LING L645: Advanced Natural Language Processing

FINAL PROJECT REPORT

TOPIC: Sentiment Analysis on IMDB Movie Reviews

Prof: Damir Cavar

Dwipam Kataria (ddkatari@umail.iu.edu)

Krishna Mahajan (krimahaj@indiana.edu)

Sanjana Agrawal (sa14593@umail.iu.edu)

1. Introduction

Several websites like IMDB, Rotten Tomatoes and Fandango aggregate user reviews on movies. The objective of this project is to apply some of the Advanced Natural Language Processing techniques learnt during the class to perform Sentiment Analysis on IMDB movie reviews and compare the performance of various Machine Learning classifications algorithms, thereby making the best model to obtain results with a high accuracy. The report consist of all the phases of the project starting from conceptual understanding of sentiment analysis to Data understanding to Data modeling.

2. Important Terminology (Background)

Sentiment Analysis:

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Sentiment analysis is widely applied to reviews and social media for a variety of applications, ranging from marketing to customer service.
A basic task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level—whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral. Advanced, "beyond polarity" sentiment classification looks, for instance, at emotional states such as "angry", "sad", and "happy".

BeautifulSoup4:

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with any parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

Bag-of-words model:

In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.
The bag-of-words model is commonly used in methods of document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier.
In practice, the Bag-of-words model is mainly used as a tool of feature generation. After transforming the text into a "bag of words", we can calculate various measures to characterize the text. The most common type of characteristics, or features calculated from the Bag-of-words model is term frequency, namely, the number of times a term appears in the text.

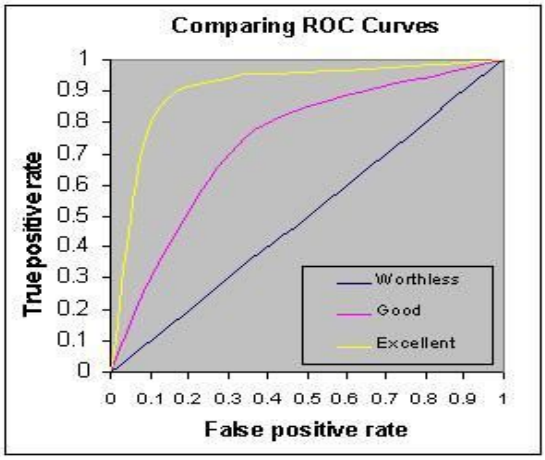
Word2vec:

This tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research. The word2vec tool takes a text corpus as input and produces the word vectors as output. It first constructs a vocabulary from the training text data and then learns vector representation of

words. The resulting word vector file can be used as features in many natural language processing and machine learning applications.

Area under ROC Curve:

The AUC is a common evaluation metric for binary classification problems. Consider a plot of the true positive rate vs the false positive rate as the threshold value for classifying an item as 0 or is increased from 0 to 1: if the classifier is very good, the true positive rate will increase quickly and the area under the curve will be close to 1. If the classifier is no better than random guessing, the true positive rate will increase linearly with the false positive rate and the area under the curve will be around 0.5.



3. Implementation and Results

The implementation task is divided into several parts.

Data Overview:

The dataset was obtained from Kaggle - Bag of words meets bags of popcorn online data science competition. This dataset consists of 50,000 movie reviews. Training set consists of 25,000 movie reviews that have been labeled either as positive (1) or negative (0). The test set also consists of 25,000 unlabeled movie reviews. In addition, we have also provided 50,000 unlabeled extra movie reviews.

Data Cleaning:

The data is very noisy. It consists of HTML_tags, abbreviations, smileys, punctuation marks and multiple punctuation marks like '!!!'. Hence, issues that one generally faces when processing text from online resources exist in this dataset. We first used a Python library called BeautifulSoup4 to remove HTML markups. We removed the punctuation marks and numbers using regular expressions. However, removing symbols like '!!!!' or ':(' resulted in a loss of useful information since multiple exclamation marks shows extra emphasis on a statement and the sad smiley shows disappointment thus expressing an important sentiment. We then converted all the words to lowercase and removed all the stop words. We used the Natural Language Toolkit

(NLTK) python package to tokenize all the words in the review and then lemmatized these tokens. Finally, we joined all these tokens to retrieve clean reviews.
We performed these tasks for the training, test and unlabeled extra dataset and appended the reviews after being cleaned, to the train and test sets respectively.

Data Modeling:

We experimented with several machine learning algorithms for each of the following approaches gave significant results for the Area under the ROC Curve (AUC):

- Bag of words (Term Frequency), one gram model
- Term Frequency - Inverse Document Frequency (TF-IDF), one gram model
- Term Frequency - Inverse Document Frequency (TF-IDF), two gram model
- Vector Averaging (Unsupervised learning using Google's Word2vec algorithm)
- Bag of Centroids (Unsupervised Learning using Google's Word2vec algorithm and K-means clustering)
- Ensemble of Naive Bayes and Gradient Descent

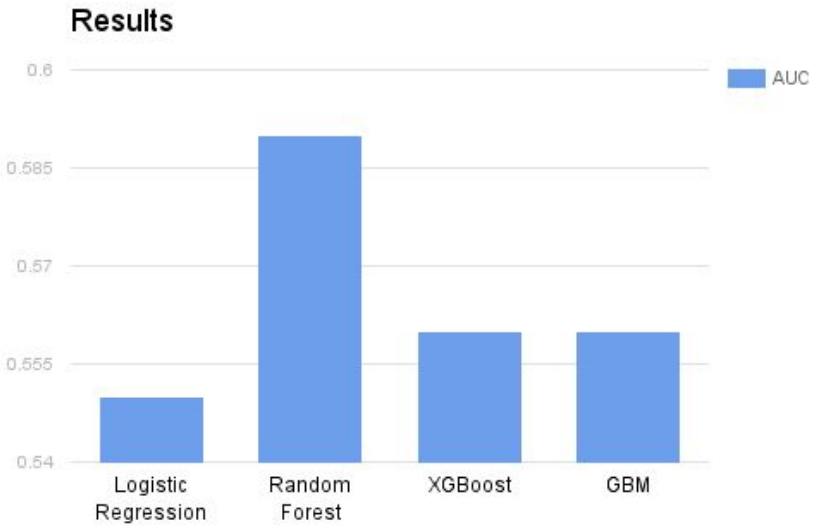
Several machine learning algorithms that were used for the above tasks to best fit the data are:

- **Logistic Regression**: It measures the relationship between the dependent variable and one or more independent variables by estimating probabilities using a logistic(sigmoid) function.
- **Random Forest**: They operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.
- **XGBoost and Gradient Boosted Trees**: XGBoost is short for "Extreme Gradient Boosting." The goal of this library is to push the extreme of the computation limits of machines to provide a scalable, portable and accurate library. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.
- **Naive Bayes:** It is a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.
- **Stochastic Gradient Descent: Stochastic gradient descent (SGD)**, also known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization method for minimizing an objective function that is written as a sum of differentiable functions. In other words, SGD tries to find minimums or maximums by iteration.

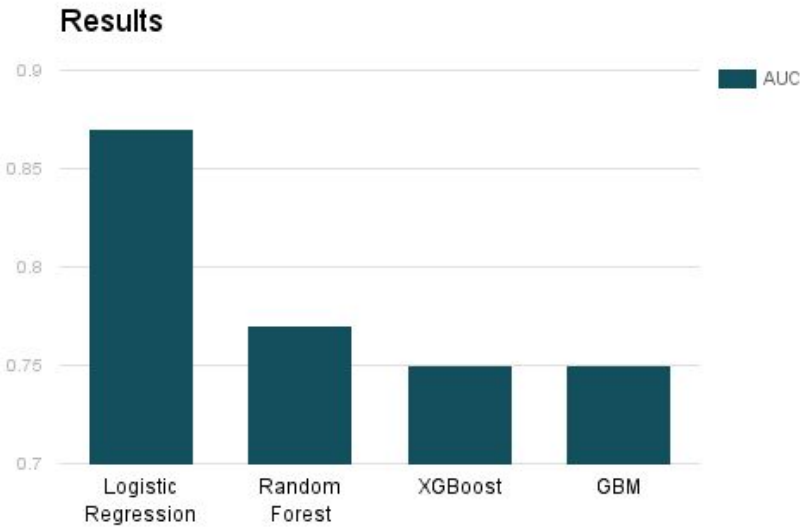
The approaches have been explained in detail with the results followed at the end of each algorithm. The evaluation metric we used was Area under the ROC Curve (AUC).

Bag of words (Term Frequency), one gram model: We used scikit-learn's module 'CountVectorizer' to vectorize each review using most frequent words. To limit the size of the vocabulary, we only considered 5K most frequent words in the reviews. This approach seemed

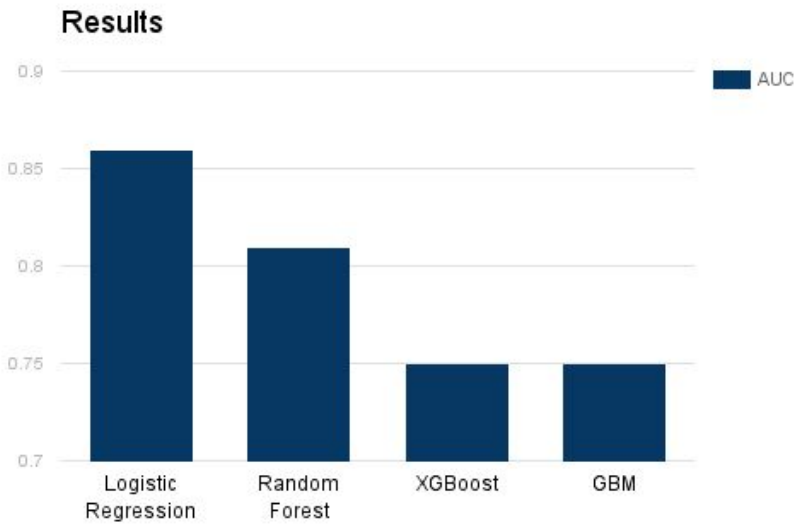
to be a good option as we had already removed the stopwords. Prior to fitting the model on the test data, we did some parameter tuning and cross validation to avoid overfitting and underfitting. However, the results were unsatisfactory as they contributed to the bottom 25% of the Kaggle scoreboard. The results have been formulated in the graph below.



Term Frequency - Inverse Document Frequency (TF-IDF), one gram model: Used scikit learn's module "TfidfVectorizer" to vectorize each review as per TF-IDF principle. We limited the max features to 5K. Again, prior to fitting the model on the test data, we did parameter tuning and cross validation. This approach resulted in a significant improvement compared to the previous model. Using logistic regression, we achieved a score of 0.87 which was in the top 30% of the Kaggle scoreboard. The results are shown below.



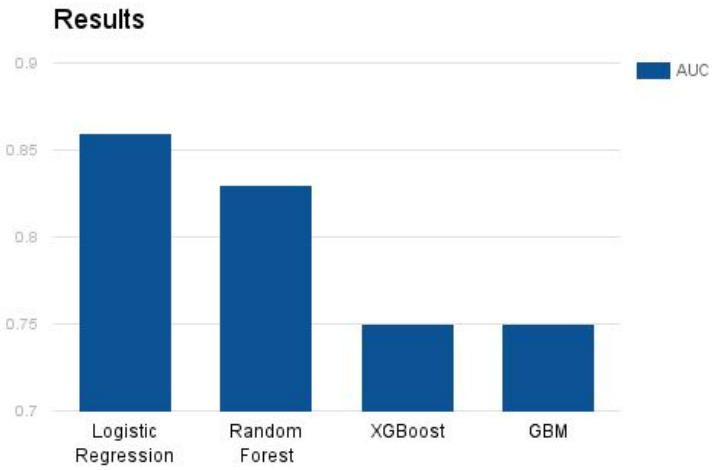
Term Frequency - Inverse Document Frequency (TF-IDF), two gram model: Using the same TF-IDF approach mentioned above, we fit the data using the Two Gram model and set our 'max_features' parameter to 12K. Compared to the previous model, after cross validation and parameter tuning, we observed that there was no significant improvement in Logistic Regression but Random Forests showed an improvement to 0.81. The results are shown below.



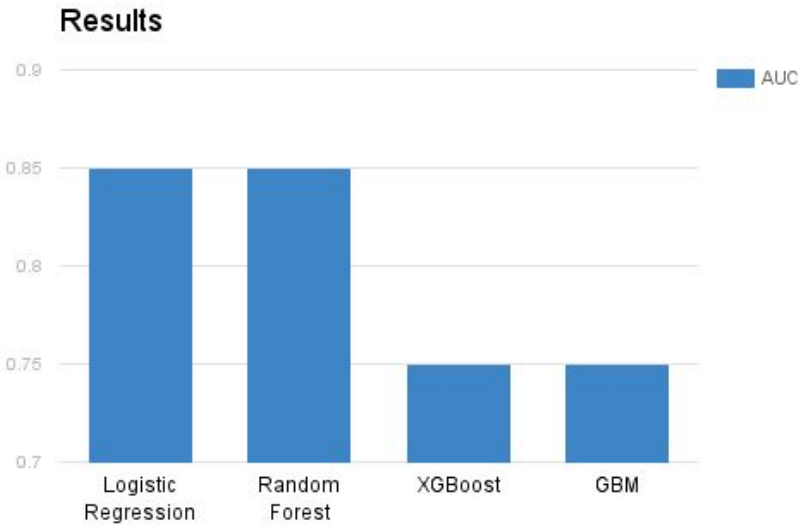
Vector Averaging (Word2vec model): After unimpressive results with supervised learning, we decided to shift to unsupervised learning algorithms such as Word2vec which takes into account the context of the sentences rather than just counting frequency of words and feeding to ML algorithm. So we used the extra unlabeled set of 50K reviews and converted all the reviews into approximately 0.8 million sentences (as Word2vec relies on similarity between sentences). We used Python's gensim library for the same.

After training the data using Word2vec, we obtained a vocabulary of 16000+ words. Each word is represented in a 300 dimensional feature. To represent a whole review in 300 dimensional features, we added all the vectors of the individual words in the review. We then averaged the review vector by dividing it with the number of words in sentences, thus the name Vector Averaging.

Disappointingly, this algorithm did not provide results as expected. It did not perform that much better than the Bag of Words approach. After research, we found out that this was because Word2vec provides exemplary performance only if it has an initial training data of around a billion sentences. Since we used only 0.8M sentences, the performance wasn't extraordinary but good enough. The results are shown below.



Bag of Centroids(Word2vec algorithm and K-means clustering): Word2Vec creates clusters of semantically related words, so another possible approach was to exploit the similarity of words within a cluster. We found the centers of these words using K-Means. Using trial and error, we decided to fix value of K = 1/5th of the vocabulary size. Each word in the 16000+ vocabulary belongs to one of the 3298 word clusters that was formed. As seen in Image 1, most of the clusters made sense. We transformed each review to a vector of size = # clusters. Entries of each vector = # words in that clusters.
Because of the same reasons stated above for the previous approach we used, a lack of sentences in the training set caused the algorithm to not give satisfactory results. Its results were comparable to the TF-IDF approach. The results are shown below.



```
Cluster 3
['biographical', 'somerset', 'sleuth', 'maugham', 'excalibur', 'adapte
d', 'originated', 'deathtrap', 'steinbeck', 'hellman', 'masterwork',
 'prose', 'macbeth', 'hammett', 'novella']

Cluster 4
['locks', 'holding', 'burying', 'locking']

Cluster 5
['pleasence', 'hare', 'donald', 'pleasance', 'sutherland']

Cluster 6
['motifs', 'cues']

Cluster 7
['pauline', 'davidson', 'sadie']
```

Image 1

Ensemble of Naive Bayes and Stochastic Gradient Descent: For the final approach, we decided to keep it simple and used an ensemble of Naive Bayes and Stochastic Gradient Descent. We preprocessed our data using the TF-IDF approach. We used scikit-learn's MultinomialNB as our classifier with alpha set to 0.0005. This approach gave us an excellent **AUC score of 0.97** and was amongst the top 10% rank on the Kaggle scoreboard. This was thus our simplest and fastest approach.

4. Conclusion and Future Scope

We thus used six different approaches to provide a comparative analysis amongst all of them. We learned the applications of several important algorithms used in Natural Language Processing. We achieved our best results with an ensemble of Naive Bayes and Gradient Descent with the area under the ROC curve being 0.97. As a future scope, we want to try to implement word2vec from scratch.

5. References

[1] https://en.wikipedia.org/wiki/Sentiment_analysis
[2] https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[3] https://en.wikipedia.org/wiki/Bag-of-words_model
[4] https://opensource.googleblog.com/2013/08/learning-meaning-behind-words.html
[5] https://code.google.com/archive/p/word2vec/
[6] http://gim.unmc.edu/dxtests/roc3.htm
[7] https://www.kaggle.com/wiki/AreaUnderCurve
[8] https://en.wikipedia.org/wiki/Random_forest
[9] https://github.com/dmlc/xgboost
[10] https://en.wikipedia.org/wiki/Naive_Bayes_classifier
[11] https://en.wikipedia.org/wiki/Stochastic_gradient_descent
[12] http://scikit-learn.org/stable/