**Sentiment Analysis on Product Reviews**

Sahithi Desu Ankit Kumar

[sldesu@syr.edu](mailto:sldesu@syr.edu) [akumar15@syr.edu](mailto:akumar15@syr.edu)

MS Computer Sc. MS Computer Sc.

**Abstract**

With the explosion of social networking sites, blogs and review sites a lot of information is available on the web. This information contains emotions and opinions about various product features and the makers of these products. This form of opinion and feedback is important to the companies developing these products as well as the companies that want to develop better rival products. Sentiment Analysis is the task of analyzing all this data, retrieving opinions about these products and services and classifying them as positive or negative, in other words good or bad. The key parts of any review of any product are the numeric rating and the textual description provided along with this product. In our project we will take into consideration both these vectors for product reviews to conclusively decide on a classifier that is best suited to analysis of product reviews. We have gathered reviews and based on the features that best describe the sentiment for each review, we have created a feature set of 1000 features, and with this limited set we will determine which classifier gives the best result on review type data. To determine the best classifier we perform evaluations on it, by running various data set generators, calculating the resubstitution and generalization errors for each classifier. We then use the mean of these results to compute the paired Student’s t-test to relatively compare the performance of the classifiers. Based on the results of this evaluation, we can state which is the best classifier.

**Approach**

We followed a simple approach to train a classifier and run the tests on it. We have used Document Level Sentiment Analysis to determine the opinion on single entity(product).

The steps we followed are as discussed below,

1. **Gather data:** In this step, we wrote a web crawler targeted at amazon. The web crawler was responsible for gathering all the data from amazon's product reviews page. We provide a product id to the program and this program randomly opens pages and scrapes the pages for the product review, along with the rating of the review. The reviews we collected are stored along with a class label for each review, all the reviews, which get a rating of 3 or more stars were classified as positive and all the ratings which got a review score below 3 were classified as negative. By doing this we were able to create a training set, that has been classified by real humans and thus can be assumed to be accurate. Later, we will see that, we have used this metric to evaluate our classifiers accuracy.  
   While creating this web crawler, we used Jsoup to grab the webpage and identified the different elements we needed to get the reviews and the rating provided by a user. The challenge faced was to circumvent the various robot/crawler prevention techniques applied by amazon, such as honey pots, user-agent checking and robots.txt files. We did this by doing a random walk of the product review pages and continuously requerying pages when we received forbidden 403 error. In this way we collected over 5000 reviews of a product and applied the class labels to the reviews based on the star rating provided by the reviewing users.

1. **Pre-process data:** In this step, we decided to process the data before we are able to extract the features. The various pre-processing steps we applied are,
   1. **Tokenization:** We tokenized the reviews using the tweet-tokenizer. The reason for this approach as opposed to splitting the sentences to tokenize them is clear when we compared the content of the reviews and the twitter. We found striking similarity. The only real discerning feature we came across was that while tweets are limited to 140 characters, reviews have no such limitation, this can be easily over-looked as it doesn’t have any affect on our feature-set generation.
   2. **Remove Punctuations:** We remove the punctuations from the reviews, this is done so that we do not have any unwanted punctuations in our resultant feature-set.
   3. **Remove Stop-words:**  This is a basic requirement, so that we can focus on words that are actually relevant to the document instead of, determiners, prepositions and coordinating conjunctions, which can appear a number of times in any given training set, if not removed. Removal of stopwords is crucial to supervised learning.
   4. **Spell-check:** We perform spell-check on the documents(reviews) to ensure that the feature-set being generated has relevant words and not commonly misspelled words, apart from this, spell-check allows for accurate frequency calculation, which is crucial when the basis of the feature-set generation is frequency distribution over the set of processed documents. To accomplish this we have a big.txt file which consists of about a million words. The file is a concatenation of several public domain books from Project Gutenberg and lists of the most frequent words from Wiktionary and the British National Corpus. We then extracted the individual words from the file and trained a probability model(based on occurrence of each word). After this, we implemented Smoothing over the parts of probability distribution, that would have been zero(words that have not occurred in the *big.txt* file) by bumping them up to the smallest possible count. We performed this process of spell-checking two times using an edit-distance of 2, this was done after analysing that spell-checking twice gives the best result.
2. **Feature-set generation (Using Bag-of-words method):** In NLP and information retrieval, bag-of-words is used as a simplified representation. Here, a text is represented as a bag(multiset) of its words, disregarding the word order and the grammar associated with the text. To generate the feature-set we have considered using two techniques, tf-idf and term frequency. Upon analysis, we noticed that tf-idf based feature extraction results in removal of words important to the classification of text as positive or negative. Tf-idf ends up penalising words that are crucial for the definition and also appear a large number of times in the document. One instance in our study set that we noticed was for the word “great”, the word great occurred 786 times, whereas the word “of” occurred 745 times. If we used tf-idf, we will end up removing “great”, which is key in defining what a user thinks of an application. The alternative to this approach is the frequency distribution method for generating the feature-set. After removal of stop-words, this method gives a feature set that appears to be very similar to a good feature set.
3. **Classification**
   1. **Building classifiers**
      1. **K- Nearest Neighbor:** K-nearest neighbours is one of the classification algorithms that trains itself by using similarity measures within its boundary.

* Parameters: Here in our Evaluation, we have tuned the parameters of this classifier by giving different values of k. Since we are dealing with two-class problem we have chosen to give odd values to k. We assigned values of k as 3, 5 and 9.
  + 1. **Naive Bayes:** It is a classification technique based on Bayes’ theorem with an assumption of independence among predictors. It is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of [feature](https://en.wikipedia.org/wiki/Feature_vector) values, where the class labels are drawn from some finite set. It is not a single [algorithm](https://en.wikipedia.org/wiki/Algorithm) for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is [independent](https://en.wikipedia.org/wiki/Independence_(probability_theory)) of the value of any other feature, given the class variable.
    2. **Decision Trees:** Decision trees are powerful tools for classification and prediction. They represent rules, which can be understood by humans and used in knowledge system such as database. The following are the key requirements:
* Attribute-value description: Object or case must be expressible in terms of a fixed collection of properties or attributes for e.g., hot, mild, cold.
* Predefined classes (target values): The target function has discrete output values for e.g.,boolean or multi class.
* Sufficient data: Enough training cases should be provided to learn the model.
* Parameters: We have tuned the parameters of this classifier by varying the min depth as threshold. The values for the min depth here are 2,5,10,30.

* + 1. **Support Vector Machine:** This algorithm is mostlyused in classification problems where each data item is plotted as a point in n-dimensional space where n represents number of features we have, with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper -plane that differentiate the two classes very well.
* Parameters: We have used RBF kernel and as well as Linear Kernel to tune our Parameters by setting c values as 1 and 5.

1. **Evaluation:**

**Why Evaluation?**

One should focus on the Predictive capability of the classifier rather than how fast it takes to classify or build models, scalability etc. Most widely used metric evaluation is accuracy. Accuracy is defined as follows:

"Accuracy"="a+d" /"a+b+c+d" ="TP+TN" /"TP+TN+FP+FN"

But there are limitations for this. We can’t just rely on this to perform evaluation on our classifier to pick the best one.

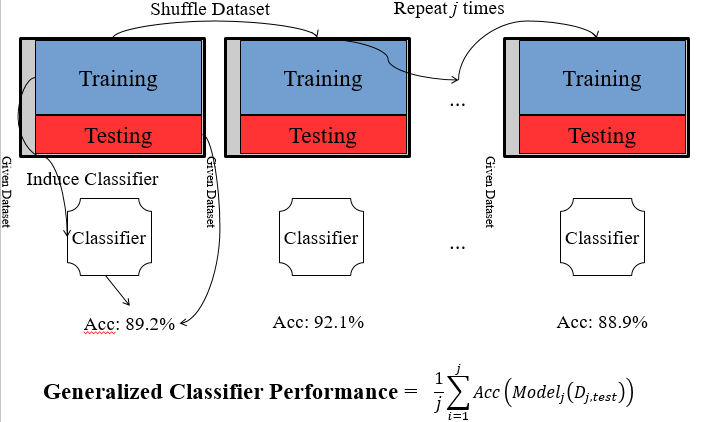
For example: Consider a 2- class problem with number of class labels 1 = 9990 and number of class labels 0 = 10. Then the model always predicts the class label as 1 and the accuracy of the classifier is always 9990/10000= 99% . Accuracy here is misleading because the model here does not predict the class 0 labels.

Performance of the classifier depends on many other factors apart from the learning algorithm.

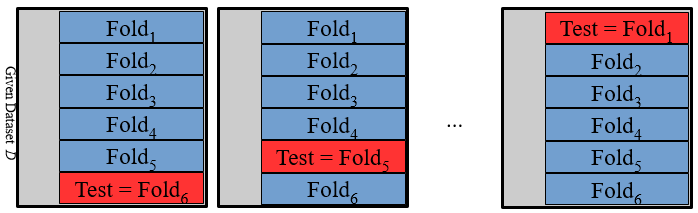
* Cost of misclassification
* Class distribution
* Size of training and testing sets

**Methods of Estimation:**

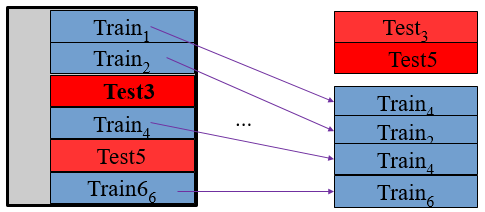
* **Hold Out Generator:** Here we reserve 3/4th of the data as the training data and ¼ th of the data as the testing data and evaluate the performance of the classifier using the testing data.



* **Cross Validation:** We have chosen 10 fold and stratified sampling to perform cross validation. Broken down the Data set ‘D’ into 10 equal folds and repeated the classifier building process 10 times. Each time we have left one fold for testing. We have combined all these partitions into random shuffles and calculated the average of the errors. By doing stratified sampling we made sure that class distribution in any fold set is similar to entire data set. It is useful to avoid building classifier that has never seen a particular class label.



* **Resample Generator:** Here we followed Random Sampling with substitution method. This always creates new testing set and training set by sampling data from the original data set ‘D’. On every run we took 0.623 fraction of data as our training set by randomly picking instances from the data set ‘D’ upto the specified fraction and remaining samples as testing set. This method is also known as 0.623 Bootstrap generator.



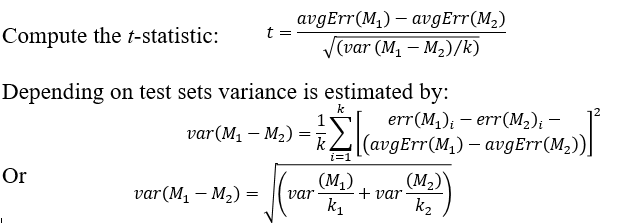
For each and every classifier along with tuning their parameters we had run all these generators mentioned above on every classifier. Each combination of classifier and generator was run for 5 times and each time we have done random shuffling in order to make sure that the data has been distributed evenly on every run so that there won’t be any bias in evaluating the performance of the classifier.

* 1. **Average Re-substitution errors:** This measure defines the training error rate in the data with certain classifier. It is calculated by finding the number of wrongly classified instances divided by total number of training instances.

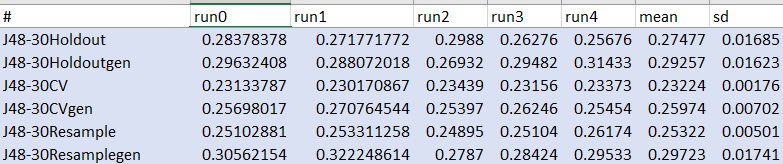
*Re-substitution error = wrongly predicted instances /total number training instances*

* 1. **Generalization errors:** This is a metric to measure the error rate on testing on a particular classifier. It gives the testing error in the data. We have calculated this using “getEvaluation” method in Weka library(genErr=getEvaluation().errorRate)

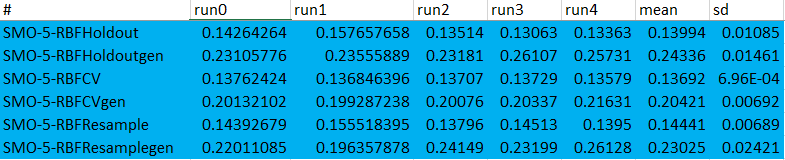
The Mean and Standard Deviation is taken of these resubstitution errors and Generalization errors over 5 runs.

1. **Picking a winner:**
   1. **Paired Student’s-t test:** This is a hypothesis test where we do null hypothesis on two classifiers that they are same and prove the hypothesis to be true/false based on the value generated by the t-test.   
      Just depending on the Mean values of the resub and gen errors for the classifiers is not sufficient in determining which classifier performs the best. For an instance, the mean values for two classifiers M1 and M2 are as follows. M1\_mean =0.933 and M2\_mean = 0.944. With this information we can’t conclude that M2 is better than M1 since these are experimental results. We need some kind of test statistic that can be used to determine if the two sets of data are significantly same or different from each other. The test statistic we used here is student t-test.   
      Given the value of t computed, it computes the degree of freedom (k-1) or min(k1-1,k2-1).Given the value of p (the chance that the results are likely to be from the same process) our t value should be less than 0.05 in order to accept the null hypothesis that we have made. The p value must be selected based on the domain knowledge. Usually p=0.05 is acceptable in Computer Science field.   
      We have used Weka API in our java program to compute the Student t-test.
   2. **Analysis for Picking Good Classifier:**Stored all the results of the evaluation into csv file and compared the Mean and standard deviations of resubstitution and generalization errors for all the combinations of classifiers with generators.   
      Based on these results we have picked SMO-RBF Kernel with C=5,SMO-RBF Kernel with C=1, SMO-linear Kernel with C=1, Naive Bayes, J48-30 classifiers.  
      To make sure that those results are true we have compared them using the t- values that have been generated using student t-test.   
      We picked the ones whose p value is close to 0 i.e lesser than 0.05 and ignored the ones whose value is greater than 0.05.  
      After doing this analysis we have concluded that SMO-RBF kernel with C=1 and Naive Bayes classifiers are best for this data.   
        
      **Here are screenshots of Mean and Standard Deviation of Resubstitution and Generalization errors.**

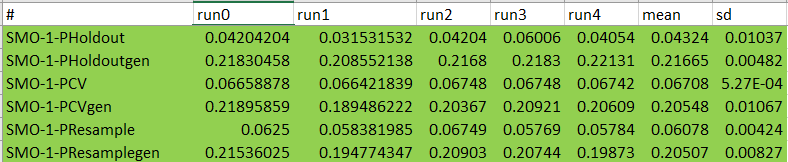
For Decision Tree with minimum depth as 30:



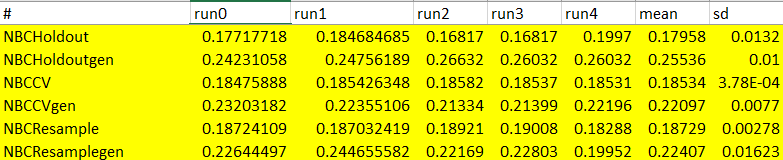
For SMO RBF Kernel C=5.0



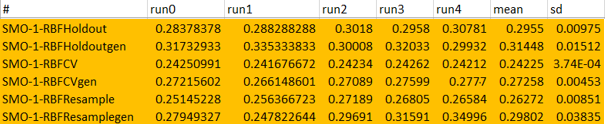
For SMO Linear Kernel C=1.0



For Naive Bayes



SMO RBF Kernel with C = 1.0



**Screenshots for Student t-test:**

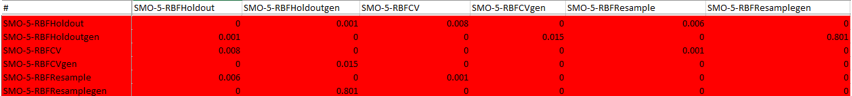
* For Decision Tree with minimum depth as 30:

t-test11.png

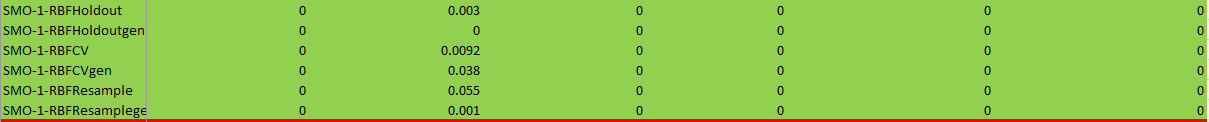
* For Naive Bayes

ttest2.png

* For SMO RBF Kernel C=5.0



* For SMO RBF Kernel C=1.0



**Experimental Results**After building an adequate method for feature-set extraction and picking classifiers training them and using student-t test to get the best classifier, we ran the trained classifier on the data set that we gathered from facebook, yahoo and viber reviews on the amazon app store.

The following are the results gathered,

Facebook review set,

Using Naive Bayes classifier - 89%

Using Support Vector Machines with RBF Kernel(c=1) - 72%

Yahoo mail review set,

Using Naive Bayes classifier - 81%

Using Support Vector Machines with RBF Kernel(c=1) - 70%

Viber review set,

Using Naive Bayes classifier - 71%

Using Support Vector Machines with RBF Kernel(c=1) - 87%

**Conclusion:**

After carefully analyzing the results of the experiments performed, we have concluded that for app reviews, we can use, Naive Bayes classifier and Support Vector Machines with RBF Kernel(c = 1)