Kaggle Challenge Analysis – Sharnam

Challenge: -

To preprocess the reviews in any way you find suitable and build your own ML model that can predict the sentiment of movie reviews. Train your model on the first 20000 reviews (with their sentiment as the target variable). The goal is to determine whether a given movie review has a positive or negative sentiment.
https://www.kaggle.com/c/word2vec-nlp-tutorial/overview

Raw Data and Pre-processing

The given labeled dataset is perfectly balanced. It has 12500 lines of positive sentiments and 12500 lines of negative sentiments.The data used for training the model is the first 20000 lines of the labeled data. Even these first 20000 rows are almost perfectly balanced , positive reviews - 9972 and negative reviews - 10028. Generally we consider the data not balanced at ratios more far apart than 60:40. Hence, the training data we have is really good.

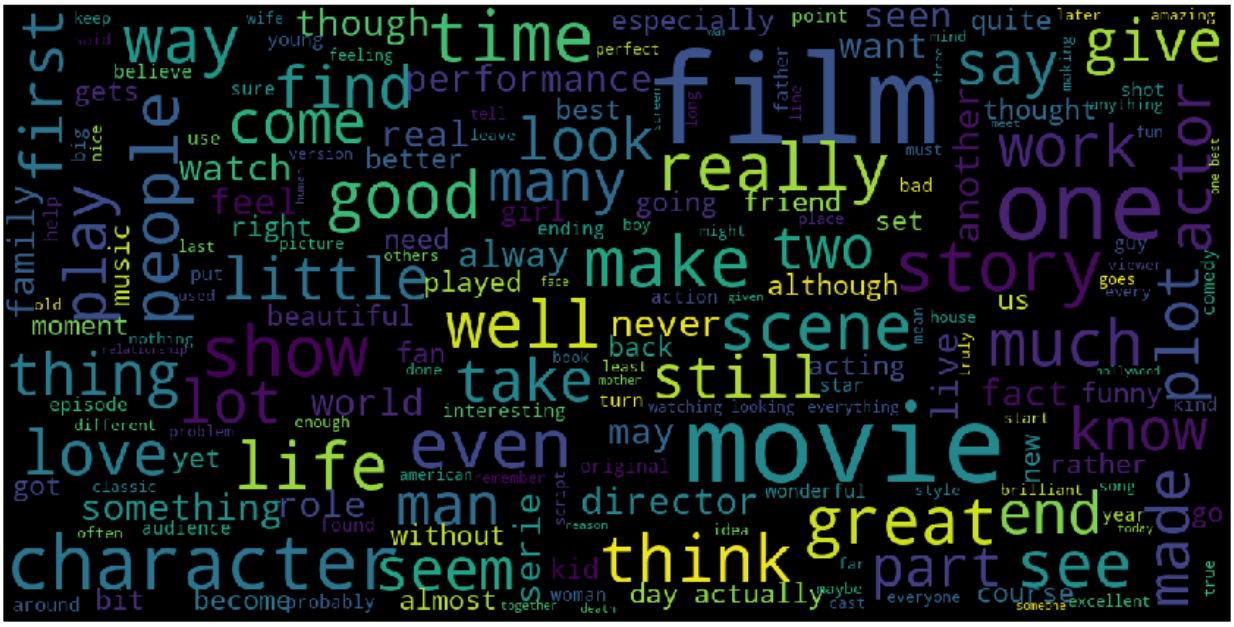
The average string length for all raw reviews without any pre-processing is 1333.166.

Now the data set is pre-processed. The HTML tags are removed. Then punctuations and exclamations are removed. (To develop a better algorithm we can actually use exclamations are strong sentiments, for simplicity this is ignored in the analysis). The reviews are then converted into words. Certain redundant words that do not convey any emotion are removed.

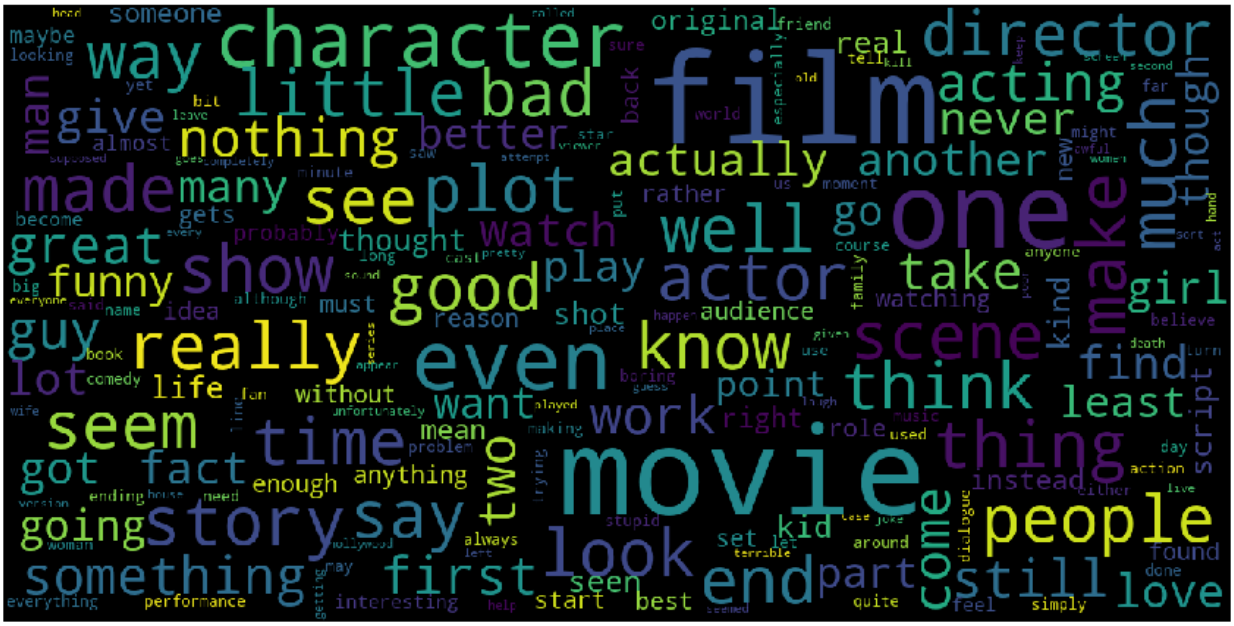
Word Cloud

A word-cloud for positive and negative sentiment reviews is made after removing redundant words

Word Cloud for positive sentiment reviews



Word Cloud for negative sentiment reviews



Just glancing as these clouds indicates that some words like film and movies are very common in both sentiments. Good and great could be strong indicators of positive reviews. Words like bad and nothing seem to be indicators of negative reviews.

Numeric Conversion and Machine Learning

The reviews are now converted to a numeric representation on which we can use machine learning. A choice was made between a normal bag of words model or a deep learning based approach to NLP. Deep learning based models can extract the underlying relationships between words and use this. For Deep learning models like word2Vec to work very well a large dataset is required. The Kaggle tutorial mentions that millions of rows of data would be required to give an excellent output through word2vec that exceeds the performance of Bag of Words. The tutorial mentions that the Bag of Words in most cases exceeds the performance of deep learning based model word2vec. A deep learning based model will be able to out-perform bag of words but would require a rigorous algorithm and larger set of data. The given unlabeled data set is just 50000 rows much lesser than the mentioned millions. Through online websites and kaggle competitions we can infer that bag of words is expected to give somewhere near 85% accuracy directly while other algorithms after a lot of evaluation and tuning can outperform by a few more percent. Some models have been able to achieve more than 90% accuracy. Depending on our accuracy/precision/roc requirement we can choose our model. Given the small data set size, the amount of tuning required to get good results with deep learning approaches and assuming that a decent model is fine for us a bag of words model is chosen to proceed with. The bag of words has just collected into an array the number of occurrences of each word in the reviews.

Testing Model Built

To implement an ML algorithm on this numeric data ,the random forest algorithm is used.
The Test data set does not have any sentiment labels. The bag of words and random forest are useful only for labeled data. As deep learning is not used the unlabeled data is not required. Hence to test our model the only available labeled data is in the last 5000 rows of the labeltraindata.tsv file. The random forest algorithm is tested with different parameters to train the model and then the model is compared to the sentiment values on the last 5000 rows of the labeled data. As ours is a balanced data set, the confusion matrix, accuracy and precision are compared. Depending on the model used, other measures like F1 or ROC may be a better comparison point.

A decent confusion matrix is obtained.
[[2125 347]
[383 2145]]
Model Accuracy is: 85.4
Model Precision is: 0.8607544141252006

With the trees as 100 around 0.83-0.84 accuracy is obtained with a precision of 0.85
With trees as 300 - 0.85 accuracy is obtained with precision around - 0.86
These values are decent and indicate that this method is fine.
The trees can be increased further to increase accuracy but the computational time would increase.The random forest can still be tuned better to get an estimated 1-2% more accuracy.

Predicting Test Data
Then this same ML model is run on the test data. The ML model predicts the sentiments of this unlabeled test data set. It then saves these along with the id's into a csv file.

Inferences
Overall, the obtained results with the bag of words vectorization and random forest classifier are decent. For classifiers, Adaboost or Logistic regression or other methods that are good with sparse data sets can be tested. To get even higher accuracies neural network based models can also be tested. With large datasets a deep learning based approach would be better.

Appendix - Console Output
Exact console Output for code

number of rows for sentiment 1 in whole dataset: 12500
number of rows for sentiment 0 in whole dataset: 12500
number of rows for sentiment 1 in the first 20000 rows: 9972
number of rows for sentiment 0 in the first 20000 rows: 10028
The average string length is: 1333.16635
Cleaning and parsing the training set movie reviews...

  review_text = BeautifulSoup(raw_review).get_text()
Review 1000 of 20000

Review 2000 of 20000

Review 3000 of 20000

Review 4000 of 20000

Review 5000 of 20000

Review 6000 of 20000

Review 7000 of 20000

Review 8000 of 20000

Review 9000 of 20000

Review 10000 of 20000

Review 11000 of 20000

Review 12000 of 20000

Review 13000 of 20000

Review 14000 of 20000

Review 15000 of 20000

Review 16000 of 20000

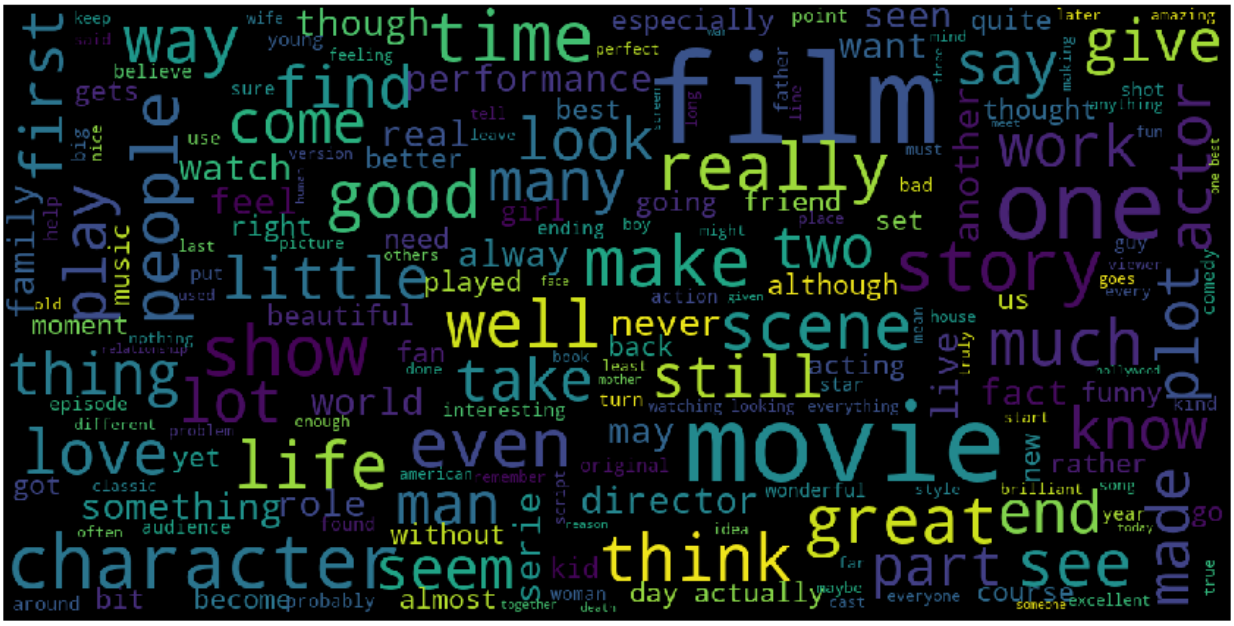
Review 17000 of 20000

Review 18000 of 20000

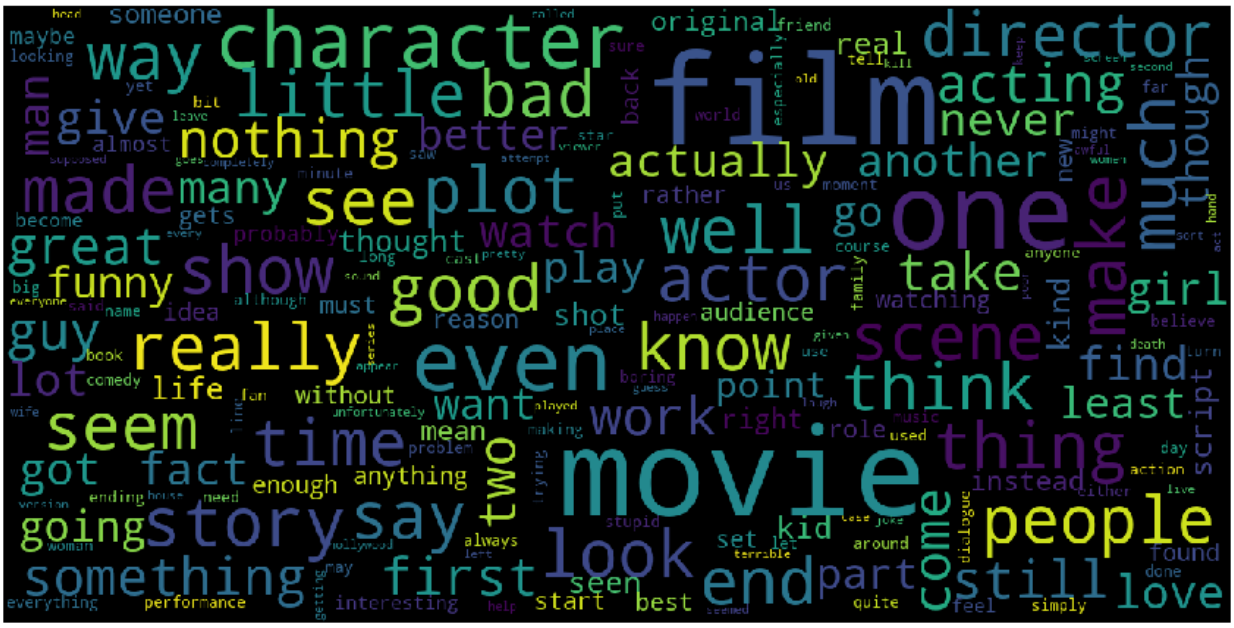
Review 19000 of 20000

Review 20000 of 20000

Showing the wordcloud for positive sentiments



Showing the wordcloud for negative sentiments



Training the random forest...
Cleaning and parsing the model trial 5000 label set movie reviews...
  review_text = BeautifulSoup(raw_review).get_text()
Review 1000 of 5000

Review 2000 of 5000

Review 3000 of 5000

Review 4000 of 5000

Review 5000 of 5000

Built Model is compared to the labels for the last 5000 rows of labeled train dataset
Confusion Matrix is: [[2125 347]
 [383 2145]]
Model Accuracy is: 85.4
Model Precision is: 0.8607544141252006
Cleaning and parsing the test set movie reviews...

  review_text = BeautifulSoup(raw_review).get_text()
Review 1000 of 25000

Review 2000 of 25000

Review 3000 of 25000

Review 4000 of 25000

Review 5000 of 25000

Review 6000 of 25000

Review 7000 of 25000

Review 8000 of 25000

Review 9000 of 25000

Review 10000 of 25000

Review 11000 of 25000

Review 12000 of 25000

Review 13000 of 25000

Review 14000 of 25000

Review 15000 of 25000

Review 16000 of 25000

Review 17000 of 25000

Review 18000 of 25000

Review 19000 of 25000

Review 20000 of 25000

Review 21000 of 25000

Review 22000 of 25000

Review 23000 of 25000

Review 24000 of 25000

Review 25000 of 25000

References

https://www.kaggle.com/c/word2vec-nlp-tutorial/overview
https://github.com/MohammadWasil/Sentiment-Analysis-IMDb-Movie-Review
https://towardsdatascience.com/sentiment-analysis-with-python-part-1-5ce197074184
https://medium.com/machine-learning-101/https-medium-com-savanpatel-chapter-6-adaboost-classifier-b945f330af06