Predicting sentiments in movie reviews

Milind Solanki,* Priyanka Harlalka,* Rishika Garg,* and Sumedh Masulkar*

E-mail: milind@; priyah@; rishika@; sumedh@

Abstract

Sentiment analysis is a field dedicated to extracting subjective emotions and feelings from text. One common use of sentiment analysis is to figure out if a text expresses negative or positive feelings. Written reviews are great datasets for doing sentiment analysis, because they often come with a score that can be used to train an algorithm. We hereby attempt to use various machine learning techniques in order to understand and then predict sentiments in movie reviews.

Introduction

Googles Word2Vec is a deep-learning inspired method that focuses on the meaning of words. Word2Vec attempts to understand meaning and semantic relationships among words. The Movie Reviews are categorized in two types: Positive and Negative. The methods used by us are as follows:

1. We use Bag of Words
2. We use Word2Vec and employ Random Forests, SVM and Bag of Centroids
3. We also use TfIdf values and apply Logistic Regression

*To whom correspondence should be addressed

4. We perform a Naive Bayes Analysis

5. We use Latent Semantic Analysis

Bag of Words

First of all, all the reviews are cleaned. We remove stop words, non-letters, html codes. Creating feature vectors is equivalent to providing a numeric representation to the words in the cleaned movie review. We create feature vectors using Count Vectorizer and fit_transform.

We first learn the vocabulary, then create vectors of the length of the vocabulary(in case the size of the vocabulary is not that large), or else we can take the most frequently used words. We create feature vector for each review, and have each word as a feature whose value depicts its frequency.

After creating all these vectors we create a random forest, creating a bag of words from the training set. This performs a little better than Word2Vec and produces an accuracy of 0.84568.

Word2Vec

Word2Vec is a neural network implementation that learns distributed representations for words. Unlabeled data are also used to create meaningful representations in Word2Vec.

Word2Vec model is trained using list of parsed sentences. This Word2Vec model consists of a feature vector for each word in the vocabulary, stored in a numpy array. The dimension of this array is (number of words in the model's vocabulary * the size of the feature vector). For a given review we find its feature vector in the model, if it is present if the vocabulary set. And then to find the feature vector for a review, we take an average of all the word vectors.

We used average review vectors to train random forest, with different number of trees and then calculated the accuracy which came out to be 0.83632 for 350 trees. We then

performed SVM using the same averaged word vectors to find an accuracy of 0.74156. Given such feature vectors, using Random Forests and SVMs are two most intuitive and commonly used techniques.

The third method that we use is called as Bag of Centroids. Word2Vec creates clusters of semantically related words. Words with similar meanings appear in clusters, and clusters are spaced such that some word relationships, such as analogies, can be reproduced using vector math. Hence, we exploited this feature. Each word of vocabulary is assigned a cluster. Then for each review, we created a numpy array of dimension (1 * number of centroids). Then, for each word in review, we found its centroid, say i and incremented the i^{th} element of the array. That is how, we converted reviews into bags-of-centroids and then using these vectors we trained random forest and using same vectors for test review we extracted results.

TfIdf and Logistic Regression

Tf Idf : Term frequency Inverse document frequency
TfIdf Vectorizer converts a collection of raw documents to a matrix of Tf Idf features. It is equivalent to a CountVectorizer followed by TfIdf Transformer.

TfIdf is a numerical statistic that is intended to reflect the importance of one word in a document. It is used as a weight in information/sentiment retrieval. Tf-Idf is the product of two statistics, term frequency and inverse document frequency. We create Tf-Idf feature vectors for the training set. We can then perform the same random forest and svm techniques but then the results would not be very different. We here apply a new technique called as Logistic Regression.

Logistic Regression is used when in the regression model the response variable is categorical. The variance of the response variable is a logit function. The estimated probabilities are used to predict the category. F(x) is interpreted as the probability of the dependent variable equalling a success. Uptil now, We did not take into consideration the relationship between different words in a sentence or paragraph. Logistic Regression takes the mutual relationship

into consideration and hence this produces the maximum accuracy of about 0.95294.

$$Var(t) = \sigma(t) = e^t/(e^t + 1)$$

$$F(x) = 1/(e^{-t} + 1)$$

where, $t = \beta_0 + \beta x$

Naive Bayes

Naive bayes is a popular algorithm for classifying text. Although it is simple, it often performs as well as much more complicated solutions as per the results. It is based on the Bayes Formula.

$P(Y|x_1, x_2, ..., x_n) = \frac{P(Y)P(x_1|Y)P(x_{2|Y})....P(x_n|Y)}{P(x_1)P(x_2)..P(x_n)}$

All the reviews are cleaned, *i.e.* the html tags and markups are removed, all letters are converted to lowercase and stopwords are removed. This is done in order to ensure that the html mark ups and non-important words that appear a lot in texts do not effect our results as the method is based on frequency of words. All positive reviews are joined and converted into a single text and negative reviews are joined into another text. Probability of words ($P(x_i)$) and words given class ($P(x_i|y)$), i.e., the prior probabilities are calculated for all words. The probability of a class $P(y)$ is simply the number of reviews of a given class divided by total number of reviews.

Then the reviews to be tested are cleaned the same way as described above. Posterior probability of class Y given words occurring in the review ($x_1, x_2..x_n$) is calculated for both the classes(positive and negative). The review is classified into the class(positive or negative) for which the probability of review is higher.

This method produces an accuracy of 0.76644 which is not as good as Logistic Regression because here in the formula we assume all words to be independent of each other, which does not happen perfectly in real reviews. Also, this method does not go into semantics and

context of the words/sentences which leads to wrong results in sarcasm and negation etc.

Latent Semantic Analysis

LSA analyses the relationships between a set of documents and the terms they contain. It assumes that words that are close in meaning will occur in similar pieces of text. We construct a matrix with words in rows and review in columns, construct a SVD, to reduce number of rows. Words are then compared by taking the cosine of the angle between the two vectors (or the dot product between the normalizations of the two vectors) formed by any two rows. Values close to 1 represent very similar words while values close to 0 represent very dissimilar words.

LSA model is made using models LsiModel(), then we indexed similar reviews in the LSA model using similarities. MatrixSimilarity(). For each incoming review we found the most similar review in the LSA model using index. We then checked for the K-nearest neighbours, in the previous matrix and then classify the review accordingly. This method produces an accuracy of 0.72423 which makes it fall in the same category as Naive Bayes and SVM. In Naive Bayes the independence of feature vectors/words is assumed and here also we take the similarity between words to be the judging criterion. It can so happen that the reviews contain similar kind of words but convey a very different meaning. The integration of same word differently can produce very different meanings. Hence the classification may not be that accurate.

Summary

Accuracy

Table 1: Accuracies

Method	**Accuracy**
Bag of Words	0.84568
Word2Vec - Random Forest	0.83632
Word2Vec - SVM	0.74156
Word2Vec - BagOfCentroids	0.84516
TfIdf-logistic regression	0.95294
Naive Bayes	0.76644
Latent Semantic Analysis(LSA)	0.72423

1. Word2Vec gives less accuracy than BagOfWords because the element-wise average of the vectors doesn't produce spectacular results.

2. BagOfCentroids gives better accuracy than Random forest and SVM as it exploits the fact that Word2Vec creates clusters of semantically related words.

3. Logistic Regression takes the mutual relationship of different words into consideration and hence this produces the maximum accuracy.

4. Naive Bayes is not as good as Logistic Regression because in Naive Bayes we assume all words to be independent of each other.

5. Latent Semantic Analysis(LSA) does not perform as well as other methods as this method is used to cluster similar documents which is based on frequency of words and similarities of words used. But reviews of same class need not be similar to others of same class.

Plots

1. Random Forest

(a) The graph shows error rate with number of trees in Random forest of Word2Vec-Random Forest method.

(b) It is observed that here error level off is at around 350 number of trees.

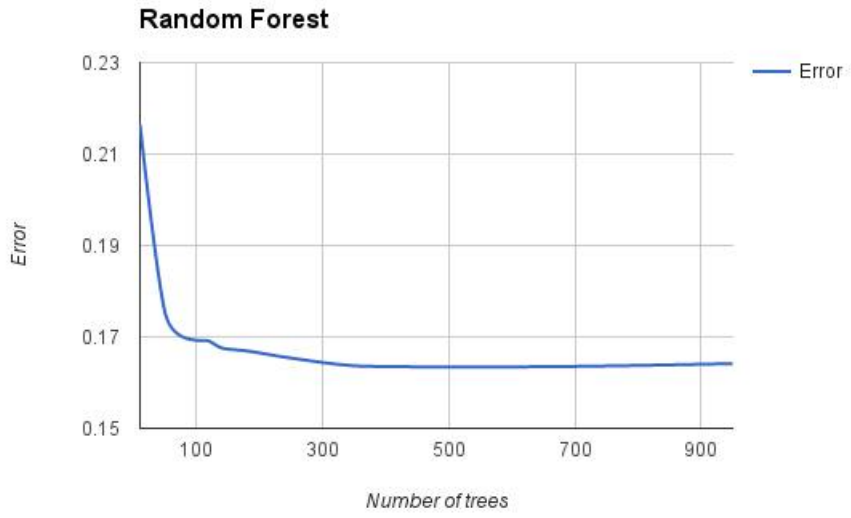

Figure 1: Graph for Random Forest

2. Naive Bayes

(a) The graph shows error rate with size of training data in Naive Bayes method.

(b) It is observed that error decreases with increase in size of training data. This should be because with more data, the classifier gets tuned, i.e. the probabilities improve.

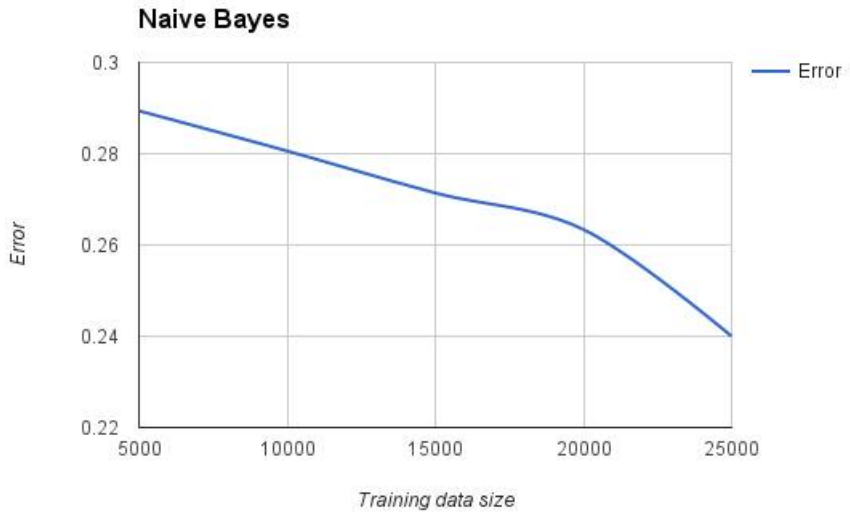

Figure 2: Graph for Naive Bayes