Sentiment Analysis of Movie Reviews using Machine Learning Techniques.

CS – 584 Machine Learning Fall 2014 Final Project

Aditya Kundu
akundu3@hawk.iit.edu

Alaa Ayach
aayach@hawk.iit.edu

Nov. 19, 2014

1. INTRODUCTION

Sentiment analysis is becoming one of the most profound research areas for prediction and classification. Automated sentiment analysis of text is used in fields where products and services are reviewed by customers and critics. Thus, sentiment analysis becomes important for businesses to draw a general opinion about their products and services. Our analysis helps concerned organizations to find opinions of people about movies from their reviews, if it is positive or negative. One can in turn formulate a public opinion about a movie.

Our goal is to calculate the polarity of sentences that we extract from the text of reviews. We will model sentiment from movie reviews and try to find out how this sentiment matches with the success for these movies. In other words, if a movie review is positive, negative or neutral. But this task can be difficult and tricky. Consider a sentence “the movie interstellar was visually a treat but the story line was terrible”. Now one can clearly see how categorizing this sentence as negative, positive or neutral can be difficult. The phrases “visually a treat” and “story line was terrible” can be considered positive and negative respectively but the degree of their ‘positiveness’ and ‘negativeness’ is somewhat ambiguous. We use a score for common positive and negative words and use this score to calculate the overall sentiment of a sentence.

2. PROBLEM STATEMENT

Text can be categorized in two types based on its properties in terms of text mining: ‘subjectivity’ and ‘polarity’. The focus of our project is to find the polarity of the text which means that we are interested in finding if the sentence is positive or negative. We use machine learning techniques classify such sentences and try to find answers to the following questions:

1. What machine learning techniques are useful for this purpose? Which one out of them performs the best and which techniques are better than the others?
2. What are some of the advantages and disadvantages of traditional machine learning techniques for sentiment analysis?
3. How difficult the task of extracting sentiment from short comments or sentences can be as compared to the traditional topic based text classification?

3. DATA

We will use the Internet Movies Database (IMBD) movie review dataset. This data consists of unprocessed, unlabeled html files from the IMDb archive of the `rec.arts.movies.reviewsnewsgroup`, `http://reviews.imdb.com/Reviews`. In The dataset we have 2000 processed down-cased text files. These files are divided in two categories with respect to their classification as "pos" and "neg", indicating the true classification (sentiment) of the component files. Each line in each text file corresponds to a single sentence, as determined by “Adwait Ratnaparkhi's” sentence boundary detector MXTERMINATOR. This is the second version of the original data set and has 2000 reviews. We may use the first version of the dataset which has 10,000 reviews.

Other Data

We used the AFINN data which is a list of English words rated for valence with an integer between minus five (negative) and plus five (positive). The words have been manually labeled by Finn Årup Nielsen in 2009-2011. There are two versions:

- AFINN-111: Newest version with 2477 words and phrases.
- AFINN-96: 1468 unique words and phrases on 1480 lines. There are 1480 lines, as some words are listed twice.

URL of this data is here:
http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/6010/zip/imm6010.zip

4. IMPLEMENTATION & METHODOLOGY

We went through stages of preparing and cleaning the data to get it ready for the classification algorithms. There are some basic transformations that are required in order to process text data. For our implementation it also includes calculating sentiment scores of words and sentences. Preprocessing is an important step for preparing the data to avoid errors while classification. Even though our original data was preprocessed to remove noise and set the class labels the following steps were performed.

1. Text Preprocessing

This stage includes getting the actual text for all the data we have and trying to separate the individual reviews by considering each review is a single line of the file. As a result, this method will turn into just splitting the content of the file by the end of the line character.
Other part of this stage is to convert the resulted reviews into lower case, so in that case we can get matches with the AFINN data that we will work on in the next stage.

Also to avoid mismatch cases we omitted punctuations, numbers and control characters to get better matches.

2. Categorizing words

Although in the AFINN data we have a ten degrees of labeling the words' positivity and nega, we assigned scores but in four categories:

A. Very Positive (from 4 to 5 stars)

B. Positive (from 1 to 3 stars)

C. Very Negative (from -5 to -4 stars)

D. Negative (from -1 to -3 stars)

3. Sentence Scoring

We saw in the previous stage how we could score words in reviews but what about sentences?

In this case we accumulate the score of the sentence as by adding the scores for words composing the sentence. In that way, if the sentence has more negative and very negative words the final sentence score will be negative and vice versa if the sentence has more positive and very positive words.

At last our final corpus will consist of both positive-like and negative-like reviews added in one dataset.

4. Cross Validation

We used cross validation with the parameter "interleaved" to test all the data we have for different training parts. we used the function ***cvFolds()*** provided by the package **cvTools.** We did a 5-fold cross validation and in some cases we also used a 10 fold cross validation. Cross validation is a good method to evaluate the performance of a model as it divides the data into two parts. One part is used to train the model and build the classifier and the second part is used to test the accuracy of the predictions of the model. This process is done iteratively and different subsets of the data are used to are used for training and testing at each iteration.

5. Modeling and evaluation

We used three main models for our baseline of accuracy of predictions. These algorithms were implemented in R using Rstudio 3.1.1.

1. Naïve Bayes
2. Support Vector Machines
3. Random Forests

We then make feature selection using unigrams, bigrams and trigrams of the data and using these n-grams as features to train a model we built a logistic regression classifier to test if n-grams help for a better classification. The feature selection and logistic regression was done in python.

5. RESULTS & EVALUATION

1. Naïve Bayes

Naive Bayes works on the principle of probabilities and the Bayes rule given by:

$$\boldsymbol{P(c|d) = \frac{P(c)P(d|c)}{P(d)}}$$

Where P(c|d) is the probability of a given document (text) belongs to class c, which is the classification part which we are interested in. Below is the confusion matrix for the naïve bayes classifier in our project. We see that the classifier has misclassified many examples and the accuracy obtained of 61.996% is not that great.

Confusion Matrix

	Actual		
Predicted		**Positive**	**Negative**
	Positive	2795	1516
	Negative	2536	3815

2. Support Vector Machines

This algorithm tries to find a margin between the data points for a given dataset that are divided in two set of classes that separates them. The thickness of the margin is determined by the distance between the support vectors which are data points that are closest to the separating function and lie exactly on the borders of the margin. Using this margin and the support vectors we the classify examples.

Below is the confusion matrix of the performance of the support vector machine. We can see that this classifier has misclassified more number of data points as compared to naïve bayes. The accuracy of this model comes out to be 59.308% which is lower than that for naïve bayes.

Confusion Matrix

	Actual		
Predicted		**Positive**	**Negative**
	Positive	33431	24499
	Negative	14548	23480

3. Random Forest

Random forests implement decision tree algorithms to learn a number of decision trees to make a classification decision. At every step of the learning process an attribute is selected to split to two or more different parts and this process is iteratively repeated until the attributes are exhausted or we have reached a pure classification split. A pure classification split is when the split parts represent only one class that they belong to. At every split we try to reach a local optimum solution.

Below is the confusion matrix for the random forest and the performance is better than that of support vector machines but not better than naïve bayes. The accuracy obtained for this classification algorithm is 61.781%.

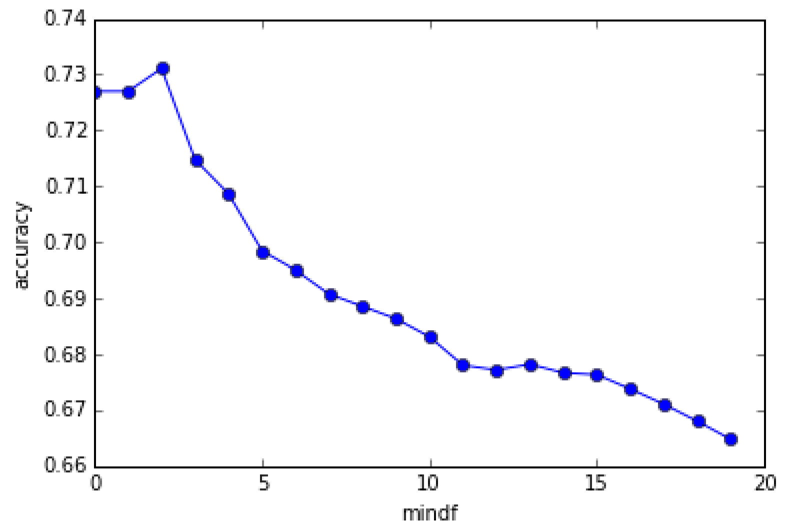
Confusion Matrix

	Actual		
Predicted		**Positive**	**Negative**
	Positive	27269	15964
	Negative	20710	32015

4. Feature Selection, N-gram and Tuning

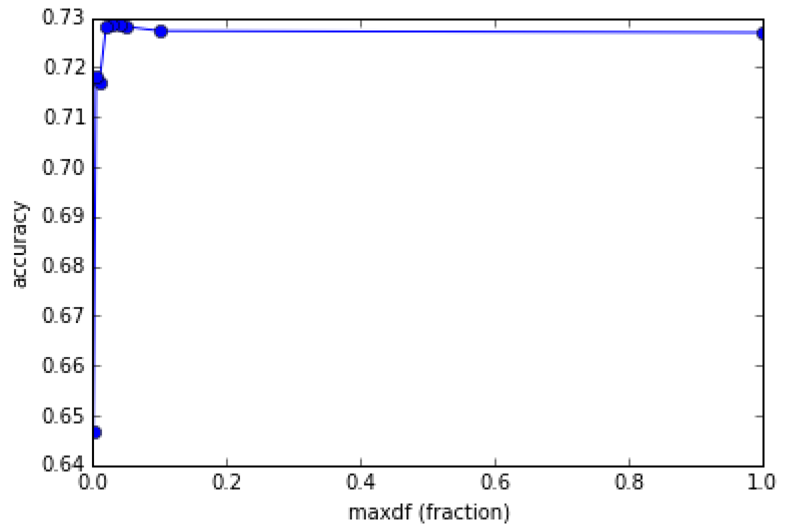
Feature selection is an important aspect and useful method for improving accuracies and decreasing the computation time and the memory usage of an algorithm. We developed n-grams for our texts and then tested the logistic regression algorithm by selecting them as features for training of the model.
We start by building a document term matrix for the given data which indicates the sentences as rows and the corpora of words in the entire text as the columns. The values in the matrix are the frequencies of the words appearing in a sentence. The matrix we obtained was highly sparse.

1. Reducing the number of terms by limiting term frequency as min



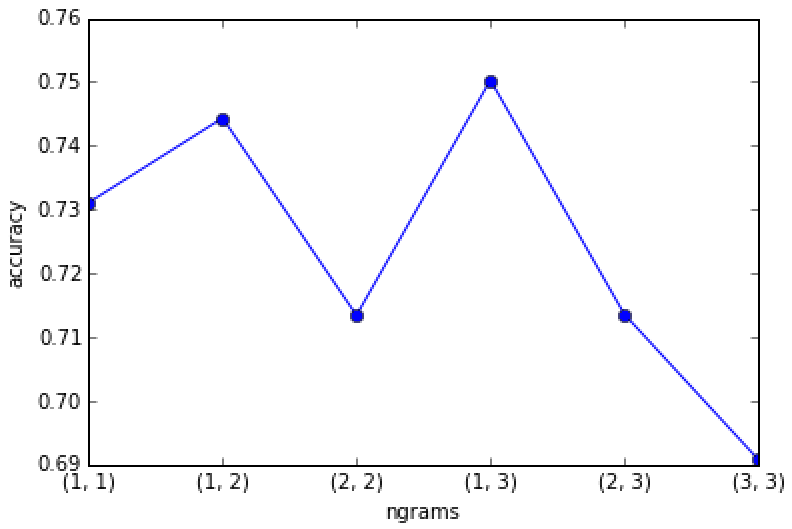
We saw that this wasn't very helpful except for some values for mindf.

2. Reducing the number of terms by limiting term frequency as max



Also a little increase of the accuracy was accomplished by changing the maxdf

3. N-grams



We see that when using (1, 3) (i.e. one-term to 3 terms) n-grams we got the best accuracy

6. CONCLUSION & FUTURE WORK

METHOD	ACCURACY (%)
Naïve Bayes	61.996
Support Vector Machine	59.308
Random Forests	61.781
One-three n grams with logistic regression	75.333
Average on n-grams	73.561

We learned that the traditional machine learning classification algorithms do not work very well with sentiment analysis of text as compared to their performance with topic based classification. We also learned that out of the three algorithms we used for the baseline Naïve bayes performed the best by giving 61.996% accuracy. However the proposed solution for feature selection by using n-grams and then classifying by logistic regression performed much better than the traditional approaches. Our goal of learning how difficult and challenging the task of sentiment analysis using traditional approaches are, was met and all of them performed poorly in classification as compared to n-grams after feature selection.

7. BIBLIOGRAPHY

1. Bo Pang Lee, Lillian Lee and Shivakumar Vaithyanathan. "Thumbs up? Sentiment Classification using Machine Learning Techniques," in *Proc. EMNLP, Philadelphia*, 2002, pp. 79-86.
2. Bo Pang and Lillian Lee. "Seeing Stars: Exploiting Class Relationships For Sentiment Categorization With Respect To Rating Scales" in *Proc. ACL*, 2005, pp. 115-124.
3. B. Pang and L. Lee. "Opinion Mining and Sentiment Analysis". *Now Publishers Inc*, July 2008.
4. Informatics and Mathematical Modelling, Technical University of Denmark. Internet: http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010
5. Finn Årup Nielsen. " A New: Evaluation of a word list for sentiment analysis in microblogs", *Proc. ESWC Workshop on 'Making Sense of Microposts'*, 2011 May, 93-98.
6. Bo Pang and Lillian Lee. "Movie Review Data". Internet: http://www.cs.cornell.edu/people/pabo/movie-review-data/ .