LING 406 Term Project
Sentiment Classifier

Keshav Malhotra
kmalhtr3@illinois.edu

Introduction

Sentiment analysis is used in a variety of different platforms on the web today, for instance, in movie reviews, social media platforms, product reviews on e-commerce websites like amazon, bestbuy etc. Characterizing a given piece of text as either positive, negative or neutral is usually the main the goal of this technique. This is very useful for a company's data analytics for recognizing the experiences of customers using the product/service, in order to get a clearer view of the consumer's emotions associated with the business and help point out the areas that need improvement.

With the advent of big social media platforms like Facebook and Instagram, sentiment analysis has gained a lot of interest and has become a very powerful tool to monitor sentiments of a big user base and visualize it using plots. Facebook has over 2.2 billion monthly active users and therefore it has access to petabytes of data which is constantly logged for targeted advertisements and content improvement. This helps attract more customers to the platform and fill their feeds with posts they want to see and ads they want to click on. Therefore, in this case, sentiment analysis can be a very powerful resource to learn more about the active users and what they are interested in.

Problem Definition

Sentiment Analysis is the process of identifying and categorizing opinions expressed in a piece of text. The input can be a review on a product, movie, tv show, or even comments/conversations on social media posts. The output is positive, negative or neutral, but can branch out to be more specific, for instance, involving different degrees of emotions such as anger, contentment, happiness etc.
The task at hand can be tricky as understanding natural language requires machines to have a broader understanding of the world around them and being able to devise classifiers requires modelling complex linguistic structures which can be used as features to improve the classifier's accuracy. It also requires training the classifier on a large set of accurate corpora(human tagged) from diverse sources, which can then be used for testing and validation on more data and devised in real life applications for further improvement.

Previous Work

Sentiment analysis has been studied extensively in the past. In paper (1) (check Reference section at the end), the authors mention the large amount of content generated by users on social media and use Machine Learning and Semantic Sentiment analysis based Algorithms for Suicide Sentiment prediction. They use data from Twitter APIs to gather a large amount of tweets and manually generate tweets that can be linked to suicidal behaviour and build several machine learning classifiers such as Support Vector Machine, Maximum Entropy and Naive Bayes (using Weka), which are then used on testing data for validation and comparison of their accuracies. In their results they observe that around 66% percent of the tweets in their datasets are suspected tweets with a some degree of risk involved. Models which performed the best on their sets of testing data include Sequential Minimal Optimization, Naive Bayes and Decision Tree classifiers.

Paper (2) also uses the same 3 broad machine learning techniques mentioned in the paragraph above to classify IMDb(International Movie Database) reviews as Positive, Negative or Neutral using different features such as unigrams, bigrams, Part of speech tagging etc. They compare the accuracies of the different classifiers and come to conclusion that Naive bayes tends to do worst and SVMs(for unigrams) tend to do best although the differences aren't very large.

A study at stanford (3) discusses deep learning for sentiment analysis of movie reviews. They use a dataset containing 50,000 labelled(positive or negative) and 50,000 unlabelled movie reviews from IMDb and classify them using Random forest, Logistic regression and SVM techniques. The highest accuracy is yielded by logistic regression(86.6%) followed by SVM(85.8%) and then Random Forest(84.0%).

Approach

Baseline Method : Bag of Words model

In this method, I acquired 2 large lists, one containing commonly used positive words and the other containing commonly used negative words from resource (4). The model simply iterates over all words in a given review and keeps counters for positive and negative words appearing in the review and assigns either positive or negative output to that review based on whichever value is greater. Since there was no training data used in this model, I used the entire dataset containing 2000 reviews for testing and calculated the accuracy of the model to be 69.85%.

Data Preprocessing

I obtained the polarity dataset from NLTK and converted it into a list of 2000 tuples, each tuple containing a list of words from a review as its first element and it's label('pos' or 'neg') as it's second element.
Example tuple : (["The","movie","was","good"],'pos ')

Sample Size/Features

I used a Frequency distribution to pick out the most commonly used words from the polarity dataset. I started with 100 most frequent words as the feature set and moved up to test different number of features until I saw no significant improvement in the accuracies. The runtimes for training the classifiers was getting really large beyond 5000, especially for decision tree, so I chose to use that as my upper limit so I can run my model several times with different combinations of features discussed in the next section

Number of Features	Naive Bayes	Decision Tree	SVM
100	75.25%	62%	73.5%
500	75.25%	**66.75%**	74.75%
1000	71%	56%	68.5%
5000	**78.75%**	66.5%	**76.25%**

Table 1. Contains accuracy for different models for different feature sizes. The accuracy in bold represents the highest for a given classifier(in a column)

Feature Addition

The first step in my approach after I gather the data was to get rid of stopwords(words that do not contain important significance in search queries eg a, an, the, also, out, part etc).

I used three different additional features which were tested using all the classifiers.

1. **Marking Negation**: This was done using the NLTK SentimentAnalyzer, which basically appends "_NEG" to words that follow a negation such as don't, not etc. This helps improve the case for unigram models individual words and can judge the review, "I don't like this movie" to be positive since it contains a positive word, "like". Since I am only using unigram models for my machine learning classifiers, I picked this feature to improve my baseline model.

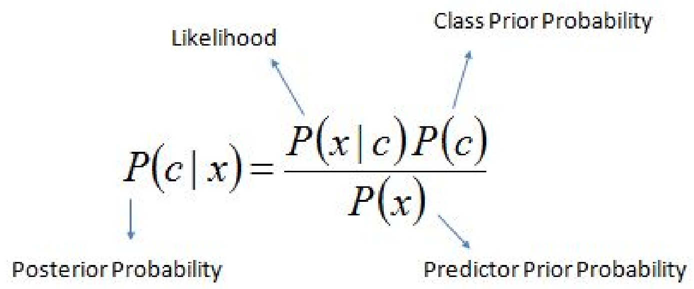
2. **Using Adjectives and Adverbs only**: This was done using NLTK Part of Speech tagging method and it’s useful because adjectives and adverbs give a good measure of the subjectivity of the person writing the piece of text and his sentiments towards the movie are reflected clearly in them.

3. **Stemming**: Stemming is the process in which inflected words such as argue, arguing, argued are all reduced to their word stem - argue. This helps reduce vocabulary size, thereby intended to sharpen the results obtained by the different ML classifiers.

Machine Learning Classifiers

I used three different Machine Learning algorithms, discussed below, to build classifier and then test them. My training data comprised of 80% of the reviews(obtained after shuffling the dataset), and the rest 20% was used as testing data.

1. **Naive Bayes:** This classifier is based on a simple probabilistic model derived from Bayes Theorem with an assumption of independence of among predictors. In other words, Naive Bayes classifier assumes that presence of a particular feature in a class is unrelated to the presence of any other feature. Despite the fact that conditional independence of words does not exist in real world situations, Naive Bayes still performed quite well(See table 2 in Results).



$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Figure 1. Depicts the probabilistic model behind the Naive bayes classifier

2. **Decision Tree:** A decision tree is a decision support tool that uses a tree like graph or model of decisions and their possible consequences. It is a way to display an algorithm that contains conditional control statements. It’s important that the data being used contains several key tokens(in our case, the review should contains key positive or negative words for making the correct decision for classification).

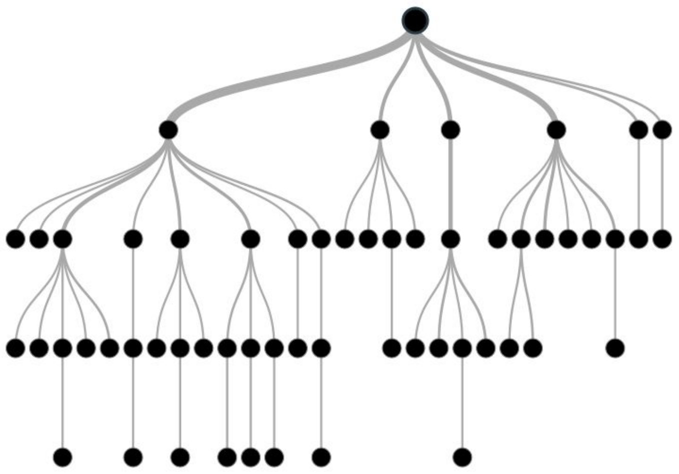
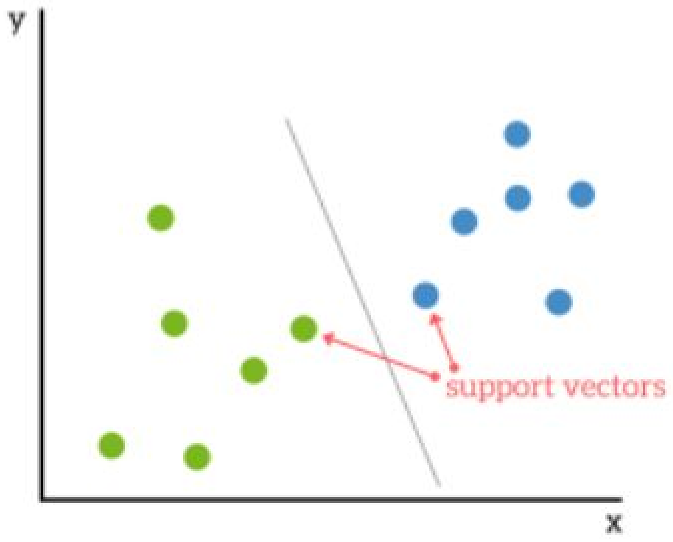


Figure 2. Shows workflow of a typical decision tree

3. **Support Vector Machine:** This is another supervised machine learning algorithm that is employed for both classification and regression purposes. The basic idea behind an SVM is finding a hyperplane that best divides a data into 2 classes as shown below. It is shown to be a highly effective technique for text classification and therefore I picked this as my third choice.



Results

Features	Naive Bayes	Decision Tree	SVM
Negation	67.5%	55.5%	66.0%
Adjective/Adverbs	74.5%	64.75%	71.25%
Stemming	72.75%	62.2%	**79.65%**
Pure BoW	**78.75%**	**66.5%**	76.25%

Table 2: contains the accuracies for all the classifiers after applying **individual** features. The accuracy in bold represents the highest one for a particular column

Features	Naive Bayes	Decision Tree	SVM
Neg + Stem	66%	54.5%	68.0%
Stem + Adjv	**76.5%**	**66.5%**	**72.5%**
Adj + Neg	68.75%	56.5%	67.75%
Stem + Adjv + Neg	68.75	59.5	68.5

Table 3: contains the accuracies for all the classifiers after applying combination of features. The accuracy in bold represents the highest one for a particular column

Discussions and Conclusions

After trying several machine learning models, I found the accuracy of the SVM model with stemming to be the highest at 79.65%, followed by Naive Bayes without any features, at 78.75%. Most models with or without the addition of features showed improvement from the baseline accuracy of 69.85%, except the Decision Tree model which showed the worst results that declined with addition of features. The reason for this is that decision tree needs several key tokens for text classification and random forest works bad for high sparse dimensions.The combination of different features also didn't improve the accuracy of classifiers from their pure BoW performance. Naive Bayes's simplicity resulted in a high average accuracy and I learned that addition of features for this classifier didn't improve accuracy which makes sense since the model runs on conditional independence between words. In the future, I'd look out for other features and try bigram and trigram models, since they probably contain more useful meaning for text entries. I would also like to try a different classifier model such as logistic regression.

References

(1) Marouane Birjali, Abderrahim Beni-Hssane, Mohammed Erritali, Machine Learning and Semantic Sentiment Analysis based Algorithms for Suicide Sentiment Prediction in Social Networks, Procedia Computer Science,Volume 113,2017,Pages 65-72, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2017.08.290.

(2) Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. Association for Computational Linguistics, 2002.

(3) H. Pouransari, S. Ghili, "Deep Learning for Sentiment Analysis of Movie Reviews", 2014.

(4) https://github.com/jeffreybreen/twitter-sentiment-analysis-tutorial-201107.git