Final Evaluation Report: Sentiment Analysis on Movie Reviews

Talha Hanif Butt, L18-1864, FAST-NUCES, Lahore

December 9, 2018

1 Abbreviations

AUC_ROC Area under the curve of Receiver Operating Characteristic
Acc Accuracy
BAC Balanced Accuracy Rate
Naive Bayes Naive Bayes Multinomial Classifier
LSVC Linear Support Vector Classifier
ANN Artificial Neural Network
LSTM Long Short Term Memory

2 Introduction

This is the final evaluation report of the term project for the Introduction to Data Science: Tools and Techniques course providing literature review, data exploration, proposed methodology, experimental setup, results and conclusion related to the problem of sentiment analysis on movie reviews.

3 Literature Review

"There's a thin line between likably old-fashioned and fuddy-duddy, and The Count of Monte Cristo ... never quite settles on either side."

The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis, originally collected by Pang and Lee [1]. In their work on sentiment treebanks, Socher et al. [2] used Amazon's Mechanical Turk to create fine-grained labels for all parsed phrases in the corpus. This competition presents a chance to benchmark your sentiment-analysis ideas on the Rotten Tomatoes dataset. You are asked to label phrases on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive. Obstacles like sentence negation, sarcasm, terseness, language ambiguity, and many others make this task very challenging.

4 Dataset

The dataset is comprised of tab-separated files with phrases from the Rotten Tomatoes dataset. The train/test split has been preserved for the purposes of benchmarking, but the sentences have been shuffled from their original order. Each Sentence has been parsed into many phrases by the Stanford parser. Each phrase has a PhraseId. Each sentence has a SentenceId. Phrases that are repeated (such as short/common words) are only included once in the data.

train.tsv contains the phrases and their associated sentiment labels. A SentenceId is also provided so that the track of which phrases belong to a single sentence can be maintained.

test.tsv contains just phrases.

The sentiment labels are:

0 - negative

1 - somewhat negative

2 - neutral

3 - somewhat positive

4 - positive

4.1 Data Exploration

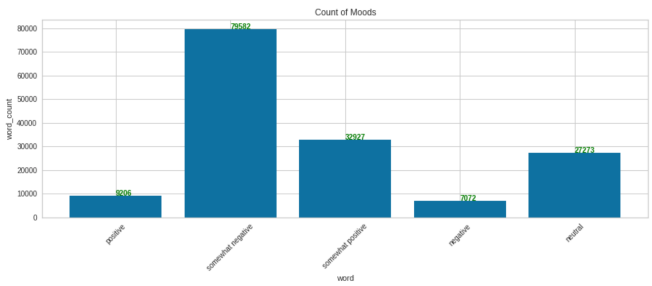

Figure 1: Class Distribution in terms of Label Counts

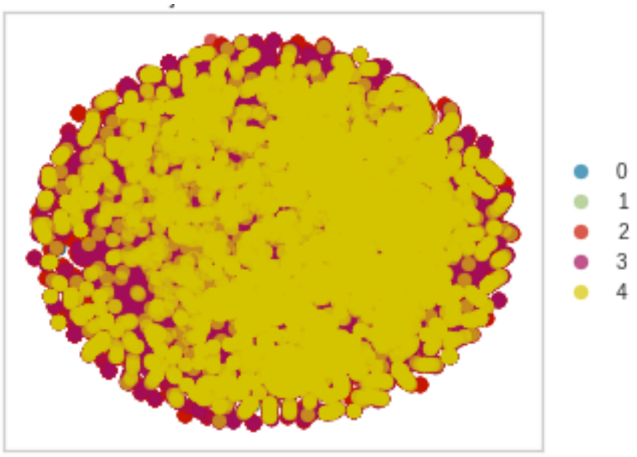

Figure 2: Class Distribution using TF-IDF

word	word_count
film	7712
movie	7586
one	3958
like	3211
character	2900
story	2850
time	2472
rrb	2438
make	2433
good	2304

Figure 4: Top 10 Words in the Dataset

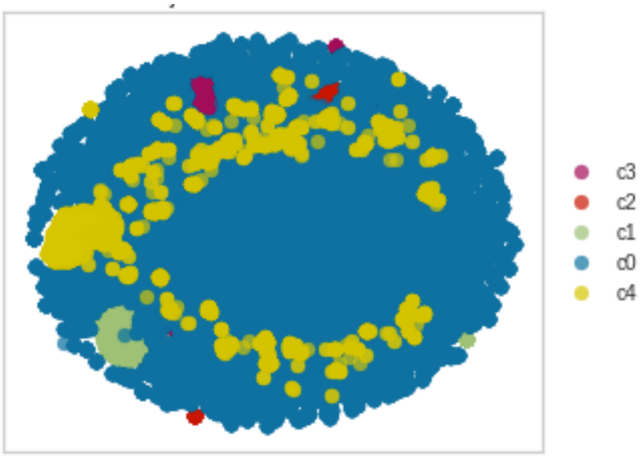

Figure 3: Class Distribution using K-Means

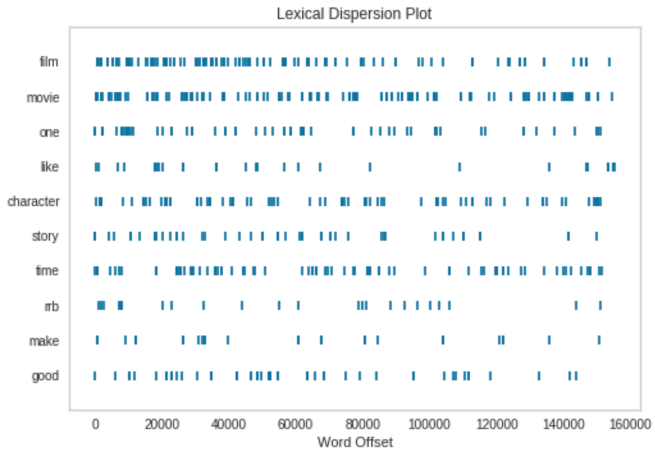

Figure 5: Dispersion of Top 10 Words

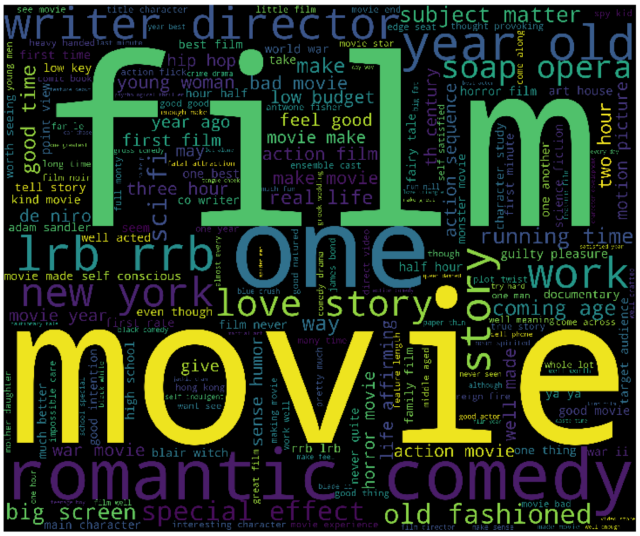

Figure 6: Word Cloud of the Dataset

5 Proposed Methodology

Four methods have been applied which are as follows:

1. TF-IDF + Naive Bayes Multinomial Classifier
2. Linear Support Vector Classifier (LSVC)
3. Artificial Neural Network (ANN)
4. Long Short Term Memory (LSTM)

5.1 TF-IDF + Naive Bayes Multinomial Classifier

5.1.1 TF-IDF

TF*IDF is an information retrieval technique that weighs a terms frequency (TF) and its inverse document frequency (IDF). Each word or term has its respective TF and IDF score. The product of the TF and IDF scores of a term is called the TF*IDF weight of that term.

Put simply, the higher the TF*IDF score (weight), the rarer the term and vice versa.

The TF*IDF algorithm is used to weigh a keyword in any content and assign the importance to that keyword based on the number of times it appears in the document. More importantly, it checks how relevant the keyword is throughout the web, which is referred to as corpus.

For a term t in a document d, the weight Wt,d of term t in document d is given by:

Wt,d = TFt,d log (N/DFt)

Where:

TFt,d is the number of occurrences of t in document d. DFt is the number of documents containing the term t. N is the total number of documents in the corpus.

The TF (term frequency) of a word is the frequency of a word (i.e. number of times it appears) in a document.

For example, when a 100 word document contains the term cat 12 times, the TF for the word cat is

TFcat = 12/100 i.e. 0.12

The IDF (inverse document frequency) of a word is the measure of how significant that term is in the whole corpus.

For example, say the term cat appears x amount of times in a 10,000,000 million document-sized corpus (i.e. web). Lets assume there are 0.3 million documents that contain the term cat, then the IDF (i.e. log DF) is given by the total number of documents (10,000,000) divided by the number of documents containing the term cat (300,000).

IDF (cat) = log (10,000,000/300,000) = 1.52

Wcat = (TF*IDF) cat = 0.12 * 1.52 = 0.182

5.1.2 Naive Bayes Multinomial Classifier

The general term Naive Bayes refers the strong independence assumptions in the model, rather than the particular distribution of each feature. A Naive Bayes model assumes that each of the features it uses are conditionally independent of one another given some class. More formally, if I want to calculate the probability of observing features f1 through fn, given some class c, under the Naive Bayes assumption the following holds:

$p(f_1, ..., f_n|c) = \prod_{i=1}^{n} p(f_i|c)$

This means that when I want to use a Naive Bayes model to classify a new example, the posterior probability is much simpler to work with:

$p(c—f_1, ..., f_n) \propto p(c)p(f_1|c)...p(f_n|c)$

Of course these assumptions of independence are rarely true, which may explain why some have referred to the model as the "Idiot Bayes" model, but in practice Naive Bayes models have performed surprisingly well, even on complex tasks where it is clear that the strong independence assumptions are false.

Up to this point we have said nothing about the distribution of each feature. In other words, we have left p(fi—c) undefined. The term Multinomial Naive Bayes

simply lets us know that each p(fi—c) is a multinomial distribution, rather than some other distribution. This works well for data which can easily be turned into counts, such as word counts in text.

The distribution you had been using with your Naive Bayes classifier is a Guassian p.d.f., so I guess you could call it a Guassian Naive Bayes classifier.

In summary, Naive Bayes classifier is a general term which refers to conditional independence of each of the features in the model, while Multinomial Naive Bayes classifier is a specific instance of a Naive Bayes classifier which uses a multinomial distribution for each of the features.

5.2 Linear Support Vector Classifier (LSVC)

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well.

5.3 Artificial Neural Network (ANN)

The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living neurons and dendrites.

The human brain is composed of 86 billion nerve cells called neurons. They are connected to other thousand cells by Axons. Stimuli from external environment or inputs from sensory organs are accepted by dendrites. These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send the message to other neuron to handle the issue or does not send it forward.

ANNs are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value.

Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values.

5.4 Long Short Term Memory (LSTM)

Long Short Term Memory networks usually just called LSTMs are a special kind of RNN, capable of learning long-term dependencies.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

The key to LSTMs is the cell state.

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. Its very easy for information to just flow along it unchanged.

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means let nothing through, while a value of one means let everything through!

An LSTM has three of these gates, to protect and control the cell state.

6 Experimental Setup

The dataset was divided in to train and test sets using a ratio of 70, 30 respectively. TF-IDF vectorizer was used as a preprocessing step for Naive Bayes Multinomial Classifier and LSVC while Stemming and Padding for LSTM and StopWords filtering for all the methods. ANN and LSTM use a batch size of 512. For plotting the AUC_ROC, labels were binarized.

7 Comparative Analysis

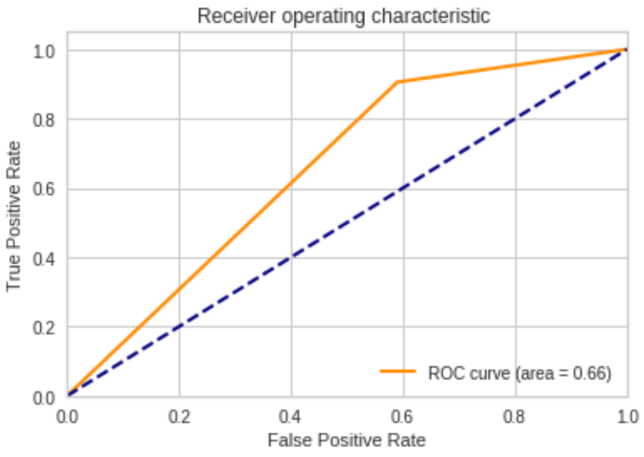

Figure 7: AUC_ROC for Naive Bayes

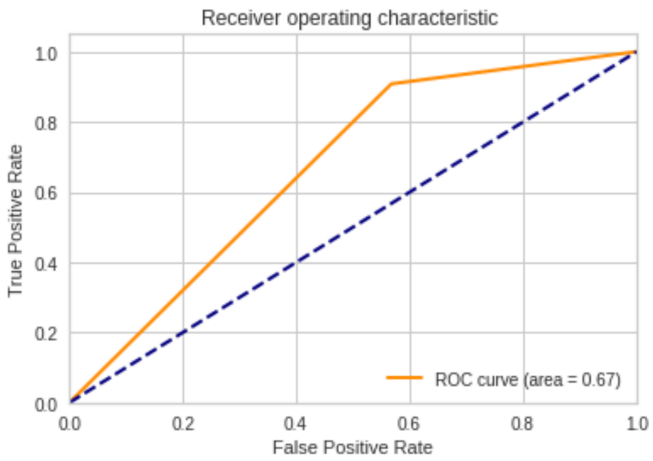

Figure 8: AUC_ROC for LSVC

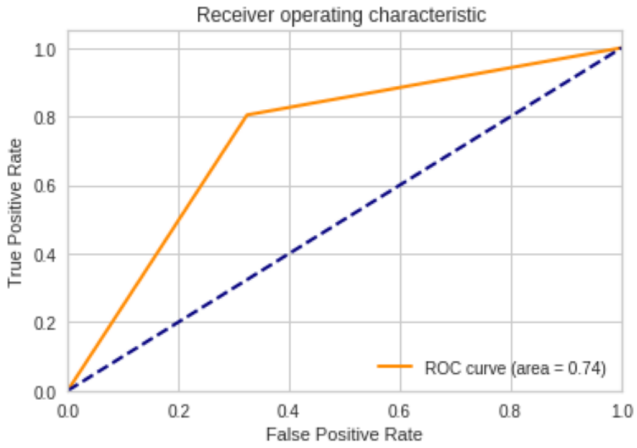

Figure 9: AUC_ROC for Artificial Neural Network

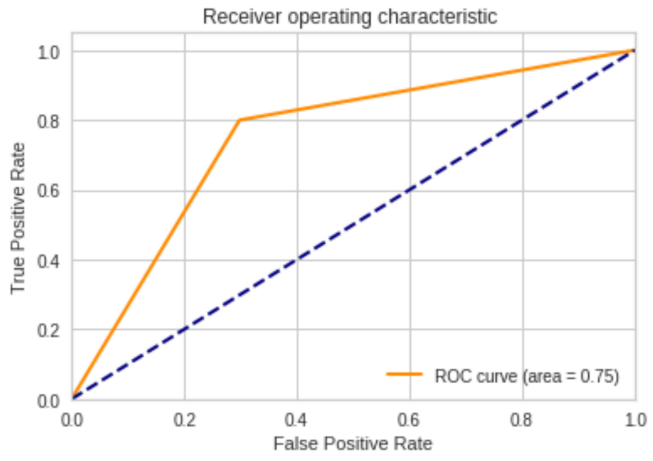

Figure 10: AUC_ROC for LSTM

Model	Acc	BAC	AUC_ROC
Naive Bayes	0.591	0.364	0.66
LSVC	0.595	0.372	0.67
ANN	0.634	0.407	0.74
LSTM	0.652	0.528	0.75

Table 1: Performance Comparison

8 Results

Naive Bayes Multinomial Classifier provides an accuracy of 0.591, LSVC provides an accuracy of 0.595, NN provides an accuracy of 0.634 and LSTM has an accuracy of 0.652 with BAC rate of 0.364, 0.372, 0.407 and 0.528 while AUC_ROC of 0.66, 0.67, 0.74 and 0.75 respectively.

9 Conclusion

LSTM has proved to be the best among the models applied and it shows that the three gates for managing information and the capability of LSTM's to learn long term dependencies has helped the nwetwork to better classify the reviews.

10 References

[1] Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In ACL, pages 115124.

[2] Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Chris Manning, Andrew Ng and Chris Potts. Conference on Empirical Methods in Natural Language Processing (EMNLP 2013).

[3] Stuart J. Russell and Peter Norvig. 2003. Artificial Intelligence: A Modern Approach (2 ed.). Pearson Education. See p. 499