M S RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous Institute Affiliated to VTU)

Department of Information Science and Engineering

A Project Report on

SENTIMENT ANALYSIS ON MOVIE DATABASE

Submitted in partial fulfillment of the CIE for the subject

Natural Language Processing(IS52A4)

by

GOPAN SAMANTARAY (1MS14IS041)

ESHAN GULHATI (1MS14IS036)

Under the guidance

of

Rajeshwari S B

Assistant Professor

Department of ISE, MSRIT

Bengaluru-560054

INTRODUCTION

Movie reviews are an important way to gauge the performance of a movie. While providing a numerical/stars rating to a movie tells us about the success or failure of a movie quantitatively, a collection of movie reviews is what gives us a deeper qualitative insight on different aspects of the movie. A textual movie review tells us about the the strong and weak points of the movie and deeper analysis of a movie review can tell us if the movie in general meets the expectations of the reviewer. Sentiment Analysis is a major subject in machine learning which aims to extract subjective information from the textual reviews. The field of sentiment of analysis is closely tied to natural language processing and text mining. It can be used to determine the attitude of the reviewer with respect to various topics or the overall polarity of review. Using sentiment analysis, we can find the state of mind of the reviewer while providing the review and understand if the person was “happy”, “sad”, “angry” and so on. In this project we aim to use Sentiment Analysis on a set of movie reviews given by reviewers and try to understand what their overall reaction to the movie was, i.e. if they liked the movie or they hated it. We aim to utilize the relationships of the words in the review to predict the overall polarity of the review. Dataset: The dataset used for this task was collected from Large Movie Review Dataset which was used by the AI department of Stanford University for the associated publication . The dataset contains 50,000 training examples collected from IMDb where each review is labelled with the rating of the movie on scale of 1-10. As sentiments are usually bipolar like good/bad or happy/sad or like/dislike, we categorized these ratings as either 1 (like) or 0 (dislike) based on the ratings. If the rating was above 5, we deduced that the person liked the movie otherwise he did not. Initially the dataset was divided into two subsets containing 25,000 examples each for training and testing. We found this division to be sub-optimal as the number of training examples was very small and leading to under-fitting. We then tried to redistribute the examples as 40,000 for training and 10,000 for testing. While this produced better models, it also led to over-fitting on training examples and worse performance on the test set. Finally, we decided to use CrossValidation in which the complete

dataset is divided into multiple folds with different samples for training and validation each time and the final performance statistic of the classifier is averaged over all results. This improved the accuracy of our models across the boards. A typical review text looks like this:

I'm a fan of TV movies in general and this was one of the good ones. The cast performances throughout were pretty solid and there were twists I didn't see coming before each commercial. To me it was kind of like Medium meets CSI.

Did anyone else think that in certain lights, the daughter looked like a young Nicole Kidman? Are they related in any way? I'd definitely watch it agin or rent it if it ever comes to video.

Dedee was great. Haven't seen in her in a lot of things and she did her job very convincingly.

If you're into to TV mystery movies, check this one out if you have a chance.

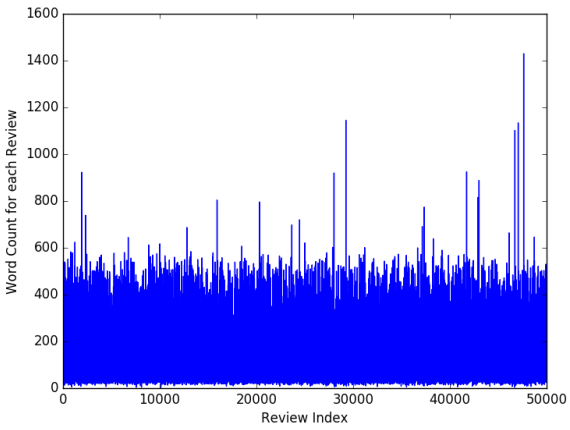
As seen above, one necessary pre-processing step prior to feature extraction was removal of HTML tags like “ ”. We used simple regular expressions matching to remove these HTML tags from the text. Another important step was to make the text case-insensitive as that would help us count the word occurrences across all reviews and prune unimportant words. We also removed all the punctuation marks like ‘!’, ‘?’, etc as they do not provide any substantial information and are used by different people with varying connotations. This was achieved using standard python libraries for text and string manipulation. We also removed stopwords from the text for some of our feature extraction tasks, which is described in greater detail in later sections. One important point to note is that we did not use stemming of words as some information is lost while stemming a word to its root form.

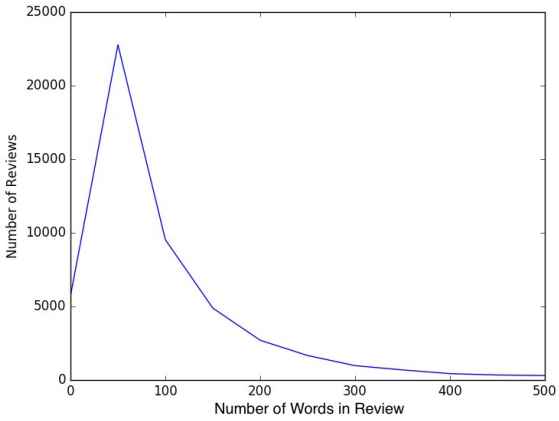
Predictive Task

The main aim of this project is to identify the underlying sentiment of a movie review on the basis of its textual information. In this project, we try to classify whether a person liked the movie or not based on the review they give for the movie. This is particularly useful in cases when the creator of a movie wants to measure its overall performance using reviews that critics and viewers are providing for the movie. The outcome of this project can also be used to create a recommender by providing recommendation of movies to viewers on the basis of their previous reviews. Another application of this project would be to find a group of viewers with similar movie tastes (likes or dislikes). As a part of this project, we aim to study several feature extraction techniques used in text mining e.g. keyword spotting, lexical affinity and statistical methods, and understand their relevance to our problem. In addition to feature extraction, we also look into different classification techniques and explore how well they perform for different kinds of feature representations. We finally draw a conclusion regarding which combination of feature representations and classification techniques are most accurate for the current predictive task.

Exploratory Analysis

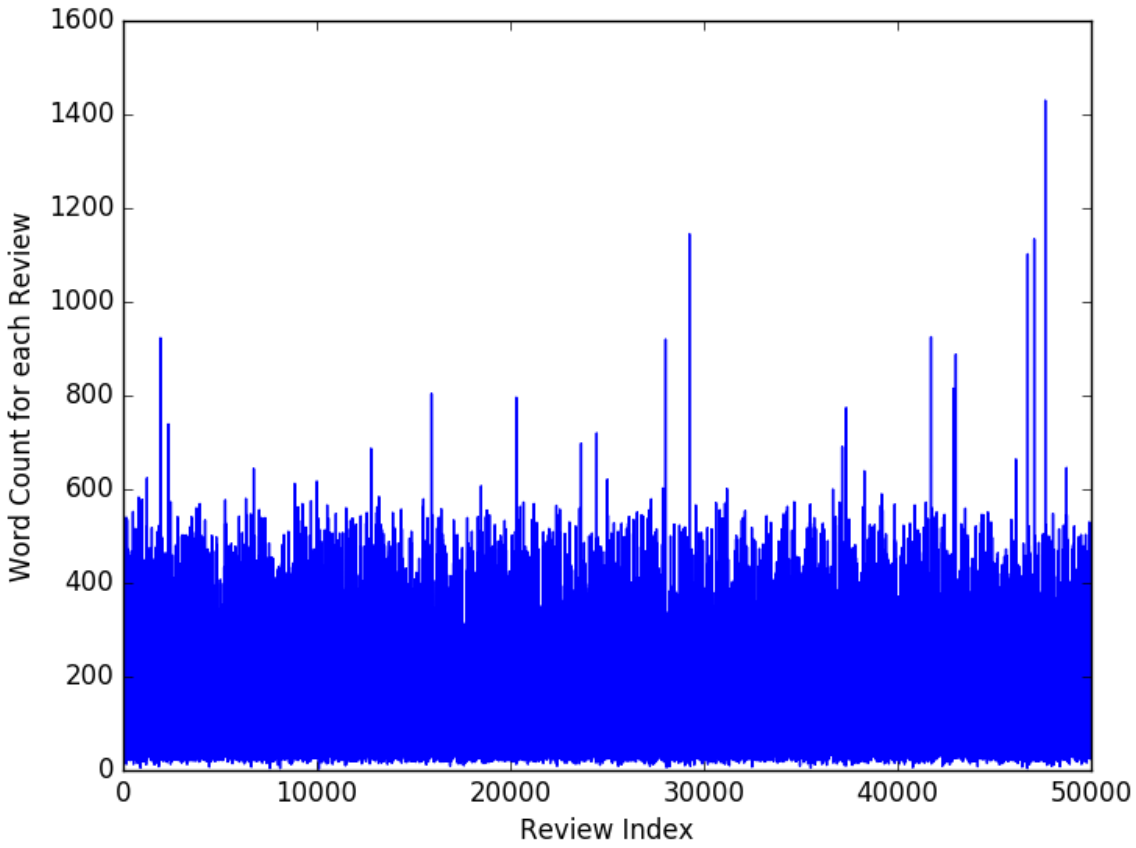
One of the starting points while working with review text is to calculate the average size of reviews to get some insight on quality of reviews. The average number of words per review is around 120. The graphs below clearly indicate the variation of the word count for each review. From this information we deduced that in general people tend to write pretty descriptive reviews for movies and as such this is a good topic for sentiment analysis. Also, people generally write reviews when they have strong opinions about a movie; they either loved it or hated it.





Apart from the word count per review another interesting metric was occurrence count of words across reviews. Some words have higher occurrence counts as compared to others depending on their relative importance. Below is the list of 20 most occurring words in negative and positive reviews along with a graph showing variability of word occurrences across all reviews. Also, the average word occurrence count was around 33 over all 50000 reviews. From all this information and the below graphs, it is clear that “Bag of Words” is not a very good model for doing sentiment analysis of reviews because similar words have high counts in both positive and negative reviews. Also, overall number of unique words is huge (1,63,353) across all the reviews and hence we use only top 50,000 and 1,00,000 of these during training. Also, this realization prompted us to move to other methods of feature extraction like n-gram modelling and TF-IDF counts of each words.

Negative Reviews		**Positive Reviews**	
Movie	Film	Film	Movie
Like	Even	Like	Good
Good	Bad	Great	Story
Would	Really	See	Time
Time	See	Well	Also
Don’t	Get	Really	Would
Much	Story	Even	Much
People	Could	First	Films
Make	Made	Love	People
Movies	First	Best	Get



Feature Extraction

We used 3 methods for extraction of meaningful features from the review text which could be used for training purposes. These features were then used for training several classifiers.

• Bag of Words: This is a typical way for word representation in any text mining process. We first calculated the total word counts for each word across all the reviews and then used this data to create different feature representations. As the total number of words in the dictionary was huge (more than 1,60,000) the first feature set was created using only the 50,000 most frequent words according to their occurrence. Another feature set was created in a similar fashion but using top 1,00,000 words. In addition to this, we created another bag of words representation using all words that occurred at least twice across the whole dataset. This ensured that we remove most of the misspelled words. Also, words which occurred only once in the dataset would contribute nothing to the classifier. Another feature representation was created along the same lines but with words occurring at least 5 times. The size of these 2 features representations was roughly 34,000 and 76,000 respectively.

• N-Gram Modelling: Bag of Words ignores the semantic context of the review and concentrates primarily on frequency of each word. To overcome that, we also tried ngram modelling wherein we created unigrams, bigrams and mixture of both. While creating unigrams is more or less similar to the bag of words approach, bigrams provided more contextual information on the review text. We created one feature representation similar to the “Bag of Words” approach above but using the bigrams. In other representations, we took a mixture of unigrams and bigrams and included only those which were occurring more than once. Also, to get more insight on textual information we created a feature set using a mixture of n-gram with n = 5 and using only those grams with minimum count of 10. In case of n-gram modelling, we did not remove the stopwords as we were doing for previous cases.

• TF-IDF Modelling: While the two methods of feature extraction descried above concentrated more on higher frequency parts of the review they completely ignored the portions which might be less frequent but have more significance for the overall polarity of the review. To account for this, we

created feature representations of words using TFIDF. The feature representation for this model issimilar to the Bag of Words model except that we used TF-IDF values for each word instead of their frequency counts. To limit the number of words common to both positive and negative reviews, we ignored all the words whose count was more than 50 as they would not contribute much to the classifier.

Models

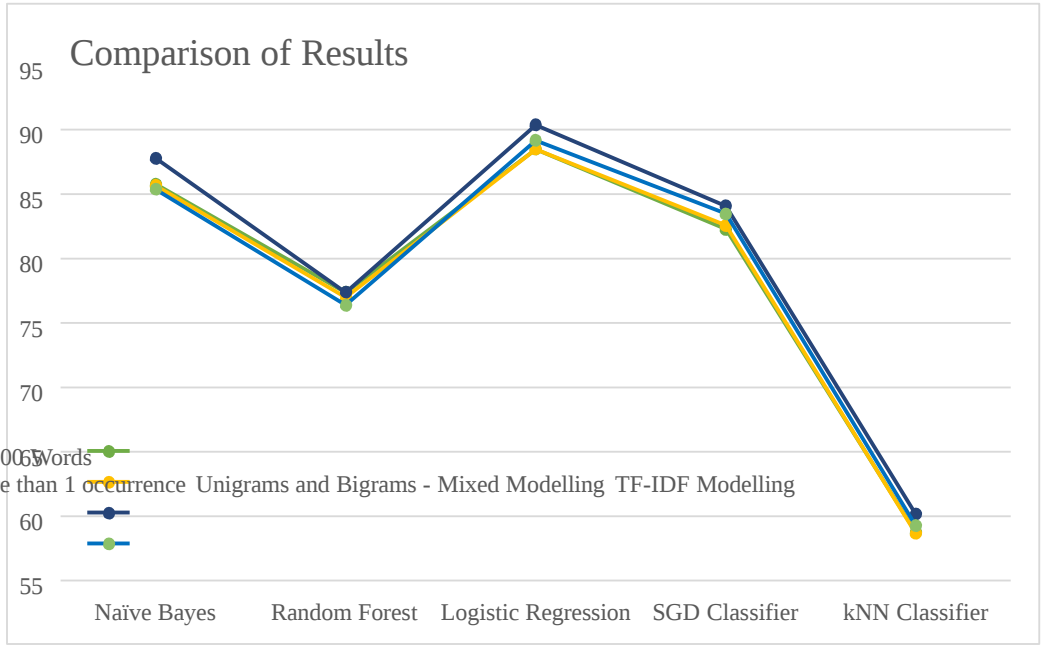
The overall task in this project is for classification of reviews as favorable or unfavorable. Therefore, for this classification task we explored multiple classification models on above feature representations. We used the models ranging from the simple Logistic Regression to the stateof-art SVM Classifier. We also used other classification models like SGD Classifier and Random Forest Classifier. Apart from these, we also trained the above feature representations on Naïve Bayes' Classifier as this is primarily used in case of text mining in combination with Bag of Words and N-Gram Modelling. We also trained a model based on k-Nearest Neighbors to match the similarity between the reviews and classify them accordingly. For all of the above models, we used sklearn[11] modules by tuning their parameters and not changing their implementations and so we will not go into their theory in this report.

Before using the above feature representations for training classifiers, we tried reducing the size of representation set by using PCA on it. But it did not give us much improvement as the feature vector was reduced only by 15% and hence we did not incorporate those reductions. One important point to note is that for performance measure we are using Mean Absolute Error and not Mean Squared Error (MSE). This is because MAE will directly tell us the amount of misclassification we are doing for each model. Also as mentioned previously, we ran these training exercises to fit parameters on set selection using cross-validation techniques.

Results

As discussed above, we tried multiple classification models on various feature representations of the textual information in the reviews. Out of these SVM Classifier failed to even converge for all of our feature sets and hence we could not get a satisfactory answer for it. Among the remaining models, Logistic Regression model seemed to have best performance across all feature representations with classification accuracy around 89%. Also, k-Nearest Neighbors classifier had the worst accuracy of around 60% across all feature representations. The general order of performance for the model was LogisticRegression > NaïveBayes > SGDClassifier > RandomForestClassifier > kNNClassifier. For a given classifier, the model that performed best used a feature set of a mixture of unigrams and bigram.

	Naïve Bayes	**Rando**	**Logistic Regression**	**SGD Classifier**	**kNN**
50,000 Words	85.8	77.4	88.5	82.3	58.8
1,00,000 Words	85.9	76.8	88.6	83.4	58.7
– More than 1 occurrence	85.7	77.0	88.5	82.6	58.7
– More than 5 occurrence	85.6	77.5	88.4	82.3	58.6
Modelling	86.5	77.1	88.7	83.2	58.6
Bigram Mixed Modelling	87.8	77.4	90.4	84.1	60.2
N = 5	86.8	77.2	89.1	83.6	59.2
TF-IDF Modelling	85.4	76.4	89.2	83.5	59.3



Conclusions

From the results above, we can infer that for our problem statement, Logistic Regression Model with feature set using mixture of Unigrams and Bigrams is best. Apart from this, one can also use a Naïve Bayes' Classifier or a SGD classifier as they also provide good accuracy percentage. One peculiar thing to note is low accuracy with Random Forest classifier. This might be because of over-fitting of decision trees to the training data. Also, low accuracy of kNN Classifiers shows us that people have varied writing styles and kNN Models are not suited to data with high variance.

One of the major improvements that can be incorporated as we move ahead in this project is [3] merge words with similar meanings before training the classifiers . Another point of improvement can be to model this problem as a multi-class classification problem where we classify the sentiments of reviewers [14] in more than binary fashion like "Happy", "Bored", "Afraid", etc . This problem can be further remodeled as a regression problem where we can predict the degree of affinity for the movie instead of complete like/dislike.