Abstract - Sentiment Analysis of reviews in marketing area help in making the best decisions regarding the product. Here, in this project, I am estimating the sentiment of movie reviews either positive or negative score taking IMDB dataset into consideration. I have used Nave Bayes classifier and Decision Tree for text classification on different sample sizes and compared the Accuracies taking single words and Bigrams into considerations. I have also evaluated Evaluation methods like positive precision and positive recall, negative precision and negative recall for Nave Bayes and Decision Tree on different sample sizes.

Sentiment Analysis of Movie Reviews

Kiran Patrudu Gopalasetty

1 Introduction

Nowadays, automated rating and recommendation systems are ubiquitous. Whether youre looking for a movie to watch on Netflix, shopping online through Amazon, a learning algorithm is helping you make the best decision. In order to do that, they need to predict the overall sentiment of different reviews and combine it with your preferences to determine how to present the best options. Now consider a company about to launch a new product. Before a general release, they want to survey potential users to understand expectations. In order to process large quantities of survey responses, sentiment analysis can be used to gauge response to their new product. Being able to quickly and efficiently parse through and predict user sentiment on such a wide range of topics is a problem that both industry and academia have been attempting to solve for some time. Methods used have ranged from Bayes models, to support vector machines, to neural networks with varying kinds and numbers of feature extractions.

With many different attempts at sentiment classification, there is always the trade off between a fast prediction and an accurate prediction. Simple models that train quickly often do not predict as well as more complex models that take longer to train. In this project, I presented an efficient and relatively accurate classification method specifically trained for movie review sentiment prediction which leverages a Naive Bayes classifier. I have also presented the work with Decision Tree classifier and compared with the results of Naive Bayes classifier.

mds
August 26, 2015

2 Goal

To classify the sentiment expressed in movie reviews to either positive or negative using natural language processing techniques.

3 Dataset Sources

I have collected data from IMDB movie reviews dataset for binary sentiment classification which provide a labelled dataset of 25,000 highly popular movie reviews for training, and 25,000 for testing. I have extracted rating and review from the dataset and randomized them for better results. The fields in our final dataset are: ID: Unique id of each review. Rating: Rating given by each user (rating¡=5 with sentiment score 0 and rating¿=7 with sentiment score 1). Review: Text of reviews given by user. Sentiment: Sentiment of the review. 1 for positive reviews and 0 for negative reviews.

For instance, a positive review will be like Bromwell High is nothing short of brilliant. Expertly scripted and perfectly delivered, this searing parody of a students and teachers at a South London Public School leaves you literally rolling with laughter. Its vulgar, provocative, witty and sharp. The characters are a superbly caricatured cross section of British society. negative review be like, Story of a man who has unnatural feelings for a pig. Starts out with an opening scene that is a terrific example of absurd comedy. A formal orchestra audience is turned into an insane, violent mob by the crazy chanting of its singers. Unfortunately, it stays absurd the WHOLE time with no general narrative eventually making it just too off putting. Even those from the era should be turned off.

4 Approach

I have considered Nave Bayes classifier and Decision Tree classifier for Text classification of movie reviews. Below, I will be explaining about each approach implementation in the project.

4.1 Naive Bayes

I chose to use the Naive Bayes model in order to predict sentiment in movie reviews. Bayes Theorem states that for independent features B the following is true:

For our problem, A would be the sentiment of the review, and B would be the features of the review. A Naive Bayes model implements just this simple equation to determine the probability of a label given a vector of features based on the probabilities of that feature given the label in a collection of training data. It is termed as naive because of the assumption

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

that every feature is independent. While this is not the case in practice, it has been shown that even with this assumption accurate predictions can be made using this model.

4. TTraining and Classifying

In order to create the classifier, first I have processed the labeled training reviews and extracted the feature sets out of them. These features include words and bigrams (2 words that appear consecutively in a string) that have been preprocessed to determine the best to use. Every feature is Boolean in nature, and the feature set of the review determines which features exist for that review. The classifier is trained by collecting every feature that is seen in every review and creating a frequency distribution conditional on the sentiment label. This accounts for (P (B, — A) in Bayes Rule which is calculated by dividing the frequency of the feature occurring with the total count of the label. In addition, a frequency distribution of the label itself is also created accounting for P(A). As P(B) is constant with respect to different labels, we can discount it for the model. A portion of the final feature collection of the model could be [awesome, best, awful, (worst, movie)] For every feature in the feature collection, the associated conditional probability distribution of that feature occurring given a label is stored. Once the model is trained, it can be used to classify new reviews. The classifier uses the same feature extraction process on a new review that it used for training, yielding a collection of words and bigrams. The classifier then uses Bayes rule to calculate the probability of every sentiment label given the feature set of the new review i.e.

$$P(sentiment \mid features) = P(sentiment) \prod_i P(feature_i \mid sentiment)$$

where,

(P (featurei, — sentiment)

is the conditional distribution for each feature and P(sentiment) is the prior distribution stored in the training phase. The maximum

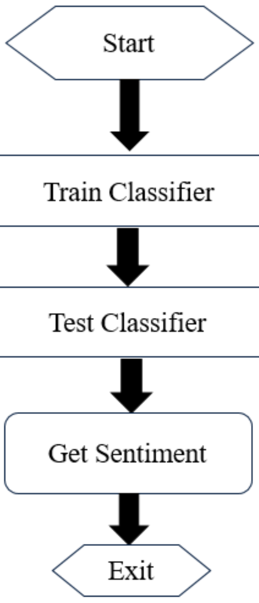
(P (sentiment, — feature)

for all possible sentiment classifications is then chosen as the predicted sentiment. However, due to the very small conditional probabilities, taking the product is computationally expensive and risks floating point errors. Thus, we actually maximize the log probability rather than the probability directly. This results in an equation as follows

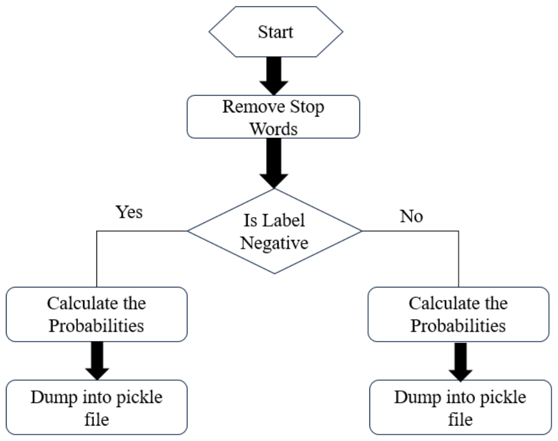
$$logP(sentiment \mid features) = logP(sentiment) + \sum_i logP(feature_i \mid sentiment)$$

As the logarithm is a monotonically increasing function, maximizing the log probability will yield the same result as maximizing the probability itself, and our classifier will be faster and more accurate at computing a sum instead of a product.

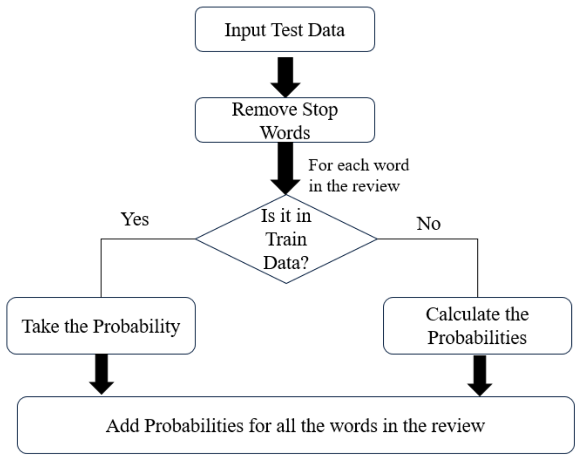
Project Flow:

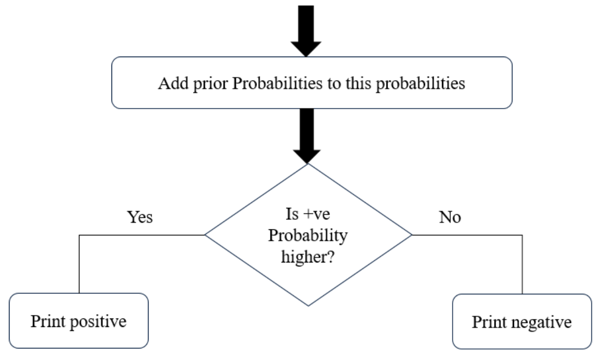


Train Classifier:



Test Classifier Get Sentiment:


Input Test Data
Remove Stop Words
For each word in the review
Is it in Train Data?
Yes
No
Take the Probability
Calculate the Probabilities
Add Probabilities for all the words in the review


Add prior Probabilities to this probabilities
Is +ve Probability higher?
Yes
No
Print positive
Print negative

4.2 Decision Tree

Decision trees create a flowchart based classifier. At each level it utilizes decision stumps, a simple classifier that check for the presence of a single feature. The label is assigned to the sentence at the leaf nodes of the tree.

The decision tree is built by recursively partitioning the feature space into two parts. At each step, the split that improves the error of the tree on the training data is used, and this greedy strategy continues until a tree of desired size is produced. For new data, a label is predicted by following the branches of the tree from the root node according to the values of the features of the new data. Decision trees have the advantage of being very easily interpretable, as the binary tree structure can be represented in a drawing, and a human being can follow the branches down the tree according to the input variables.

5 Resources

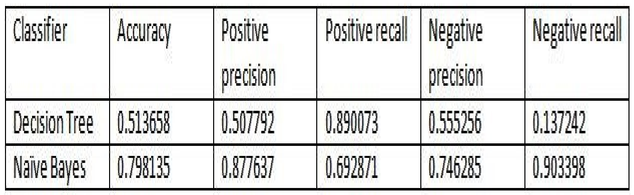
In order to carry out the feature extraction process and create the classifier, we leverage the Python Natural Language Toolkit (NLTK). The NLTK is a free and open source Python library of functions for parsing text and creating statistical analyses. Using NLTK, we created custom text parsers for feature extraction including stemming, finding bigrams. We are using Naive Bayes classifier and Decision Tree classifier in NLTK with significant performance improvements accomplished by limiting it to Boolean feature distributions and stripping out unnecessary abstractions.

6 Evaluation

In the preprocessing stage I have removed punctuation, removing numbers, best words, weighted best words, bi- grams, best word bigrams, trigrams, and stemming. I tried removing punctuation and digits from the dataset theorizing it would help remove any false positives as they are not tied directly to sentiment in writing. However, this did not increase the accuracy as I found that the data had people rating the movie in the review (for example the string I give this movie 4/10 stars). So instead of removing digits and punctuation, I used it to help gauge the sentiment of the review. As stated above bigrams are extremely helpful as the more words you add to search the more sentiment you can extract. We then got the positive and negative words from the reviews matched with the dictionary model. Using these positive and negative words I have trained our model for 2000 reviews and tested it for 1000 reviews sample for both Nave Bayes classifier and Decision Tree classifier.

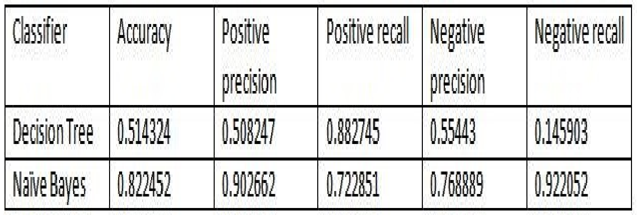
7 Experimental results

I have tested each feature extraction by varying multiple parameters including size of training set, number of bigrams and types of words to use. I then compared the accuracy for Nave Bayes and decision tree by taking different sample sizes. I have noticed that when using 1500 samples for training and 1500 for testing, there was no considerable difference in the accuracy of single words and bigrams for Decision Tree and even for the Nave Bayes the above-mentioned accuracy difference was not much. But, as I increased the sample size to 2000 reviews, I have noticed that there was a noticeable difference in accuracy when using single words and bigrams in both the classifiers.



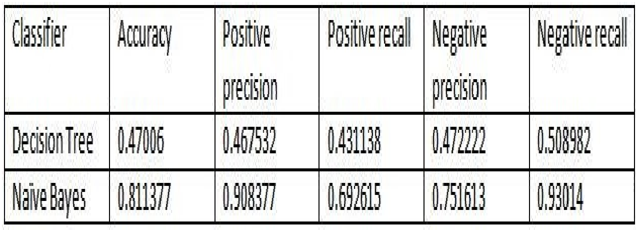
Classifier	Accuracy	Positive precision	Positive recall	Negative precision	Negative recall
Decision Tree	0.513658	0.507792	0.890073	0.555256	0.137242
Naïve Bayes	0.798135	0.877637	0.692871	0.746285	0.903398

Figure 1: Single word feature results for 1500(train and test) sample set



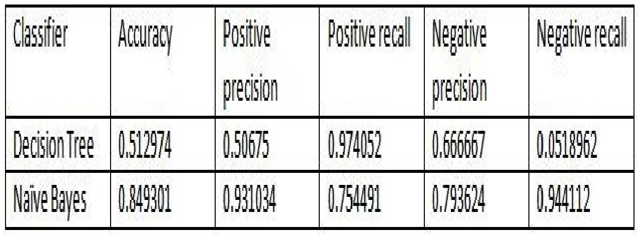
Classifier	Accuracy	Positive precision	Positive recall	Negative precision	Negative recall
Decision Tree	0.514324	0.508247	0.882745	0.55443	0.145903
Naïve Bayes	0.822452	0.902662	0.722851	0.768889	0.922052

Figure 2: Bigram results for 1500 (train and test) sample set



Classifier	Accuracy	Positive precision	Positive recall	Negative precision	Negative recall
Decision Tree	0.47006	0.467532	0.431138	0.472222	0.508982
Naïve Bayes	0.811377	0.908377	0.692615	0.751613	0.93014

Figure 3: Single words Features results (2000 train and 1000 test)



Classifier	Accuracy	Positive precision	Positive recall	Negative precision	Negative recall
Decision Tree	0.512974	0.50675	0.974052	0.666667	0.0518962
Naïve Bayes	0.849301	0.931034	0.754491	0.793624	0.944112

Figure 4: Bigrams results (2000 train and 1000 test)

I even calculated measures such as positive precision, positive recall, negative precision and negative recall for sample sizes 1500 and 2000. Here, I have noticed that positive precision, positive recall, negative precision varied considerably for decision tree classifier whereas as the effect was nominal for Naive Bayes.

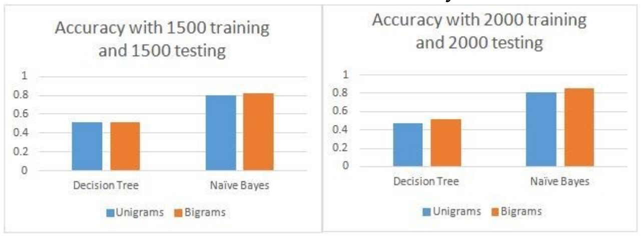


Figure 5: comparison of accuracy between Naive Bayes and Decision Tree for sample size 1500 and sample size

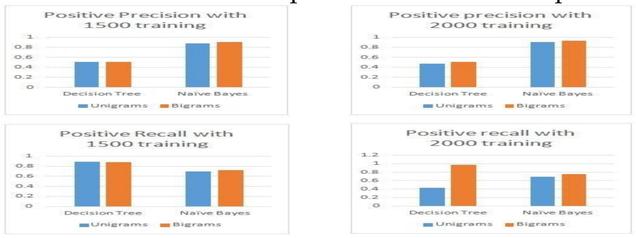


Figure 6: comparison of positive precision and positive recall for sample size 1500 and sample size

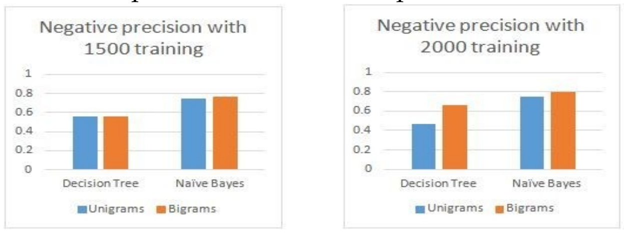


Figure 7: comparison of negative precision for sample size 1500 and sample size

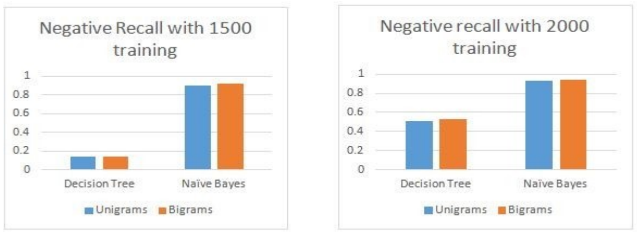


Figure 8: comparison of negative recall for sample size 1500 and sample size

From the above results, we can conclude that Naive Bayes classifier performs much efficiently than decision tree classifier. The accuracy obtained with the Nave Bayes classifier for 2000 sample set is approximately levelled to 0.8 where as for the Decision Tree classifier is of 0.52.

8 Conclusion

I have presented sentiment analysis of movie reviews by Nave Bayes and Decision Tree classifiers and I saw that Nave Bayes out-performs Decision Tree clearly. I also saw that use of different sample sizes for training and testing tests affects the accuracy of the classifier. I have also noticed that by applying different feature extraction techniques such as bigrams the accuracy of the Nave Bayes and Decision Tree classifier increased to 0.82 from 0.79 and 0.5143 from 0.5136 respectively. To increase accuracy, further work can focus on (1) more efficient test pre-processing techniques, and (2) different combinations of feature extraction techniques.

References

[1] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (HLT 11),PA, USA, 142-150.

[2] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10 (EMNLP 02),PA, USA, 79-86.

[3] Chai, K.; H. T. Hn, H. L. Chieu; Bayesian Online Classifiers for Text Classification and Filtering, Proceedings of the 25th annual international ACM SIGIR conference on Research and Development in Information Retrieval, August 2002, pp 97-104

[4] K.-M. Schneider. 2005. Techniques for Improving the Performance of Naive Bayes for Text Classification., in Alexander F. Gelbukh, ed., CICLing , Springer, pp. 682-693.

[5] Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In ACL, pages 115124.

[6] Thorsten Joachims, Text Categorization with Support Vector Machines: Learning with Many Relevant Features,