Classification of review sentiments

Rediet Tilahun Desta
Department of Electrical Engineering
rtilahun@princeton.edu

Abstract

In this age where we mostly buy products be they clothes or music, we would like to get the opinions of others before making the decision. However going through all reviews can be time consuming. Companies that faciliate this trading process would like to present their customers with a quick and easy way to gauge the overall response to the product; the first step of which will be analyzing the sentiments of each reviews. In this assignment, I attempt to find a good sentiment classifier which would classify reviews as either positive or negative, using a data set of 3000 reviews. I employed Natural Language Processing(NLP) techniques to represent the data as a bag-of-words which were then used to train six types of classifiers. Almost all classifiers performed really well: having similar and high values of precision and recall. The real difference was in the computational time at which multinomial and bernoulli naive bayes classifiers significantly outperformed the rest more complex classifiers while still maintaing a good precision and recall value.

1 Introduction

Sentiment analysis used to be a very human task in the past. However in recent years, with the growth of online companies, the need for automated forms of sentiment analysis were found to be of most importance. Online companies like Amazon, Facebook, Twitter and Yelp are increasingly becoming dependent on machine learning algorithms to classify the sentiment of the billions of textual inputs they receive daily. In this assignment, I am interested in learning which type of classifiers perform well in classifying the sentiments of online reviews. Moreover, I want to gain the understanding of why certain classifiers perform well while others less so.

I evaluate 6 types of classifiers in this report. In the feature extraction process, I chose unigrams and then in the feature selection component selected those unigrams that appear more than 5 times in the 2400 data set. Other feature selection techniques were also used and then compared as to their impact on classifier performance such as computational speed, precision and recall.

2 Related Work

The first indepth analysis of the field of sentiment analyis and classification was done by Pang et. al in 2008. [1] In trying to understand the sentiment expressed in a document one can dissect the problem to understanding the sentimental value of the words in that document and the sentences that make up that document. Moreover, one needs to distinguish between words and sentences that matter and those that don't matter. In the literature, these distinctions are known as subjectivity classification. [2] Tang clearly states that, "Document sentiment classification and opinion extraction have often involved word sentiment classification techniques." [2]For example, Pak et. al. use a sentiment classifier that is "based on the multinomial Naive Bayes classifier that uses N-gramand POS-tags [part of speech tagging] as features." [3]Zhang et al. found that, "accuracy is influenced by

interaction between the classification models and the feature options...Character-basedbigrams are proved better features than unigrams and trigrams in capturing Cantonese sentimentorientation." [4]

2.1 Data processing

I initially downloaded two data sets marked train.txt and test.txt. train.txt contained 2400 reviews, each marked with a 1 or a 0 by a human agent: 1 indicating a positive review and 0 a negative review. test.txt contained 600 distinct reviews from the same source. By using python's NLTK library, we were able to tokenize each document, convert it lower case words then remove words that were stop words, lemmatize and then stem it. After that we also took out words that did not appear more than 5 times in the whole data set. This resulted in a vocabulary set of size 541 words. These 541 words were used as features of each document in a bag-of-words format; their presence in the review being marked by 1.

2.2 Classification Methods

I used 6 different classification methods using Python's SciKitLearn libraries. [5]

1. *Multinomial Naive Bayes classifier* (MNB): using multinomial implementation
2. *Bernoulli Naive Bayes classifier* (BNB): using bernoulli implementation
3. *Support vector machine* (SVM):
4. *Decision tree* (DT):
5. *Random forest* (RF): using 100 trees
6. *K-nearest neighbors* (KNN): using ten nearest neighbors and the "KDTree" algorithm

Each classifier was first trained using the dataset found in train.txt. Python's SciKitLearn library provides functions that fit the features of dataset with the predetermined classes for each classifier type. However to use random forest classifiers and k-nearest neighbours classifiers, we needed to fit their hyperparameters. Hence, we used 10-fold cross validation on the training dataset and determeined the n-value which gave us the best accuracy. These n-values were then used as hyperparameters for classifiying the test data set using a random forest classifier or k-nearest neighbours.

2.3 Evaluation

We evaluated the six classification methods we selected using five different metrics. We first combined the test and training datasets.Then we used 10-fold cross validation technique on the dataset to iteratively train and test portions of overall data set. This technique guarantees that each document in the data set will be used as a testing document once. After the iteration completes we began to compare the predicted class with the predetermined class of the whole combined data set using the five aforementioned metrics. These metrics are accuracy, precision, recall, F_1 score, and time.

Below is brief definition of the five metrics:

1. $Accuracy = \frac{Numberofaccuratepredictions}{Numberofdocuments}$
2. $Precision = \frac{TruePositive}{TruePositive + FalsePositive} = 1 - FalseDiscoveryRate$
3. $Recall = \frac{TruePositive}{TruePositive + FalseNegative}$
4. $F_1\ score = 2 * \frac{precision \times recall}{precision + recall} = 2 * \frac{TruePostive}{2 * TruePositive + FalsePositive + FasleNegative}$
5. *Time* : $computationalspeedoftheprogramtofitthedatainseconds.$

3 Spotlight Classifier: Naive Bayes Classifier

Naive Bayes classifier is a very popular classifier used in text classification tasks especially as a baseline tool. It is fairly simple conceptually compared to other more sophisticated classifiers however when it comes to performance in text classification it is quite comparable even outperforming some. [**?**](Tackling the Poor Assumptions of Naive Bayes Text Classifiers, Rennie)

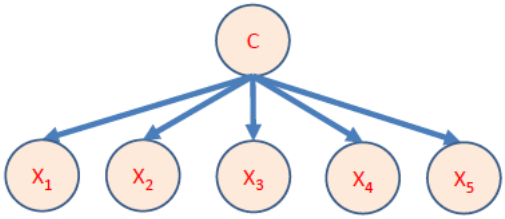


Figure 1: C is the class which has x_1 to x_5 features, while the lines are conditional probabilites. Notice x_1 to x_5 are assumed to be independent(Source:precept slides)

The basis of a naive bayes classifier is the conditional probability, which asks a very simple question: what is the probability that a certain random variable in our case a document, is either a positive or a negative review, given that it contains a certain sequence of words. Given a feature set of $F = \{\boldsymbol{x_1}, \boldsymbol{x_2}...\boldsymbol{x_N}\}$, and a class set of $C = \{\boldsymbol{c_1}, \boldsymbol{c_2}...\boldsymbol{c_N}\}$, we want to find

$$\arg\max_{c} P(C = c|F)$$

Using Bayes theorem, we know that,

$$P(C|x) = \frac{P(C)P(x|C)}{P(x)}$$

In our case the feature set contains more than one element. Hence we need to extend Bayes theorem so that C is conditioned on all the elements of the feature set and their values. Using the chain rule of conditional probability we then can further expand the equation. However, that expanded equation can become very tractable by naively assuming that all the elements in the feature set are independent of each other. Hence the previous equation can be approximated to,

$$\alpha\ P(C|x) = \frac{P(C)\prod_{i=a}^{b} P(x_i|C)}{P(x)}$$

However, that expanded equation can become very tractable by naively assuming that all the elements in the feature set are independent of each other. Hence the previous equation can be approximated to, Using the chain rule of conditional probability we then can further expand the equation.

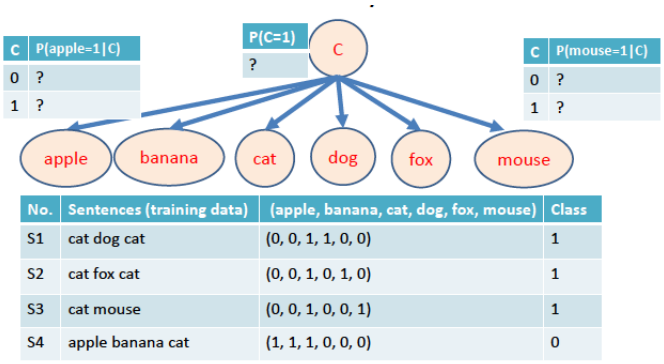


Figure 2: By using Naive Bayes equation as approximated above we can come up with a model that gives us the highest probabilty of a certain feature set belonging either to Class 1 or 0 (Source: precept slides)

4 Results

4.1 Evaluation results

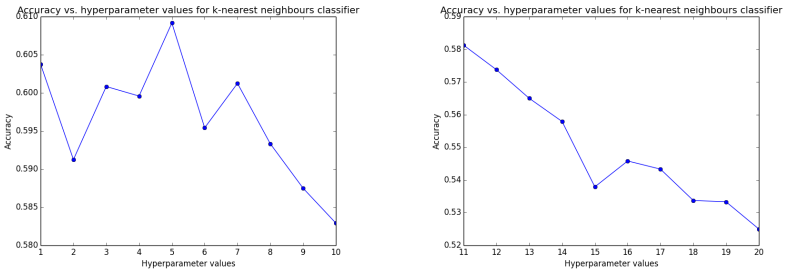
From Table 1 we can see that after performing a 10-fold cross validation training/testing on the whole dataset, the least sophisticated and fastest classifers namely multinomial naive bayes and bernoulli naive bayes classifiers perform better in accuracy, precision and computing time. This is to be somewhat expected as the problem that was at hand especially with the feature extraction and selection techniques I used, did not require a highly complex classifier. Even though it was not tested for this task, if feature selection was performed using support vector machine with linear kernel and l_1 penalty, the more complicated classifiers would begin to outperform the naive bayes classifiers.

Classifier	Accu	Prec	Recall	F_1	Time (s)
MNB	0.693	0.690	0.699	0.694	0.431
BNB	0.694	0.696	0.689	0.693	0.557
SVM	0.686	0.676	0.713	0.694	14.431
DT	0.677	0.671	0.696	0.6832	14.409
RT	0.679	0.667	0.713	0.689	14.425
KNN	0.650	0.699	0.526	0.600	21.320

Table 1: **Results from six classifiers on 3000 reviews dataset.** For each classifier, we report precision, recall, F_1-scores, and wall clock time in seconds for the one-versus-rest classification task with 10-fold cross validation.

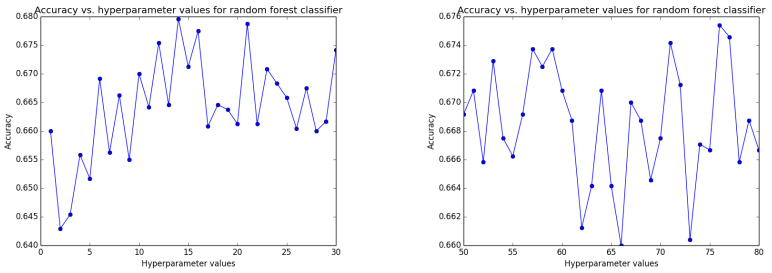
I used 10-fold cross validation to tune hyperparameters for the k-nearest neighbours classifier. It is clear from the Figure 3a and Figure 3b that as the hyperparameter value increased, the accuracy of the classifier generally decreases. From the figures, I was able to determine tht 5 is the optimal hyperparamater for the classifier together with the given data set. It should be noted that I tuned the hyperparameter without including the test data set. I included the test data set only after the hyperparameter had been tuned.

Moreover, as can be seen from Figure 4a and Figure 4b, there is a general but very slight decrease of accuracy as you increase the hyperparameter for a random forest classifier. Moreover, with an increase of the hyperparameter there was an increase of computational time taken.As can be seen from the figures, it seems like 14 is the optimal hyperparameter value for the problem we are facing. I used 14 to evalute the performance of the random forest classifier.



(a) hyperparamter values between 1-10 (b) hyperparamter values between 11-20

Figure 3: Accuracy vs. hyperparameter values for k-nearest neighbours classifier



(a) hyperparamter values between 0-30 (b) hyperparamter values between 50-80

Figure 4: Accuracy vs. hyperparameter values for random forest classifier

4.2 Computational speed

As can be seen from except for multinomial naive bayes and bernoulli naive bayes, all the rest of the classifiers take a significant amount of time to train. The time given is the amount of time taken to train a 3000 document data set using 10-fold cross validation technique. The fastest classifiers are as expected Multinomial Naive Bayes and Bernoulli Naive Bayes. They are the fastest amongst the classifiers because the number of parameters required is linear to the number of features in a learning problem. "Moreover, Maximum-likelihood training can be done by evaluating a closed-form expression,[1]which takes linear time."(Wikipedia) [**?**]

5 Discussion and Conclusion

In this work, I compared 6 classifiers to predict if a certain review had a positive or a negative sentiment. I found that all six classifiers performed relatively closely in terms of precision and recall values but the time taken to train and test data was significantly lower for the naive bayes classifiers.

If we were to perform more rigorous feature selection, the more complicated classifiers would perform better in precision and recall but also on time of computation. Moreover, future work can be done by focusing on the feature extraction and feature selection process, further tuning the response of each classifier so as to get the best result.

References

[1] Pang B, Lee L. Opinion Mining and Sentiment Analysis. Journal Foundations and Trends in Information Retrieval. 2008;2:1–135.

[2] Tang H, Tan S, Cheng X. A survey on sentiment detection of reviews. Journal Expert Systems with Applications: An International Journal. 2009;36:10760–10773.

[3] Tang H, Tan S, Cheng X. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. InLREc. 2010;10(2010).

[4] Zhang Z, Ye Q, Zhang Z, Li Y. Sentiment classification of Internet restaurant reviews written in Cantones. Journal Expert Systems with Applications: An International Journal. 2011;38:7674–7682.

[5] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825–2830.