ILS Z534 Information Retrieval: Theory and Practice
Yelp Dataset Challenge

Project Report

Group members

Aravindh Varadharaju (avaradha@indiana.edu)
Awani Marathe (amarathe@indiana.edu)
Divya Lalwani (dlalwani@indiana.edu)
Jayagowri Rajasekar (jperumal@indiana.edu)

Under the guidance of
Prof. Xiaozhong Liu
at
Indiana University, Bloomington

Table of Contents

Introduction

This project is part of the course Z534: Information Retrieval- Theory and Practice. For this project we use Yelp data, which consists of both text, and numeric data. The data set has 42, 153 businesses and 1, 125, 458 reviews along with other details. The dataset was provided in JSON format. The project has been divided into two tasks:

1. Task I: Find the categories of each business
2. Task II: Predict the ratings of the reviews based on the content of the reviews

The remaining parts of the documents talks about the two tasks involved, methodology followed and experimental outputs.

Task I

For task 1, the objective was to predict the category of Restaurants a review would belong to. To achieve this objective, we employed 4 major steps:

1. Fetched relevant data from the yelp data set
2. Preprocessed reviews, filtering out to those words in a review that were interesting.
3. Employed IR techniques (TF-IDF specifically) for creating a feature space.
4. Used Weka for analysis and comparison of performance.

The report elaborates further on each of these steps.

Data gathering

The Yelp data set is a massive dump of data containing information about businesses, reviews, tips etc. We were motivated to select all data relating to Restaurants for two reasons:

1. The data set would be manageable to run on all student and school machines
2. A specific data set like the one related only to restaurants provided more clarity in understanding of its analysis and performance reviews of techniques that were applied on it.

We used MongoDB to load and filter data. We constructed mongo queries that resulted in two JSON files that were used for further manipulation:

1. The first RestaurantsReviews.json file contains every business id in the data set (connected to Restaurants) and the corresponding categories associated with it (8454 entries). Business ids here could be considered a primary key. The categories associated are of the form – [“Chinese”, “Indian”, “Mexican”] (all the categories belonging to the same business)

2. The second BusinessReviews.json file contains every business id associated with every review (86,579 entries). The reviews and ids may be repeating.

The mongo output was then processed by our python script that created a dictionary associating every business id with corresponding categories and clubbed reviews.

Data Preprocessing

We used Python's NLTK tool kit for preprocessing the data contained in each review. It involved the following steps:

1. ***Removing Punctuations and Special Characters:***
   While parsing the reviews text, it was important to remove characters like punctuations, digits and special characters, which are all of no importance. So, we removed all such characters from the text.
2. ***Removing Stop-words:***
   Some words added little or no relevance to review text. Words like "I", "the", "but" etc. called stop words. We used NLTK package and our own list of stop words in order to remove such words.
3. ***Stemming:***
   Words like "brief" and "briefly" were treated as different words. Although, we knew that they belonged to same root "brief" and conveyed the same sentiment. So, we considered it important to convert a word to its canonical form. We used stemming from nltk package for this purpose.

Feature Space Construction

Reasons for selecting up to 16 categories:

Since the dataset was massive, we used only subcategories of the "Restaurants" category. There were approx. 217 unique subcategories. To understand the entire process in a better way, we narrowed down the number of categories in our dataset. We used 16 of the most popular categories to build our dataset and these categories are: Italian, Burgers, Mediterranean, Chinese, Steakhouses, Mongolian, Barbeque, Japanese, Pakistani, Afghan, Mexican, Nightlife, American (New), Bakeries, Breakfast & Brunch, Thai, Middle Eastern and Sushi Bars

Spell check:

Today's generation has a different spelling dictionary. For instance, friends is spelled as frns, girl as gal and so on. It was important that such words did not lose the context, so we tried to implement spell check in python. We decided against using a spell checker for the following reasons. The spell checker in python used an enchant library that used C language libraries that were found on the Mac OS, installed using homebrew, hence it wasn't platform independent. Further, it took a larger computation time with little or no improvement in the performance using the existing data set and it sometimes filtered out unique words that it interpreted incorrectly.

TF/IDF:

Possibly the most crucial part of the process was the use of TF-IDF IR technique for creating a good feature space. We calculated the term frequency for every word in the reviews for the specific category. We calculated the IDF as function of (LOG_{10} (n/d)) where n was the total number of reviews associated with a particular category and d was the total number of documents that word occurred in. The product of every word's TF and IDF across all reviews for every specific category was stored in a Priority Queue that returned the top 20 (shortlisted after several runs) words with maximum score. These words were added as relevant features for that category in the feature space.

Weka Analysis and Comparison

Using all words versus nouns:

For building feature space, we did not use all the words in the text of a category because this would increase the length of feature vector exponentially, making it difficult to process data. Apart from data size issues, using all words was not appropriate, as all the words weren't relevant ones. So, it was important to pick only top few words as per the TF/IDF score. All words belong to some part of speech in English language.

We created two datasets from the resultant text of the above preprocessing (using first 1500 reviews):

1. Using top 20 relevant words by using *all word*s from the text
2. Using top 20 relevant words by using *only nouns* from the text

Then we ran **T-tests** to compare the performance of Multinomial Naïve Bayes on both these datasets. We used accuracy, precision and recall as evaluation metrics to compare performance in this case. Below are the results:

```
(1.) Tester:     weka.experiment.PairedTTester
Analysing:  Percent_correct

Dataset              (1) bayes.Naiv
---------------------------------------
AllWords              (10)   39.14 |
OnlyNouns               (10)   40.00 |
---------------------------------------
(v/ /*)                         |
```

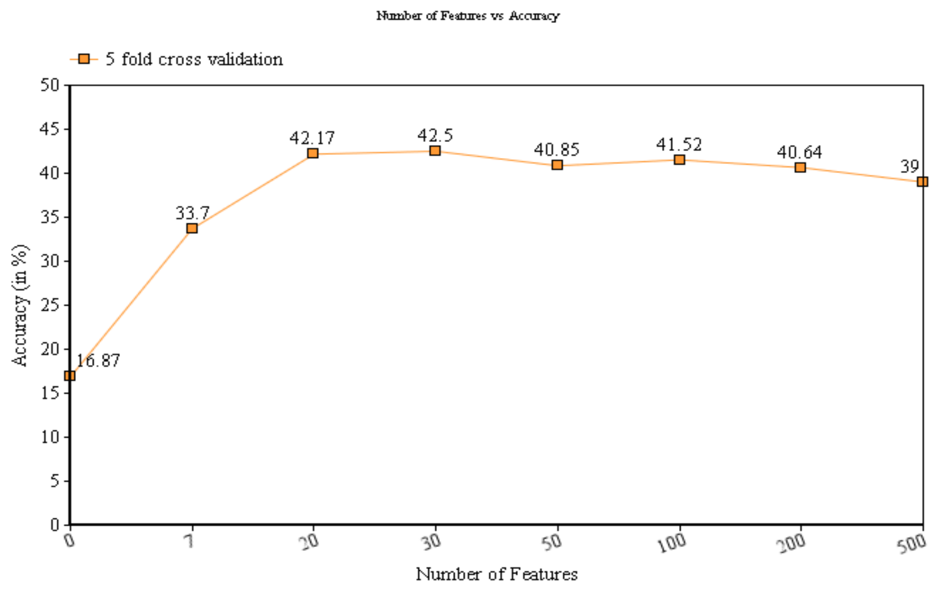
```
(2.) Tester:     weka.experiment.PairedTTester
Analysing:  IR_precision

Dataset              (1) bayes.Nai
--------------------------------------
AllWords              (10)   0.37 |
OnlyNouns               (10)   0.38 |
--------------------------------------
(v/ /*)                        |
(3.) Tester:     weka.experiment.PairedTTester
Analysing:  IR_recall

Dataset              (1) bayes.Nai
--------------------------------------
AllWords              (10)   0.50 |
OnlyNouns              (10)   0.56 |
--------------------------------------
(v/ /*)                         |
```

We saw that there is very slight difference in the performance. But, since the dataset used for comparison was not huge, this difference would have increased with the increase in the size of dataset. Also, by going through the list of words in the feature space (for the categories in which we considered ourselves domain experts), we noticed that only nouns had more relevant words as compared to all words. So, as per these results, we used only nouns to build our feature space.

Picking top n features of a category to create feature space- shortlisted to top 20 features:

Now that we knew we were using only nouns from the text, next step was to determine the optimized value of n (using parameter tuning) to find top relevant features. We had created datasets using different values of n and used **accuracy** as **evaluation metric** to compare performance of these datasets. We used 1500 entries as our data set.

The results are displayed in Graph 1:

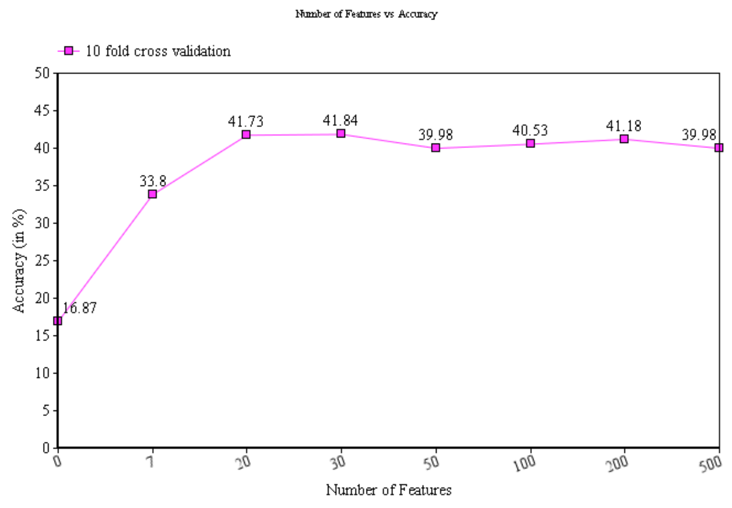


Graph 1

We could see that the accuracy increased exponentially till n= 20. Then it is stable and then decreased slightly for n=500. Highest accuracy is for n=30, but accuracy for n=20 is almost the same. The feature space increases for n=30 and it is not worth to pay the cost for a large feature space knowing that there is hardly any difference in the accuracy (for n=20). So, n=20 is the optimized value for our dataset.

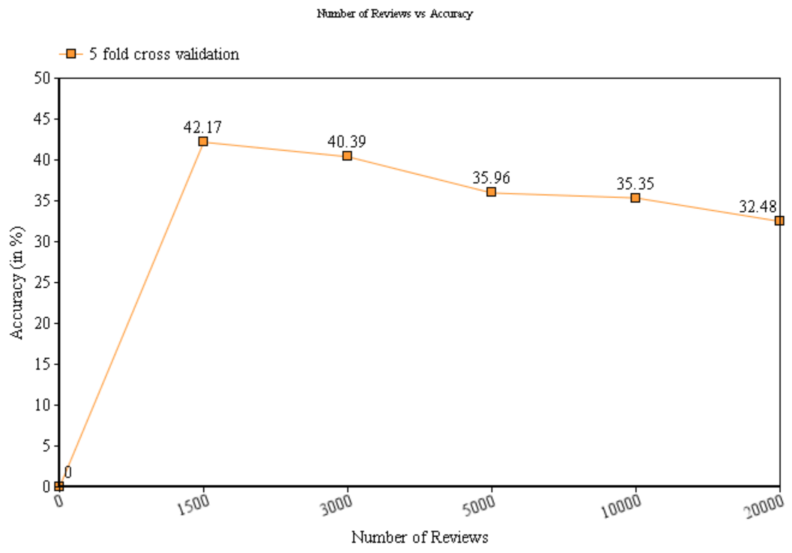
***k-cross fold validation*:**

In graph 1, we used 5 cross fold validation and in graph 2, we used 10 cross fold validation. We could see that accuracy is almost same for a given n in both the cases. Since for n=20, 5 cross fold validation gives a slight higher accuracy, we are using k = 5.



Graph 2

Choosing the length of dataset:



Graph 3

We could see that as the length of dataset increased, accuracy decreased but only marginally. We couldn't go beyond 20,000 reviews as the processing time was increasing dramatically with increase in size of the dataset, which got difficult for our local machines and Weka to handle. Dataset with 20,000 reviews is worked for Multinomial Naïve Bayes, but for other classifiers like Decision Trees and SMOs, Weka shut down due to memory constraints. So, we used dataset with 10,000 reviews to analyze the performance of different classifiers.

Results

Our final dataset consists of 16 class labels and the feature space is build using top 20 relevant words for each category.

Here are the results of building different classifiers using 5 cross fold validation from Weka on dataset of 20,000 reviews:

Algorithm/Metric	**Accuracy**	**Average Precision**	**Average Recall**
Multinomial Naïve Bayes	35.35%	0.333	0.353
Naïve Bayes	32.34%	0.34	0.323
SMO (SVM) (using PolyKernel)	30.41%	0.294	0.304
Decision Trees (J48)	27.41%	0.266	0.274

Improvements

1. A powerful spell checker that recognizes popular, colloquial words that works on all platforms might improve the chances of better, more relevant words in the feature space
2. Improvement on the TF-IDF algorithm for a better feature space

Task II

Task II of the project is to predict ratings from user reviews. The Yelp dataset has details about businesses and each business has reviews from users. Similar to Task I, the dataset was restricted to reviews to businesses under restaurants. The number of reviews provided as part of the dataset was 1125458. Each review had a usefulness rating from 1 to 5. For this task, the reviews with usefulness rating of more than 3 were taken into consideration. The number of reviews with usefulness rating greater than 3 and under restaurant category was 49876.

The dataset was provided as JSON files. These JSON files were loaded into MongoDB. Once loaded, we were able to query the data according to our requirements.

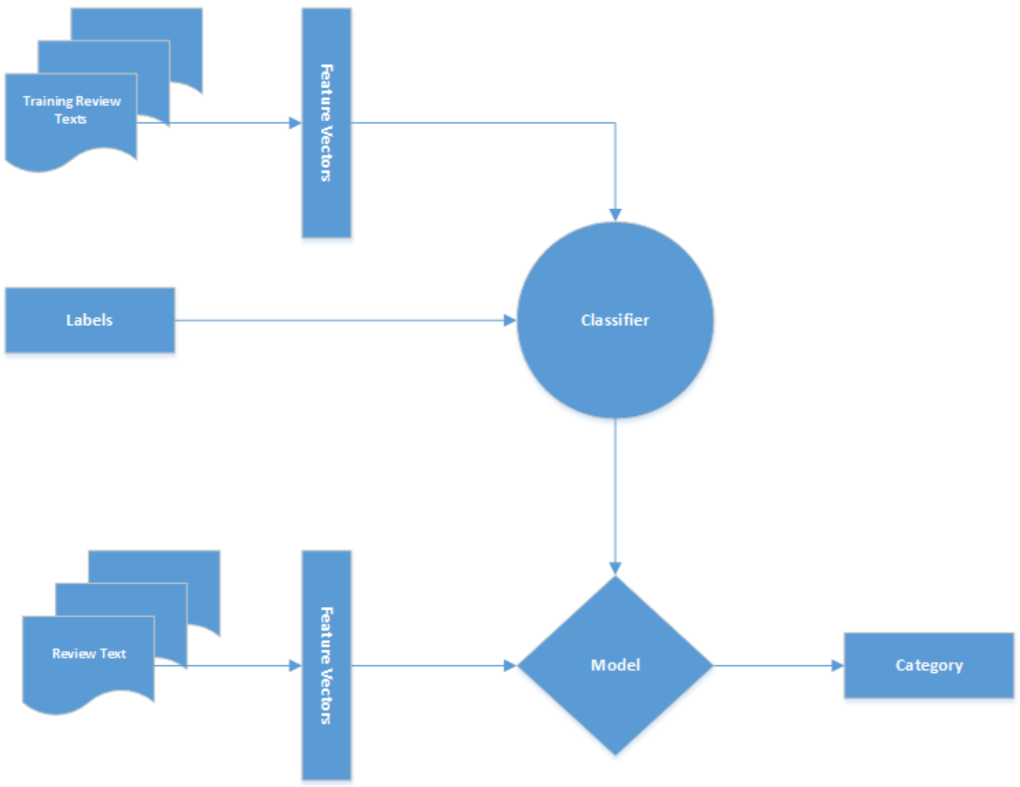
Opinions are expressed in form of reviews. Sentiment analysis aims to find the attitude of the writer from the review text. Sentiment analysis is identifying the sentiment content of the text using NLP or Machine Learning techniques. This is also referred to as opinion mining. Our task was to analyze the sentiment of these reviews and depending on the sentiment, classify the review into one of the 5 categories, where rank 1 is a category; rank 2 is a category etc. The classification is based on the sentiment scores calculated through NLTK and Machine learning techniques.

Methodology

We followed three approaches to predict the ratings from the review text. The approaches are listed below:

1. Supervised Learning: The problem to predict ratings from review text was a multi-label classification problem. So we used 3 algorithms to predict the ratings: Multinomial Bayes, Support Vector Machines and Support Vector Regressions. These algorithms were available as part of the **Python SciKit-Learn Library**.
2. Supervised Learning: The next approach is also part of supervised learning but differs slightly from the first approach. We use the **NaïveBayesAnalyzer,** which is an NLTK classifier trained on movie review corpus. The **NaïveBayesAnalyzer** returns its result as a tuple in the form: Sentiment (classification, p_pos, p_neg)
3. The third approach is kind of a semi supervised learning approach using Stanford NLP parser and SentiWordNet library.

The diagram given below gives an overview of Supervised Learning approach:



We will now look into each of the above-mentioned approaches in details below:

Supervised Learning: Multiple Algorithms

Prediction of user rating based on the review text falls under the category of multi-class classification. Multi-class classification is a task with more than two classes. There are various classification algorithms available. These algorithms predict the label for a given input sentence. In supervised classification, the classifier is trained on a labeled set of examples that are similar to the test examples. Under this category, we have considered three algorithms: Multinomial Bayes, Support Vector Machines and Support Vector Regression.

1. Multinomial Naïve Bayes classifier is suitable for classification where we use word counts or word frequency for text classification. The multinomial distribution normally requires integer feature counts but also works with fractional counts such as TF-IDF
2. Support Vector Machines: Support Vector Machines are a set of algorithms used for classification and regression. Support Vector Machines are suitable were dimensional space is high and is memory efficient. LinearSVC class in SciKit-Learn is used to perform multi-class classification on the dataset.
3. Support Vector Regression: Support Vector Regression is a SVM approach extended to solve regression problems. In our task we use SVR with linear kernels, which is part of SciKit-Learn.

For each of these algorithms, we have used TF-IDF to get the TF-IDF vector for a single document. Once we have got the TF-IDF vector for the single document, which in our case is a single review text, we the classifier to classify the text. We have used 80 percent as training data and 20 percent as testing data.

We have used 10-fold cross-validation on out sample dataset.

Evaluation Metrics

Accuracy: Accuracy is a simple measure for evaluation of classifiers. It is the percentage of reviews that were correctly classified over the reviews in the test data set.

Baseline

The reviews were rated on a scale of 1 to 5, i.e., the number of category labels is 5. So the base line accuracy is 20%.

Experimental Results

To get a sense of accuracy of the classifiers, we created 7 sample datasets and ran the classifiers on these sample datasets. The results are shown below for your reference:

Number of reviews	Multinomial Bayes	Support Vector Classification	Support Vector Regression
1000	50.8	51.1	60.4
2000	53.35	53.85	60.7
3000	54.5	52.3	62.2
4000	54.15	53.6	63.8
5000	53.64	54.72	64.6
6000	51.9	54.2	64.68
7000	47.5	52.29	63.5

Analysis

From the above table, we can infer that Support Vector Regression, which is part of Support Vector Machines perform well in this classification task. These results are based on unigrams. It would have been more interesting, if we had time to perform the experiment based on bi-grams and tri-grams.

Supervised Learning: Naïve Based Sentiment Analyzer

Under this approach, we used the Naïve based Sentiment Analyzer which comes as a part of the NLTK Classifier trained on movie review corpus.

Evaluation Metrics

Accuracy: Accuracy is a simple measure for evaluation of classifiers. It is the percentage of reviews that were correctly classified over the reviews in the test data set.

RMSE: Root Mean Square Error is used to measure the difference between value predicted by the model and the values actually observed.

Baseline

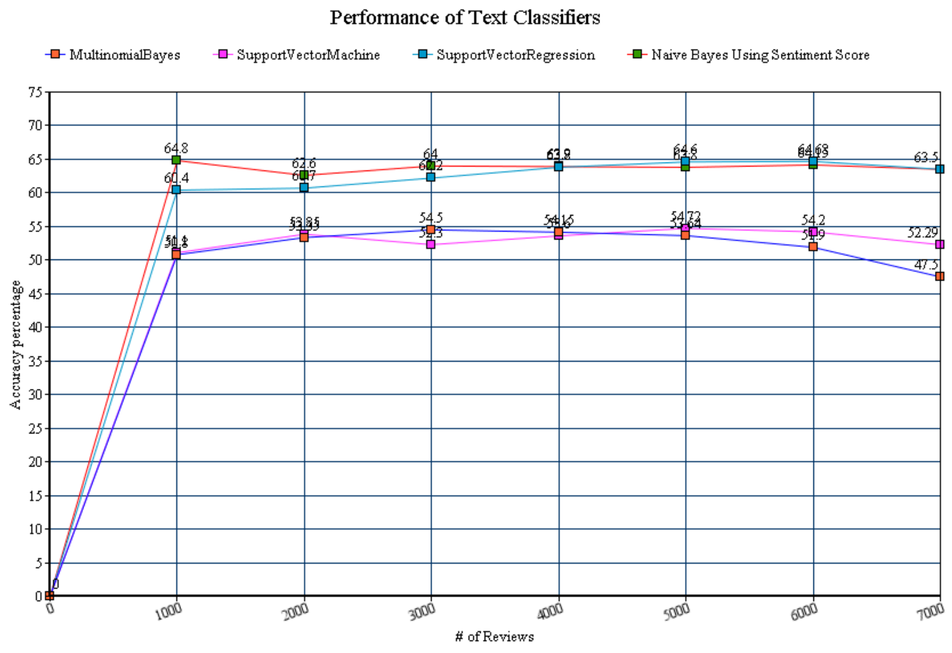
The reviews were rated on a scale of 1 to 5, i.e., the number of category labels is 5. So the base line accuracy is 20%.

Experimental Results

To get a sense of accuracy of the classifiers, we created 7 sample datasets and ran the classifier on these sample datasets. The results are shown below for your reference:

Number of reviews	Accuracy	RMSE
1000	64.8	1.86
2000	62.6	1.89
3000	64	1.86
4000	63.9	1.85
5000	63.6	1.85
6000	64.68	1.88
7000	63.5	1.86

The graph shown below gives a comparative overview of how the algorithms perform against each other:



Un-Supervised Learning: SentiWordNet and Stanford NLP Based Classifier

Methodology

SentiWordNet is a lexical resource for opinion mining. By default it comes with a corpus which is a list of synsets of WordNet with each synset assigned to a part of speech and two sentiment scores: positivity, negativity.

Steps Involved

1000 of the 86759 reviews was used.

1. The words in the review were tagged according to its part of speech using Stanford NLP part-of-speech tagger.
2. SentiWordNet is used to calculate the positive and negative score for each tagged word in the review.
3. For each word the positivity (positive score – negative score) is calculated.
4. The top 20 words in each review with maximum positivity score were used to determine the sentiment score of that review.

The following table is used to calculate the rating.

Sentiment Score	**Rating**
>= 0.75	5
>=0.55 & < 0.75	4
>=0.35 & < 0.55	3
>=0.15 & < 0.35	2
The rest	1

The following table gives an overview of results:

Accuracy	23.08%
Baseline Accuracy	20%
After accuracy adjustment	57.6%
RMSE or RMSD	1.80

GitHub Link

Source code for the work is available at https://github.com/divlalwani/InfoRetrievalYelp

Conclusion

The Yelp dataset was used for two tasks: categorizing business based on review contents and predicting user rating based on review contents. Various Machine-learning algorithms as well as Sentiment based algorithms were used in the process. Libraries from Weka, SciKit were used in the process. MongoDB was also used to load the dataset and this step simplified the task of querying the dataset.

For Task I, Multinomial Naïve Bayes algorithm seems to perform well whereas for Task II, SVM based Support Vector Regression algorithm performed well.

References

1. Naïve Bayes Text Classification: http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html
2. User Review Rating Prediction: http://cs229.stanford.edu/proj2012/LeongBhatia-UserReviewRatingPrediction.pdf
3. Weka: http://www.cs.waikato.ac.nz/~eibe/pubs/kibriya_et_al_cr.pdf
4. SciKit Support Vector Machines: http://scikit-learn.org/stable/modules/svm.html
5. SciKit Naïve Bayes Classification: http://scikit-learn.org/stable/modules/naive_bayes.html
6. TextBlob Sentiment Analyzers: http://textblob.readthedocs.org/en/latest/advanced_usage.html
7. Sentiment Analysis of Movie Review Comments: http://people.csail.mit.edu/kuat/courses/6.863/report.pdf
8. NLP Stanford Book – SVM: http://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-and-machine-learning-on-documents-1.html