ECE 684 Fall 2021: Natural Language Processing
Sentiment Analysis: Movie Review Classification

Sayan Mandal - sm693
Arjun Sridhar - as1116

1 STEP 1: IMDB Review Sentiment Analysis

We decided to IMDB review sentiment analysis for our project. IMDB is in an online movie review platform with hundreds of thousands of reviews for a diverse selection of movies. These reviews can be classified as either positive or negative. We would like to design an NLP solution to this problem and design a system to classify reviews as positive or negative based on only the text. IMDB has released a public dataset consisting of 50,000 reviews. We will go into more detail about the dataset and exploratory analysis in the next section.

1.1 Exploratory Data Analysis

1.1.1 Dataset description

We are using the IMDB Movie Review dataset which has 50000 highly polar movie reviews generated from the original Large Movie Review Dataset by Stanford AI. The dataset has two columns, the $Review$ column with movie reviews with varying length and target variable $sentiment$ with two classes – $positive$ and $negative$, class equally distributed among the reviews.

1.1.2 Data Preprocessing

Before we move to model building, we need to preprocess the text by removing HTML tags, brackets, punctuation and special characters. Additionally we removed stop words and applied stemming to generate clean reviews.

1.1.3 Denoising the text

The raw text reviews have HTML tags, brackets, punctuation and special characters. All these do not add any information to the natural language classification model. We use Python package BeautifulSoup4 to remove HTML Tags, Regex to remove square brackets, punctuation, special characters, and .

1.1.4 Remove stopwords

Reviews also have a list of generic words which do not add useful information to the classifier. Such word acts as noise in the dataset making no contribution to the inference process. We remove them using nltk package. Example of such words are: 'the', 'i', 'you', 'a', 'which' etc.

1.1.5 Stemming

Sometimes sentences in a document have the same words repeated multiple times with prefixes or suffixes. This reduces the importance of the base word in the document. We use nltk Porter-Stemmer to slice the beginning or the end of the word to remove such affixes and reduce the word to its base form.

Sample Text after preprocessing:
==
'wonder littl product film techniqu unassum old time bbc fashion give comfort sometim discomfort sens realism entir piec actor extrem well chosen michael sheen got polari voic pat truli see seamless edit guid refer william diari entri well worth watch terrificli written perform piec master product one great master comedi life realism realli come home littl thing fantasi guard rather use tradit dream techniqu remain solid disappear play knowledg sens particularli scene concern orton halliwel set particularli flat halliwel mural decor everi surfac terribl well done'

1.2 Preliminary analysis

1.2.1 Data Split

The IMDB dataset was split at 70% training samples and 30% test samples. Both Probabilistic and Neural Network model were trained on the training set and evaluated on the test set. The label distribution is given in figure 1.

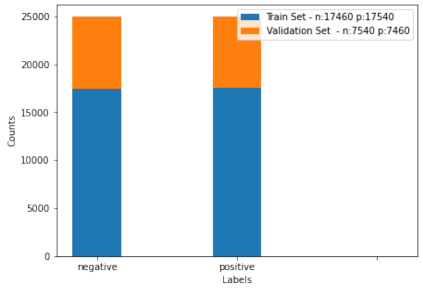
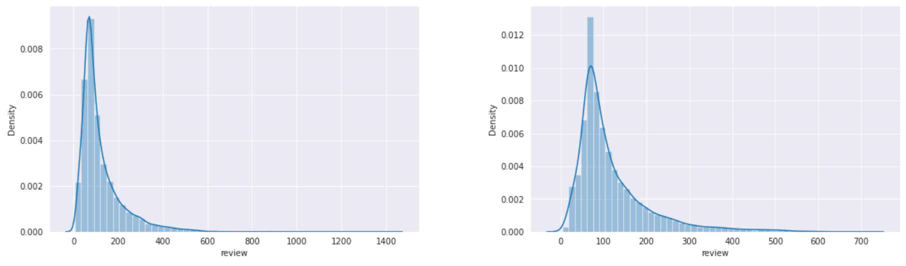


Figure 1: Data distribution among train set and test set.

1.2.2 Review lengths

Positive and Negative sentiments have almost the same review lengths with positive reviews being slightly longer than negative reviews. The average length of positive reviews is 121.65 as compared to the average length of 118.28 words in negative reviews (figure 2).



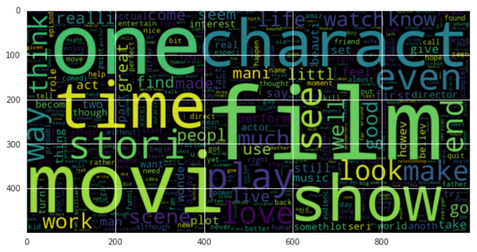
(a) Positive reviews (b) Negative reviews

Figure 2: Review length distribution.

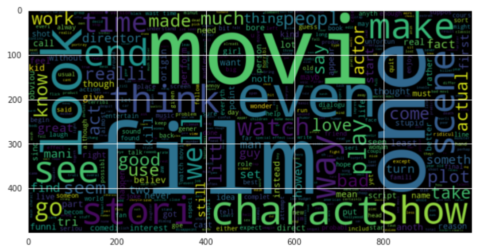
1.2.3 Word cloud

The Wordclouds for positive and negative reviews have almost similar most frequent words in the documents. These are generic words which may not add much information to the model eg 'movie', 'film', 'story', 'show' etc. On the other hand on closer inspection we can see words with definite positive meaning in English language such as 'love', 'good', 'interest', 'better' etc., which appears in the Positive review. Words with negative connotation such as 'least', 'old', 'terrible',

’never’ etc., mostly appear in negative reviews. Some of the positive and negative words seem to appear in the opposite label as well (figure 3).



(a) Positive reviews



(b) Negative reviews

Figure 3: Word clouds.

2 PART 2: Generative Model

2.1 Probabilistic (generative) Language Model

In order to generate text, we used a probabilistic model based on word frequencies. We went through the training dataset to train our language model. For each word, we calculate the probability the word occurs in a positive review and the probability it shows up in a negative review. We normalize the probability of occurrence across all words in the reviews. When sampling for a new positive review, we then use these normalized probabilities to sample across the whole word pool. This allows for occurrence of some words that don’t typically occur in positive reviews still being sampled. To generate the length of each review, we also sample from the distribution of lengths of reviews in the training set.

2.2 TFIDF Naive Bayes Classifier

We decided to use a Naive Bayes Classifier for the probabilistic approach. The Naive Bayes classifier builds a model using TF-IDF probabilities. Basically, using normalized term frequencies, we decide whether a review is positive or negative sentiment by taking the product of probabilities for each word in the review and taking the max of the two probabilities.

3 PART 3: Discriminative Neural Network Model

3.1 Long Short Term Memory Classifier (LSTM)

We use a Long Short Term Memory, a Recurrent Neural Network model to train sequences of word and predict if it is a positive review or a negative review.

Encoding Text

Like Naive Bayes, LSTM networks also cannot process raw text for sentiment prediction. We will encode the text using the word ranks based on it’s frequency. To do this we generate a word vocabulary dictionary whose values are the indices of the word list sorted in the order of term frequency. We use this mapping to map words in the document to integer values based on the dictionary. This creates a list of integer values all of which are sorted in the vocab dictionary. We additionally pad and truncate the encoded dataset to make the dataset size uniform. Padding is done by adding 0s when the encoding length is less than a certain $seqlen$, and truncating is used to cut the text if the encoding length is larger than $seqlen$. We repeat the same process for the test set and generate the input dataset for LSTM classifier.

LSTM Network Architecture

As stated we are using a simple Recurrent Neural Network i.e the LSTM network to predict review sentiments. The network will have three types of modules: Embedding - to map each token (word) to embedding vector, LSTM which takes the token embedding as input and uses an RNN model to process the sequence of data and a fully connected layer to convert the processed data back to label probability. The model takes in review in the form of encoding as input and predicts the probability of the review being positive or negative sentiment as output. The network architecture is given figure [lstm figure].

Dataloader and Training process

We convert the train and test set to torch tensors using pytorch's dataloader module and generate their respective iterators with a batch size of 50. Since the labels are binary, we use a Binary Cross Entropy loss as the loss function and ADAM optimizer with learning rate 0.001 as the gradient descent to perform each training step. The LSTM model is then trained Using standard Deep Learning training loop provided by PyTorch's implementation. Each epoch of training is followed by testing on the validation dataset where the train loss, test loss, train accuracy and test accuracy are saved along with the weights of the model with the best validation loss. Additionally weights are clipped to avoid exploding gradient problem in LSTMs

The LSTM network is trained and tested on both synthetic data and real world data, the results of which are shown in next section.

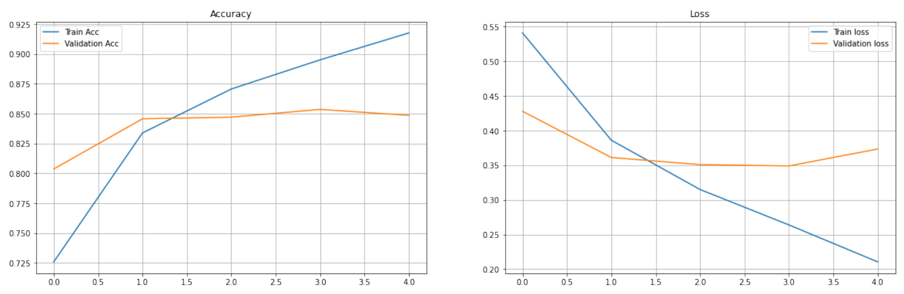


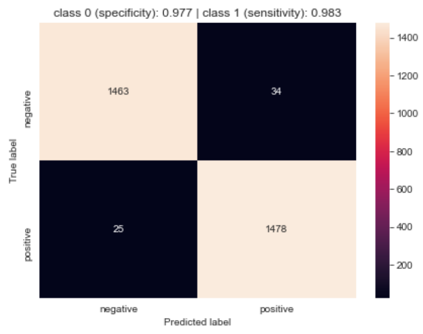
Figure 4: Accuracy (left) and Loss (right) plots for LSTM trained for 5 epochs.

4 STEP 4: Synthetic Data Results

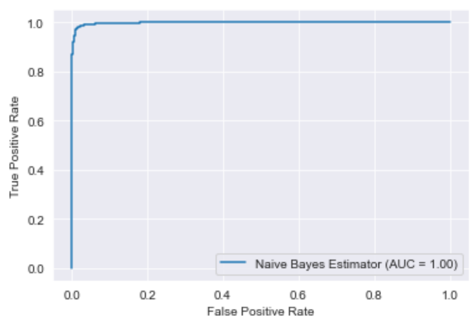
4.1 Synthetic Validation Data and Training Data

4.1.1 Model Inference using probabilistic model

Naive Bayes model accuracy: 98.37%



(a) Confusion Matrix

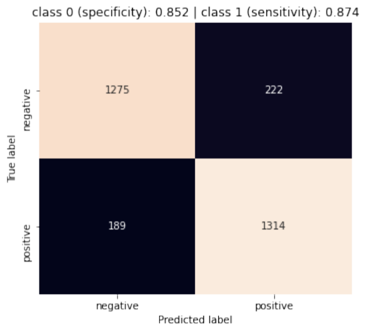


(b) Receiver Operator Curve

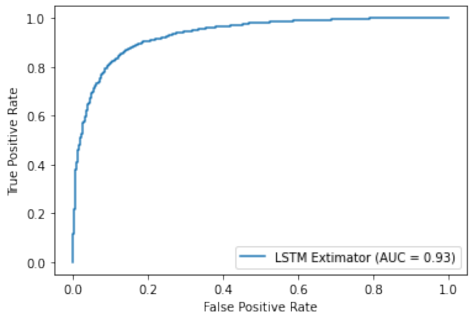
Figure 5: Naive Bayes model evaluation on synthetic dataset.

4.1.2 Model evaluation using discriminative neural network

LSTM model accuracy: 86.30%



(a) Confusion Matrix



(b) Receiver Operator Curve

Figure 6: LSTM network evaluation on synthetic dataset.

4.1.3 Synthetic Data Analysis

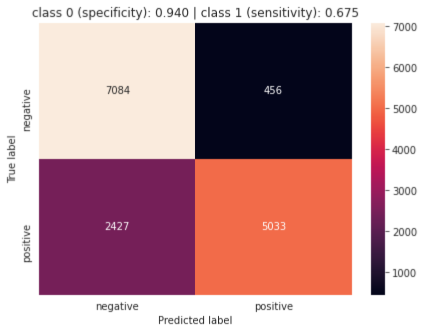
From these results we can see that both models perform very well. The Naive Bayes model has an incredibly high accuracy of 98% due to the fact that this model was used to generate the synthetic data. As such, it obviously is able to fit the generated data very well. This performance level can be seen as over fitting as real world classification should be expected to be much lower.

4.2 Real World Data Analysis (Trained On Synthetic Data)

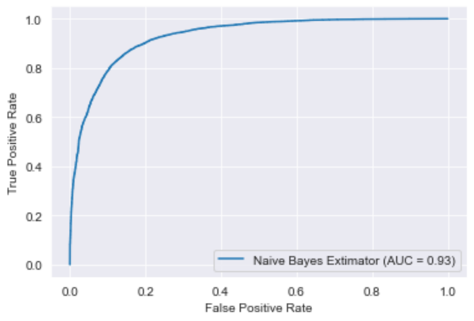
We also wanted to explore the performance of a model trained on synthetic data when applied to real data.

4.2.1 Model Inference using probabilistic model

Naive Bayes model accuracy: 80.78%



(a) Confusion Matrix

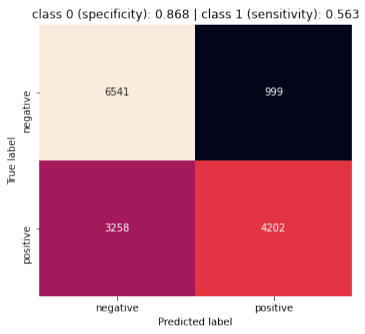


(b) Receiver Operator Curve

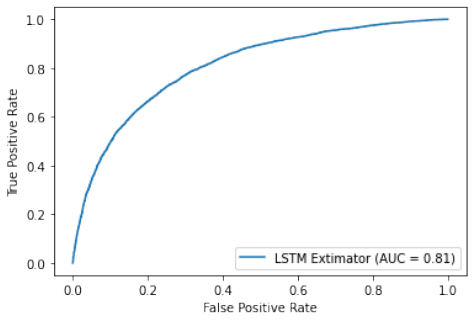
Figure 7: Naive Bayes model (trained on synthetic dataset) evaluation on synthetic dataset.

4.2.2 Model evaluation using discriminative neural network

LSTM model accuracy: 71.62%



(a) Confusion Matrix



(b) Receiver Operator Curve

Figure 8: LSTM network (trained on synthetic dataset) evaluation on real world dataset.

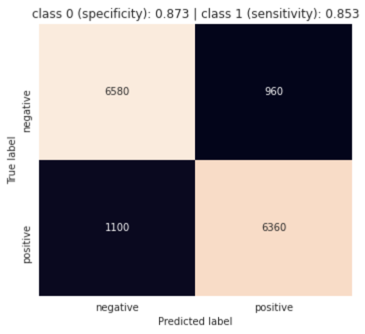
4.2.3 Real World Testing Data/Synthetic Training Data Analysis

From these results we can see that the Naive Bayes model was able to perform better on the real world data given the models were trained on synthetic data. This makes sense as the synthetic training data was generated by the same underlying probabilistic principles as the Naive Bayes model. Both of these models perform relatively well given the fact they were trained with artificially generated training data. This implies that the generated training data is a good representation of real reviews and is of high quality in terms of capturing the important information. If these models were not able to perform well, we would begin to question the effectiveness of the generated model. However the model performances are sufficient and the quality of the generated data is determined to be adequate.

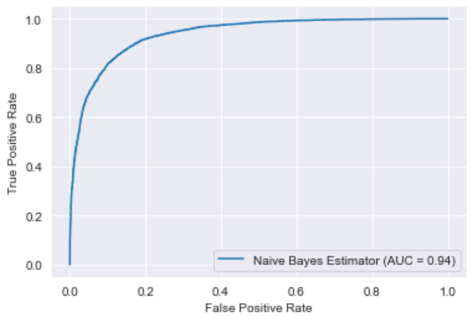
5 STEP 5: Real Data Results

5.1 Naive Bayes Classifier on real dataset

Naive Bayes model accuracy: 86.27%



(a) Confusion Matrix

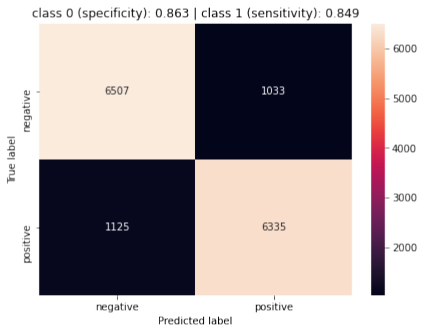


(b) Receiver Operator Curve

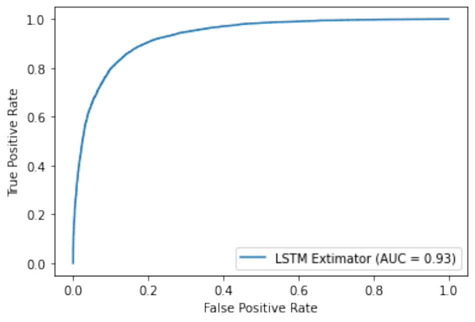
Figure 9: Naive Bayes model evaluation on real world dataset.

5.2 LSTM on real dataset

LSTM model accuracy: 85.61%



(a) Confusion Matrix



(b) Receiver Operator Curve

Figure 10: LSTM network evaluation on real world dataset.

5.3 Real World Data Analysis

As we can see from these results, both of the classifiers do a very good job of classification. Given that most humans have trouble agreeing on the sentiment, the classifier performance being 80+% is exceptional. One interesting note is that the naive bayes classifier was able to correctly classify negative reviews more accurately than positive reviews. This might be due to the fact the occurrence of some words in negative reviews is very indicative of the review being negative. It could also be that sentiment is actually a spectrum and the negative reviews present in the dataset happen to be further from the middle than the positive reviews making them easier for the naive bayes model to classify. The LSTM performed in a more balanced way for both positive and negative classes. This may have been due to the reward function assigned at during training resulting in balanced weight updates that kept the value of these predictions equal. While there was less bias in the LSTM, some bias was still present furthering the notion that this bias stems from the underlying data.

6 STEP 6: Discussion

As mentioned earlier, the results are good in terms of both the model performance and quality of the synthetic dataset. Both models, LSTM and Naive Bayes Classifier, were able to learn the underlying model in all of the situations we presented. From our analysis, we uncovered a slight bias in the dataset. Namely that the negative example are "more" negative that the positive examples are positive. This resulted in more accurate detection of negative examples in both classifiers. This brings to light an interesting point that some binary classification problems are actually linear problems fit to a binary classification, or in other words, the data is actually coming from a spectrum between the two ends which are positive and negative examples. Real reviews can be strictly positive or negative or lie somewhere in the middle. The relaxation of this problem to binary classification loses some of the meaning captured in the text and trades it for representational simplicity. Both of the methods required a significant amount of data preprocessing to remove unnecessary characters, simplify word endings, and to tokenize the text. The naive bayes model took less than a second on a CPU to complete training on the real world dataset while the LSTM model took around 2 minutes on a GPU.

	Training time in seconds	
	Real Dataset (40k examples)	Synthetic Dataset (7k examples)
Naive Bayes	0.032	0.007
LSTM	46.467	9.412

The LSTM model has significantly higher training costs without a substantial improvement in accuracy (actually resulting in slight accuracy degradation). In addition, the LSTM is less interpretable than the Naive Bayes classifier as the inner working of a neural network are impossible to assign meaning too. This is in contrast to the Naive Bayes model where we know exactly what the underlying probabilities mean and can draw an intuitive analysis and understanding for how the model works. Overall, we determined the naive bayes model to be the better solution for this problem as it is far easier to train, has higher accuracy on both the real and synthetic dataset, and is more interpretable.

REFERENCES

Maas, Andrew, et al. "Learning word vectors for sentiment analysis." Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies. 2011.