Sentiment Analysis on IMDB reviews
ECE 684 - Final Project

Aniket Dalvi, Vinayak Gupta, Vaishali Jain

November 24, 2020

Contents

1 Introduction

For the final project of the natural language processing course we chose to solve the problem of sentiment analysis. Sentiment Analysis is one the most common text classification tool where a text is analyzed to determine whether the underlying sentiment is positive, negative our neutral. In our case, we consider only positive and negative sentiments. This was done on a dataset comprising of IMDb movie reviews. We used two approaches to solve this problem as was outlined in the requirements for the project. The first one is the Multinomial Naive-Bayes algorithm using bag of words as features, a generative model. Subsequently we used an LSTM neural network, which is a discriminative model. In the following sections we first describe the dataset and the pre-processing that was required. We then briefly touch up on the conceptual ideas behind the algorithm, while also spending some time describing how we used these to solve the problem at hand.

2 Data

For this problem we used the Stanford dataset comprising of 50000 movie reviews taken from IMDb. The dataset comprises of reviews with a a label associated with it indicating whether the sentiment associated with the review is positive or negative. It is to be noted that this is a balanced dataset with 25000 positive reviews and 25000 negative reviews. Before we could feed this data into our models, we had to pre-process it. This pre-processing consisted of multiple stages:

- Remove all html tags using `BeautifulSoup`
- Remove any special characters from the reviews that are not alpha-numeric since they do not convey much sentiment
- Tokenize data using NLTK `TokTokTokenizer`
- Remove stop words that do not convey any sentiment. Examples: 'i','me', 'my', 'myself', 'between','into'
- Stemmer is then applied since we only care about the stem of the word and it can help reduce a word's variants to the stem

A pre-processing example of a review has been shown below:

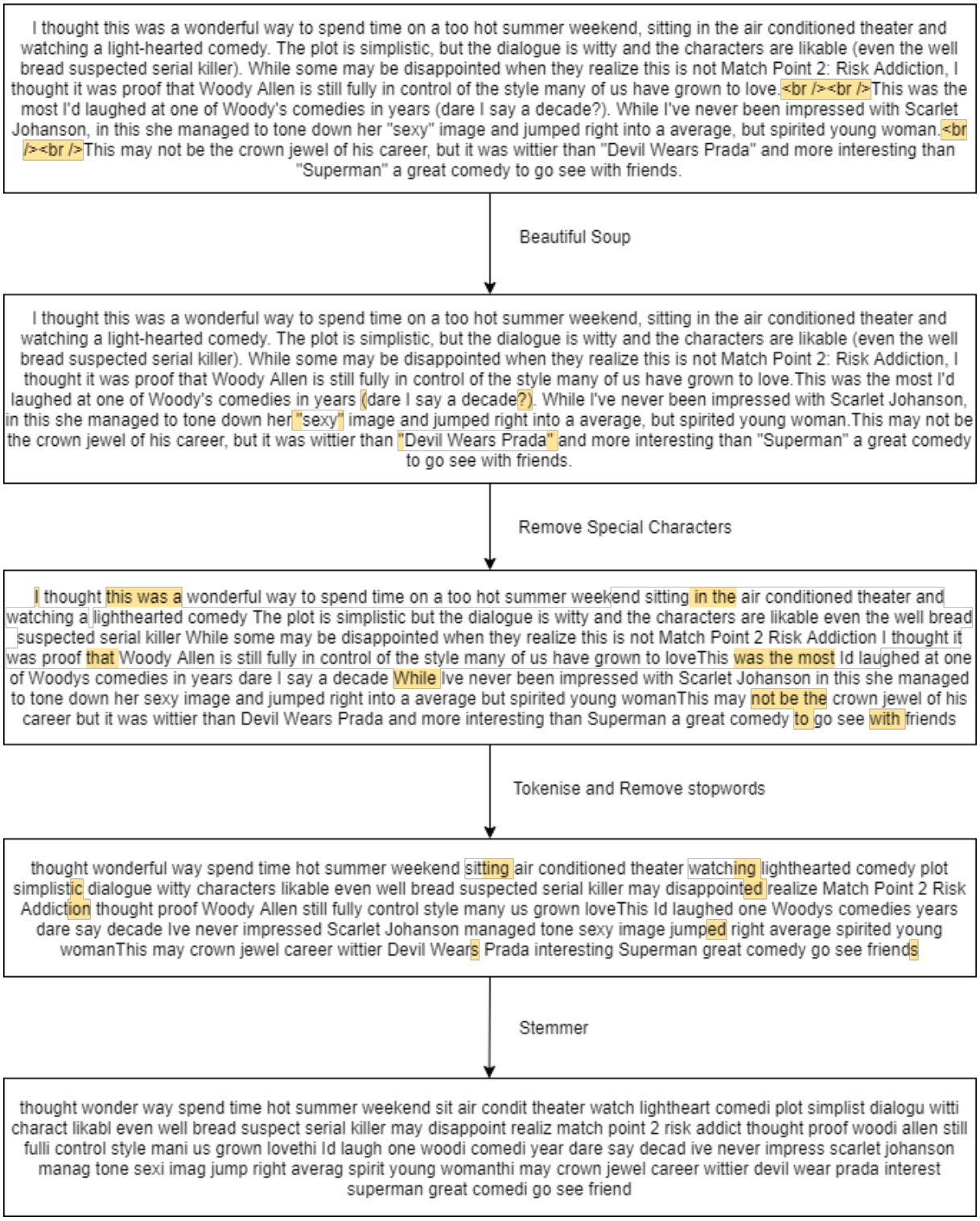

Figure 1: Data Processing

3 Generative Probabilistic Model

In the generative model we use the Naive-Bayes algorithm to make sentiment prediction. This algorithm primary revolves around the Bayes' rule which can be stated as follows:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

where X and y are random variables. Now consider an instance X of our data with given parameters $x_1, x_2, \ldots x_n$, then this can be represented as $X = (x_1, x_2, \ldots x_n)$. Then by Bayes' rule we get:

$$P(y|x_1, x_2, \ldots x_n) = \frac{P(x_1|y), P(x_2|y) \ldots P(x_n|y)P(y)}{P(x_1), P(x_2) \ldots P(x_n)}$$

where y is a label associated with the data and $x_1, x_2, \ldots x_n$ are assumed to be independent. Here we see

$$P(y|x_1, x_2, \ldots x_n) \propto P(y)P(x_1|y), P(x_2|y) \ldots P(x_n|y).$$

Therefore, the predicted label using this algorithms will just be the label y that maximizes this quantity:

$$y = argmax_y P(y)P(x_1|y), P(x_2|y) \ldots P(x_n|y).$$

In the sentiment analysis scenario y just takes two values 0 and 1 for negative and positive reviews respectively. A generative probabilistic model can also be use to generate a dataset, and this has been further discussed in a later section

4 Discriminative Model

In the discriminative approach to perform sentiment analysis, we use Long Short Term Memory (LSTM) model. A typical recurrent neural network (RNN) can be understood as multiple copies of a neural network that pass a message to the next instance being processed. The intuitive idea behind this is to transfer relevant information across instances to make better predictions.

However, the drawback with a typical RNN is that the only information being transmitted is between consecutive instances with a lot of larger context lost. This is where an LSTM model comes in handy, a typical architecture of which has been shown in figure 2. In an LSTM model there are 2 messages passed between instances, the first being a long term message as demonstrated by the top horizontal vector in the example network cell. This one goes through very minimal alterations as it is passed between instances. The second message contains more immediate information from the previous instance and typically goes through significant number of functions before being transmitted. In the figure, this is represented by the bottom horizontal vector in the cell. In case of sentiment analysis of movie reviews, we care about the larger context of the review, not just the previous word, thereby justifying the use of an LSTM rather than a typical RNN. We use Glove embedding as our word-embedding algorithm.

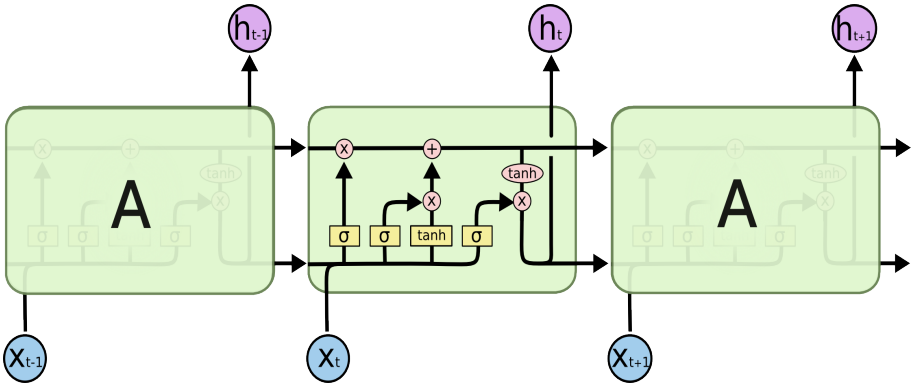

Figure 2: LSTM Architecture

5 Synthetic Data Generation

Since the Multinomial Naive Bayes model is a generative probabilistic model, we can create synthetic data from it. In our case, we have two classes namely positive and negative sentiment. When the Naive Bayes model is trained on real data (supervised learning), it learns the conditional probability distribution of the words/ n-grams (depends on how the model features are used) given the sentiment. This learned distribution can then be used for synthetic data generation purposes. Suppose our review contains 3 features (in our case it is bag of words created from the review) (x_1, x_2, x_3). Then the features are learned by the following rule:

$$P(x_1, C) = \frac{count(x_1|C)}{\sum_k count(x_k|C)}$$

where C can be positive or negative depending on the type of review and k ranges over all the features.

Once this probability distribution is learned, generating data from this distribution becomes trivial. All we need to do is specify the type of review we want to generate (positive or negative) and sample words from the distribution $P(X|C)$. While every review can have different number of words, we have decided to stick with 30 such generations for each review. Note that since our bag of words features are created from n-grams ranging from 1 to 3, the generation of every review might not exactly contain 30 words but would vary depending on the words selected. We create 1000 such reviews due to computational/time constraints to create our generated data and test our Naive Bayes model as well as the LSTM model on this generated data and compare our results.

6 Results

We split the data into training and test using 80-20 split.

6.1 Real Data

On running the Naive Bayes model, an accuracy of around 74% was obtained on the real ImDb movie reviews data. The confusion matrix for the Naive Bayes model has been shown below:

		Prediction outcome	
		p	**n**
Actual value	**p′**	True Positive (3622)	False Negative (1417)
	n′	False Positive (1186)	True Negative (3775)

The classification report for the Naive Bayes model has also been shown below:

	precision	recall	f1-score	support
Negative	0.73	0.76	0.74	4961
Positive	0.75	0.72	0.74	5039
accuracy			0.74	10000
macro avg	0.74	0.74	0.74	10000
weighted avg	0.74	0.74	0.74	10000

Figure 3: Naive Bayes Classification Report

As for LSTM, the following two graphs show the accuracy and loss on the training and test set over 6 epochs (each epoch consists of 250 passes).

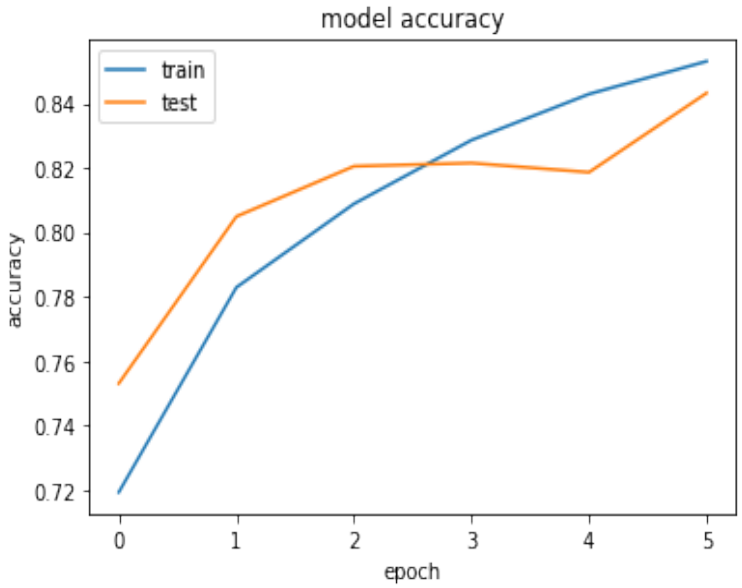

Figure 4: LSTM Model Accuracy Curve

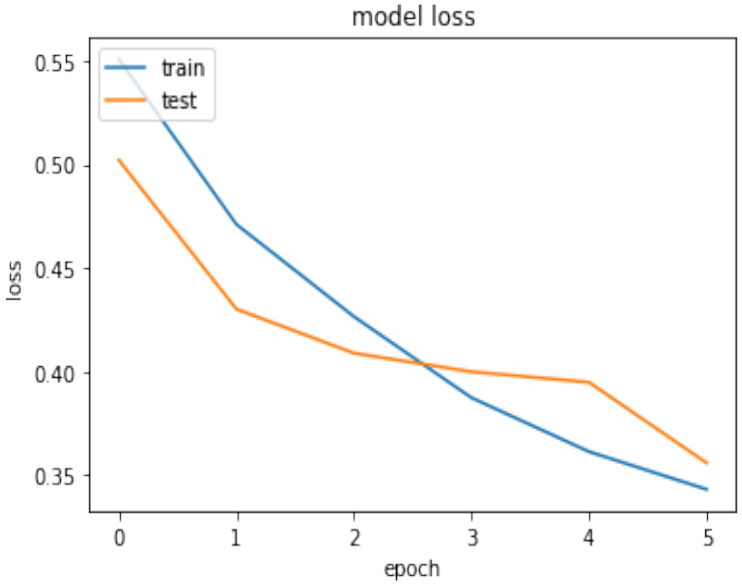

Figure 5: LSTM Model Loss Curve

The final accuracy obtained on test set was around 84%. The confusion matrix for the LSTM model has been shown below:

		Prediction outcome	
		p	**n**
Actual value	**p′**	True Positive (3954)	False Negative (1085)
	n′	False Positive (581)	True Negative (4380)

The classification report for the LSTM model has also been shown below:

	precision	recall	f1-score	support
Negative	0.85	0.84	0.84	4961
Positive	0.84	0.86	0.85	5039
accuracy			0.85	10000
macro avg	0.85	0.85	0.85	10000
weighted avg	0.85	0.85	0.85	10000

Figure 6: LSTM Classification Report

6.2 Synthetic Data

We created 1000 reviews from our Naive Bayes model where every review contained 30 n-grams between 1 and 3. The generation took nearly 2 days. On running the Naive Bayes algorithm on the generated data, an accuracy of around 58% was achieved. This made sense given the small amount of data that we had and the fact that Naive Bayes model was retrained on this data. Again, a train-test split of 80-20 was used. The confusion matrix for the NB model on generated data is shown below:

		Prediction outcome	
		p	**n**
Actual value	**p′**	True Positive (51)	False Negative (45)
	n′	False Positive (39)	True Negative (65)

The classification report for the NB model was:

	precision	recall	f1-score	support
Positive	0.59	0.62	0.61	104
Negative	0.57	0.53	0.55	96
accuracy			0.58	200
macro avg	0.58	0.58	0.58	200
weighted avg	0.58	0.58	0.58	200

Figure 7: Synthetic Data NB Classification Report

As for LSTM, the following two graphs show the accuracy and loss on the training set over 20 epochs(each epoch contains 20 passes):

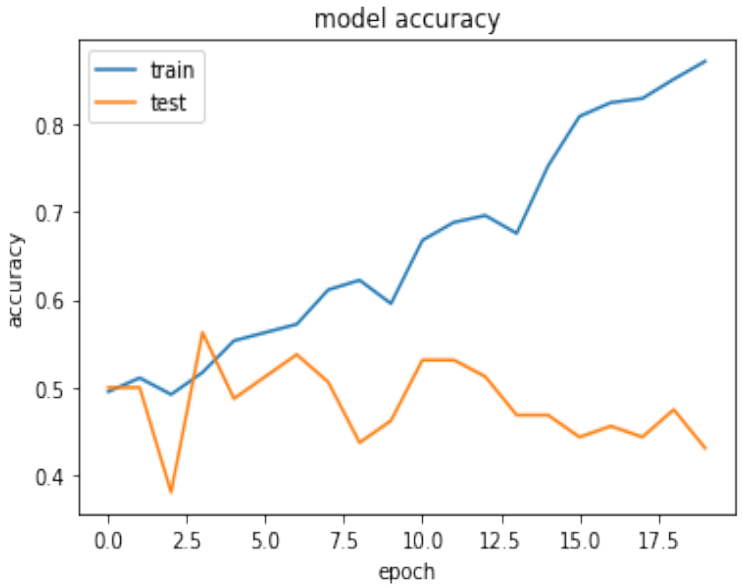

Figure 8: LSTM Model Synthetic Data Accuracy Curve

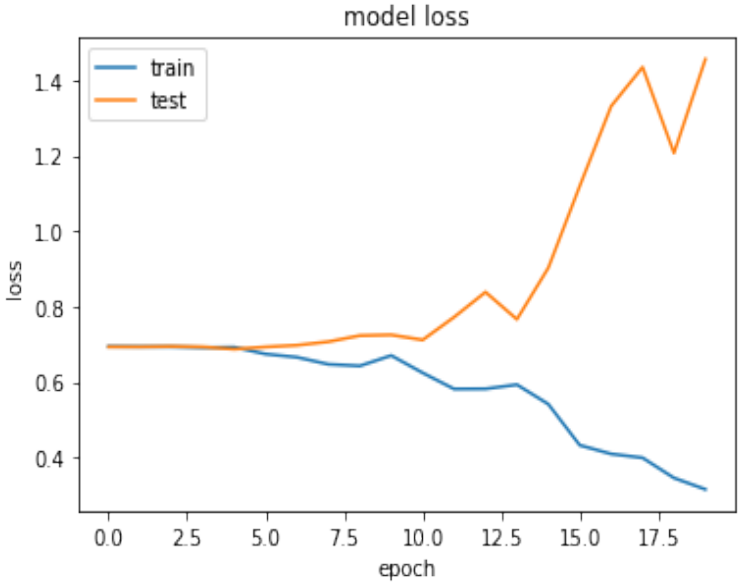

Figure 9: LSTM Model Synthetic Data Loss Curve

The final accuracy obtained on test set was around 50% . The confusion matrix and classification report for LSTM is shown below:

Prediction outcome

Actual value	p	n
p′	True Positive (30)	False Negative (66
n′	False Positive (34)	True Negative (70)

	precision	recall	f1-score	support
Negative	0.48	0.41	0.45	104
Positive	0.45	0.52	0.48	96
accuracy			0.47	200
macro avg	0.47	0.47	0.46	200
weighted avg	0.47	0.47	0.46	200

Figure 10: Synthetic Data LSTM Classification Report

6.3 Model's (trained on real data) performance on synthetic data

Rather than training on synthetic data, we used the model trained on real data to evaluate on synthetic data and found that for NB, we obtain an accuracy of 94.9% which makes sense since the synthetic data was generated from that trained model. The confusion matrix is:

Prediction outcome

Actual value	p	n
p′	True Positive (478)	False Negative (22)
n′	False Positive (29)	True Negative (471)

As for the LSTM model, an accuracy of 57 % was obtained on the generated data. The confusion matrix is shown below:

		Prediction outcome	
		p	**n**
Actual value	**p′**	True Positive (171)	False Negative (329)
	n′	False Positive (100)	True Negative (400)

7 Discussion

In terms of performance and correctness, LSTM neural network based classifier performs better than Naive Bayes on Real data. This is completely understandable as the LSTM generates it's prediction from the context of the entire review, while a probabilistic model such as LSTM is limited by multiple factors such as the assumption that each of the features (in our case bag of words) are assumed to be independent, which is definitely not true in the english language. Furthermore, if test instances contain words which were not seen in training data, the generative model is likely to fare worse as it depends purely on probabilistic numbers.

In terms of data and time, LSTM model needs more time to train and large amount of data in order to get a model which performs well consistently. For the naive Bayes classifier, however, the large amount of data isn't necessity. However, the more the data, the better would be the accuracy as it a larger dataset will reflect more accurate conditional probabilities of the instances and their parameters.

Interpretability can be intuitively understood as the extent to which a human can understand the cause of a decision by the model. It is therefore quite straight forward to see that, in terms of interpretability, the naive-Bayes model is better. Considering that it is a probabilistic model which predicts the sentiment based on how many such occurrences have occured given the parameters, it highly intuitive for a human to understand the cause of the decision. However, the LSTM model, or most neural networks in general, largely work a black-box with very limited understanding of how the parameters are being weighed, and what message is being passed nodes in order to arrive at the decision. This also explains why creating a neural network is largely an integrative trial-and-error process in order to arrive at the right model which performs well consistently.

After NB model is trained on real data, we generate some important features for each positive and negative sentiment and while most of them do not make sense because we had

use stemming and the fact that there are so many features, we can see for positive sentiment, that words like "greatest show" have good importance.

Once the data was generated, we use the model trained on real data to see how both of our models was performing. It is obvious that the NB model performed better and achieved an accuracy of 94 % since the data was generated using the same model. The trained LSTM model was only able to achieve 57% accuracy.

When the models were trained only on the synthetic data, then both models performed very poorly since we only had 800 training samples. LSTM models are data intensive so they were not able to learn the data well whereas for NB as well, its better to have larger data so it also did not perform very good.

8 References

- https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c
- https://christophm.github.io/interpretable-ml-book/interpretability.html
- https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
- https://www.kaggle.com/lakshmi25npathi/sentiment-analysis-of-imdb-movie-reviews
- https://stackabuse.com/python-for-nlp-movie-sentiment-analysis-using-deep-learning-in-keras/