEGER,
approved INTEGER,
lrb INTEGER,
nil INTEGER,
rrb INTEGER,
court INTEGER,
gains INTEGER,
extended INTEGER,
today INTEGER,
investors INTEGER,
valued INTEGER,
issued INTEGER,
acquiring INTEGER,
restated INTEGER,
hold INTEGER,
option_ INTEGER,
officials INTEGER,
plan INTEGER,
rejected INTEGER,
filed INTEGER,
feb INTEGER,
division INTEGER,
entered INTEGER,
regulatory INTEGER,
offers INTEGER,
based INTEGER,
discussions INTEGER,
privately_held INTEGER,
closing INTEGER,
largest INTEGER,
days INTEGER,
figures INTEGER,
members INTEGER,
parties INTEGER,
private_ INTEGER,
interest INTEGER,
previously INTEGER,
full_ INTEGER,
firms INTEGER,
buying INTEGER,
completion INTEGER,
rights INTEGER,
ranging INTEGER,
time_ INTEGER,
amount INTEGER,
speculation INTEGER,
declared INTEGER,
airlines INTEGER,
sales INTEGER,
partnership INTEGER,
credits INTEGER,
assets INTEGER,
federal INTEGER,
including_ INTEGER,
government INTEGER,
adjusted INTEGER,
charge INTEGER,
response INTEGER);

INSERT INTO TX_FEATURES3
SELECT *
FROM (SELECT INSTANCEID, LOWERCASETERM
    FROM TX_TERM_LOWER)
    PIVOT
    (COUNT(LOWERCASETERM)
        FOR LOWERCASETERM IN ('cts',
'shr',
'net',
'qtr',
'revs',
'note',
'loss',
'shares',
'acquire',
'acquisition',
'offer',
'buy',
'stake',
'pct',
'mths',
'agreement',
'agreed',
'avg_',
'shrs',
'terms',
'company',
'year_',
'profit',
'div',
'record_',
'merger',
'sell',
'purchase',
'bid',
'unit',
'common',
'exchange_',
'transaction_',
'acquired',
'group_',
'qtly',
'commission',
'prior_',
'dividend',
'tender',
'completed',
'outstanding',
'undisclosed',
'cash',
'price',
'firm',
'securities',
'subsidiary',
'bought',
'stock',
'buys',
'subject',
'takeover',
'disclosed',
'oper',
'completes',
'investor',
'deal',
'signed',
'quarter',
'companies',
'letter',
'approval',
'proposed',
'control',
'filing',
'management',
'tax',
'held',
'sells',
'sold',
'payout',
'corp',
'definitive',
'owned',
'excludes',
'results',
'intent',
'quarterly',
'announced',
'owns',
'board',
'total',
'shareholders',
'jan',
'part',
'talks',
'includes',
'comment_',
'proposal',
'buyout',
'made',
'holds',
'split_',
'gain',
'principle',
'pay',
'spokesman',
'investment',
'mln',
'discontinued',
'offered',
'extraordinary',
'led',
'market',
'seeking',
'makes',
'shareholder',
'make',
'plans',
'receive',
'holding',
'acquires',
'statement_',
'holdings',
'week',
'reached',
'seek',
'payable',
'directors',
'worth',
'york',
'earnings',
'received',
'told',
'chairman',
'merge_',
'income',
'international',
'ct',
'sale',
'financing',
'purchased',
'business',
'tendered',
'plc',
'sets',
'yesterday',
'periods',
'controlled',
'april',
'details',
'sees',
'approved',
'lrb',
'nil',
'rrb',
'court',
'gains',
'extended',
'today',
'investors',
'valued',
'issued',
'acquiring',
'restated',
'hold',
'option_',
'officials',
'plan',
'rejected',
'filed',
'feb',
'division',
'entered',
'regulatory',
'offers',
'based',
'discussions',
'privately_held',
'closing',
'largest',
'days',
'figures',
'members',
'parties',
'private_',
'interest',
'previously',
'full_',
'firms',
'buying',
'completion',
'rights',
'ranging',
'time_',
'amount',
'speculation',
'declared',
'airlines',
'sales',
'partnership',
'credits',
'assets',
'federal',
'including_',
'government',
'adjusted',
'charge',
'response'));
```

```
select count(*)
from TX_FEATURES3;


CREATE TABLE TX_C3 AS
SELECT TX_FEATURES3.*, TX_GOLDSTANDARD.EARN, TX_GOLDSTANDARD.ACQ
FROM TX_FEATURES3
JOIN TX_GOLDSTANDARD ON TX_FEATURES3.INSTANCEID = TX_GOLDSTANDARD.INSTANCEID;


SELECT COUNT (*)
FROM TX_C3;
```

References

A comparative study on feature selection in text categorization.pdf. (n.d.).

Appendix of Reuters-21578 & RCV1.pdf. (n.d.).

Crammer, K., & Singer, Y. (2003). A Family of Additive Online Algorithms for Category Ranking. *J. Mach. Learn. Res.*, *3*, 1025–1058.

Debole, F., & Sebastiani, F. (2005). An Analysis of the Relative Hardness of Reuters-21578 Subsets. *Journal of the American Society for Information Science & Technology*, *56*(6), 584–596. http://doi.org/10.1002/asi.20147

Earnings Estimates Glossary. (n.d.). Retrieved May 10, 2016, from http://phx.corporate-ir.net/phoenix.zhtml?c=131042&p=irol-estimatesGlossary

Evaluation of text classification. (n.d.). Retrieved February 24, 2016, from http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-text-classification-1.html

Ghamrawi, N., & McCallum, A. (2005). Collective Multi-label Classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management* (pp. 195–200). New York, NY, USA: ACM. http://doi.org/10.1145/1099554.1099591

Giddy: Mergers & Acquisitions Glossary. (n.d.). Retrieved May 10, 2016, from http://pages.stern.nyu.edu/~igiddy/articles/merger_glossary.htm

Homework: Text Categorization. (n.d.). Retrieved January 31, 2016, from http://boston.lti.cs.cmu.edu/classes/95-865/HW/HW2/

Kassab, R., & Lamirel, J.-C. (2006). A New Approach to Intelligent Text Filtering Based on Novelty Detection. In *Proceedings of the 17th Australasian Database Conference - Volume 49* (pp. 149–156). Darlinghurst, Australia, Australia: Australian Computer Society, Inc. Retrieved from http://dl.acm.org/citation.cfm?id=1151736.1151752

Rogati, M., & Yang, Y. (2002). High-performing Feature Selection for Text Classification. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management* (pp. 659–661). New York, NY, USA: ACM. http://doi.org/10.1145/584792.584911

Yang, Y., & Liu, X. (1999). A Re-examination of Text Categorization Methods. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 42–49). New York, NY, USA: ACM. http://doi.org/10.1145/312624.312647

Zhang, J., & Yang, Y. (2003). Robustness of Regularized Linear Classification Methods in Text Categorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval* (pp. 190–197). New York, NY, USA: ACM. http://doi.org/10.1145/860435.860471