CSCE-771 Project Report: Sentiment Analysis using Large Movie Review dataset

Fawad Kirmani

November 2020

1 Introduction

The goal of this work is to learn and explore different sentiment analysis algorithms. The data used in this work is large movie review dataset prepared by Andrew Mass et al. [1]. For this project, different word vectorization/embedding models are implemented to tokenize the IMDB review dataset. Multiple traditional machine learning models (like Random Forest, Logistic Regression and XGBoost) and deep learning (like CNN and LSTM) classification models have been explored to classify movie reviews into positive and negative classes. Pretrained language models based on transformers (like BERT and DistilBERT) are implemented to learn and achieve better classification scores. State of the art transformer based language model XLNet has also been implemented for this project. The main aim of this project is to learn traditional as well as state of the art methods for sentiment analysis.

In the next section titled (2) *Data*, there is discussion of the data used in this project in more detail. In the section titled (3) *Original work on the dataset*, there is brief discussion of the paper by Andrew Mass et al. [1], which prepared and used this dataset in the paper. In the the section titled (4) *Different approaches for sentiment analysis*, there is a discussion in detail about the implementation and results of methods and algorithms that are explored for sentiment analysis in this project. The work in this project has been divided into three tracks in section (4):

1. Machine Learning models (like Random Forest, Logistic Regression and XGBoost) with TF-IDF vectorization () for classification.

2. Deep Learning models (like CNN and LSTM) with GloVe Word-Embeddings for classification.

3. Classification with Transformers based Pretrained models like BERT, DistilBERT and XLNet.

2 Dataset

The data this work comes from Andrew Mass et al. [1] large movie reviews dataset. The dataset has 50,000 reviews from IMDB, allowing no more than 30 reviews per movie. The dataset has even number number of positive and negative movie reviews. The dataset contains highly polar movie reviews data. A negative review has a score ≤ 4 out of 10, and a positive review has a score ≥ 7 out of 10. The dataset is evenly divided into training and test sets with 25,000 reviews each. Both the training and test have 12,500 positive and negative reviews. This dataset is widely used to benchmark new work.

For this project, all the positive and negative reviews from both training and test datasets are combined in to one dataset. So, there are total 50,000 movie reviews in the dataset. To explore different algorithms for sentiment analysis in this project, positive reviews are labeled as 1 and negative reviews are labeled as 0.

3 Original work on the dataset

In the original work, Andrew Mass et al. [1] presented a model that uses a mix of unsupervised and supervised techniques to learn word vectors capturing semantic term–document information as well as rich sentiment content. Their proposed model leveraged both continuous and multi-dimensional sentiment information as well as non-sentiment annotations. They incorporated the model to utilize the document-level sentiment polarity annotations present in many online documents (e.g. star ratings).

Andrew Mass et al. [1] tested their model on large movie review dataset from IMDB. The model performed best when concatenated with bag of words representation and additional unlabeled dataset. The full model with bag of words and additional unlabeled dataset achieved accuracy of around 89%.

The variant of their model which utilized extra unlabeled data during training performed best. They provided the large IMDB movie review dataset publicly to serve as robust benchmark for future work. This project is concentrating on the sentiment analysis part of the Andrew Mass et al. [1] paper where this data from is used.

4 Different approaches for sentiment analysis

4.1 Machine Learning models with TF-IDF vecotization

In the first approach to classify movie reviews, words are vectorized with TF-IDF. Then these vectorized words are used as input to traditional machine learning classification moodels like Logistic Regression, Random Forest and XGBoost.

TF-IDF stands for term frequency–inverse document frequency. The term frequency (TF) is raw count of a term in a document. The inverse document frequency (IDF) is a measure of how much information the word provides, i.e., if it's common or rare across all documents. TF-IDF is the product of two frequencies TF and IDF. Given N documents. For term (word) t in document d

$$Termfrequency(TF) = tf(t, d)$$

$$InversedocumentfrequencyIDF(t) = log[N/DF(t)] + 1$$

$$TF - IDF(t, d) = TF(t, d) * IDF(t)$$

For TF-IDF transformer, ngrams of 1 to 5 are considered. After fitting ngrams count to tf-idf transformers, total 28,918,046 features are created. The dataset is divided into train and test set of 75% to 25% ratio respectively. All three models i.e., Logistic Regression, Random Forest and XGBoost are trained with GridSearchCV to find best hyperparameters for each model. Table (4.1) shows the precision, recall, f1-score and accuracy on the test dataset with all three machine learning models. Logistic regression model was overtrained, it is giving accuracy of 90% on training data but only 83% accuracy on test dataset. So, the results of Logistic Regression model are not considered for final comments in this section. The best results are achieved with Random Forest model in this section of the project, where the model achieved precision, recall and f1-score 84% (macro average). Random Forest model achieved accuracy of 84% as well which is best for this section of this project. The results with XGBoost model were not as good as Random Forest, it achieved precision, recall and f1-score of 82% only. XGBoost model achieved accuracy of 82%.

Figures (1) show the normailzed confusion matrixes for Logistic Regression, Random Forest and XGBoost classification models. Random Forest model gave the least false negative rate (FNR) and false positive rate (FPR) in this section of the project.

ML Model	**Precision**	**Recall**	**F1-score**	**Accuracy**
*Logistic Regression**	83%	83%	83%	83%
Random Forest	84%	84%	84%	84%
XGBoost	82%	82%	82%	82%

Table 1: Machine Learning models results with TF-IDF vectorization of IMDB dataset. These are the macro averages of Precision, Recall and F1-socre. Logistic Regression model is overtrained, giving training accuracy of 90%.

4.2 Deep Learning with GloVe Word Embeddings

In the second method classify movie reviews, GloVe [2] word embedding is used. Then these tokenized word embeddings are used for Deep Learning classification models like Convolutional Neural Network (CNN), Long Short Term Memory (LSTM), Autoencoder LSTM, and Bidirectional LSTM.

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

The features from GloVe embeddings are trained with different deep learning models. Training and test datasets are divided in 90% to 10% ratio. These deep learning models are implemented using Keras [9] library avialble for Python. Early stopping has been implemented to check if consecutive epochs are reducing validation set loss. If the loss is not decreasing or it is getting worse for few epoch, model training is stopped to prevent overfitting. The deep learning models implemented for this work are -

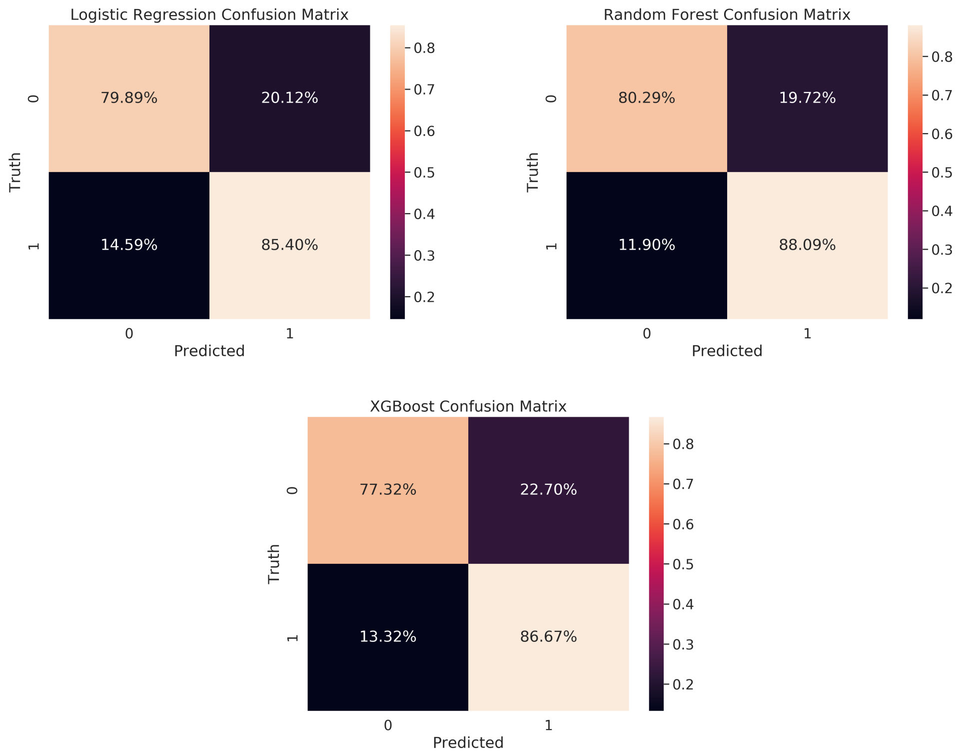

Figure 1: Normalized Confusion matrix for Logistic Regression, Random Forest and XGBoost classification models on TF-IDF vectorized lagre movie review dataset.

1. CNN: Figure (2) shows the CNN model summary. The CNN model has one embedding layer. There are two Convolutional layers and four Dense layers in the CNN model. There are 319,601 trainable parameters for CNN model.

2. LSTM: Figure (3) shows the LSTM model summary. The LSTM model has one embedding layer. The LSTM model has two LSTM layers and one output layer. There are 5,305 trainable parameters for LSTM model.

3. Autoencoder LSTM: Figure (4) shows the Autoencoder LSTM model summary. The Autoencoder LSTM model has one embedding layer. The Autoencoder LSTM model has one LSTM layers for encoding and

one LSTM layer for decoding. There are 43,583 trainable parameters for Autoencoder LSTM model.

4. Bidirectional LSTM: Figure (5) shows the Bidirectional LSTM model summary. The Bidirectional LSTM model has one embedding layer. The Bidirectional LSTM model has two Bidirectional LSTM layers. There are 9,941 trainable parameters for Bidirectional LSTM model.

CNN gave the best overall results in this section of this project. CNN achieved precision, recall and f1-score 90% (macro average). The accuracy of the CNN was model also the best i.e., 90%. But LSTM model gave the least false negative rate i.e., 7.88%. The least false positive rate was achieved by Bidirectional LSTM of 11.25%. The three models i.e., LSTM, Autoencoder LSTM and Bidirectional LSTM performed similarly in terms of precision, recall, f1-score and accuracy. The detailed results are presented the Table (4.2).

DL Model	**Precision**	**Recall**	**F1-score**	**Accuracy**
CNN	90%	90%	90%	90%
LSTM	89%	89%	89%	89%
Autoencode LSTM	89%	89%	89%	89%
Bi-directional LSTM	89%	89%	89%	89%

Table 2: Deep Learning models results with GloVe word embeddings of IMDB dataset. These are the macro averages of Precision, Recall and F1-score.

Figure (6) shows the CNN model accuracy vs epoch, CNN model loss vs epoch, and CNN normalized confusion matrix graphs. Figure (7) shows the LSTM model accuracy vs epoch, LSTM model loss vs epoch, and LSTM normalized confusion matrix graphs. Figure (8) shows the Autoencoder LSTM model accuracy vs epoch, Autoencoder LSTM model loss vs epoch, and Autoencoder LSTM normalized confusion matrix graphs.

4.3 Classification with Transformers based Pretrained models

Transformers [3] provides thousands of pretrained models to perform tasks on texts such as classification, information extraction, question answering,

```
                        CNN Model Summary
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 300, 300)          29147100
_________________________________________________________________
conv1d (Conv1D)              (None, 298, 16)           14416
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 149, 16)           0
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 149, 16)           0
_________________________________________________________________
dropout (Dropout)            (None, 149, 16)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 147, 32)           1568
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 73, 32)            0
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 73, 32)            0
_________________________________________________________________
dropout_1 (Dropout)          (None, 73, 32)            0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2336)              0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               299136
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
batch_normalization (BatchNo (None, 128)               512
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 32)                4128
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
batch_normalization_1 (Batch (None, 32)                128
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 32)                0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 33
=================================================================
Total params: 29,467,021
Trainable params: 319,601
Non-trainable params: 29,147,420
_________________________________________________________________
None
```

Figure 2: CNN model summary

summarizing, translation, text generation, etc in 100+ languages. Its aim is to make cutting-edge NLP easier to use for everyone.

LSTM Model Summary

Model: "sequential_2"

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 300, 300)	29147100
lstm_2 (LSTM)	(None, 300, 4)	4880
lstm_3 (LSTM)	(None, 8)	416
dense_5 (Dense)	(None, 1)	9

Total params: 29,152,405
Trainable params: 5,305
Non-trainable params: 29,147,100

None

Figure 3: LSTM model summary

Autoencoder LSTM Model Summary

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 300, 300)	29147100
lstm (LSTM)	(None, 300, 4)	4880
flatten (Flatten)	(None, 1200)	0
repeat_vector (RepeatVector)	(None, 5, 1200)	0
lstm_1 (LSTM)	(None, 5, 8)	38688
time_distributed (TimeDistri	(None, 5, 1)	9
flatten_1 (Flatten)	(None, 5)	0
dense_1 (Dense)	(None, 1)	6

Total params: 29,190,683
Trainable params: 43,583
Non-trainable params: 29,147,100

None

Figure 4: Autoencoder LSTM model summary

This project explored three transformers based pretrained language models for classification of IMDB large movie reviews data:

1. BERT: BERT [5] stands for Bidirectional Encoder Representations from Transformers. BERT [5] is a pre-trained deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications [5].

```
                    Bidirectional LSTM Model Summary
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 300, 300)          29147100
_________________________________________________________________
bidirectional (Bidirectional (None, 300, 8)            9760
_________________________________________________________________
bidirectional_1 (Bidirection (None, 4)                 176
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 5
=================================================================
Total params: 29,157,041
Trainable params: 9,941
Non-trainable params: 29,147,100
_________________________________________________________________
None
```

Figure 5: Bidirectional LSTM model summary

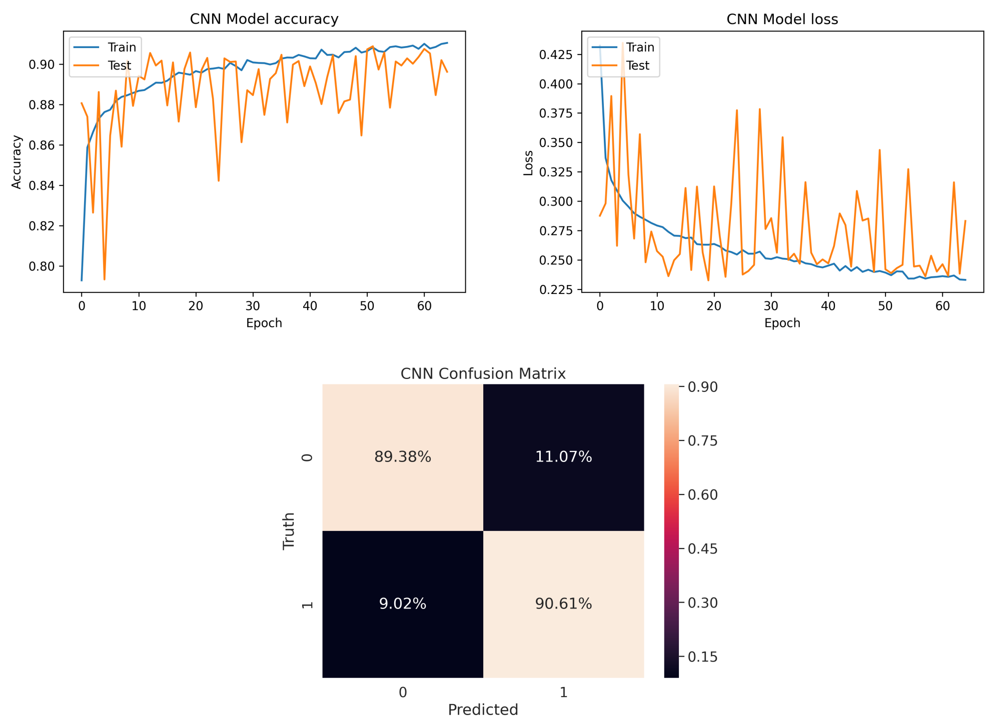

Figure 6: CNN model accuracy vs epoch, CNN model loss vs epoch, and CNN normalized confusion matrix graphs with GloVe word embeddings

2. DistilBERT: DistilBERT [6] is a distilled version of BERT. It is smaller, faster and lighter than BERT. It reduces size of a BERT model by 40%, while retaining 97% of its language understanding capabilities

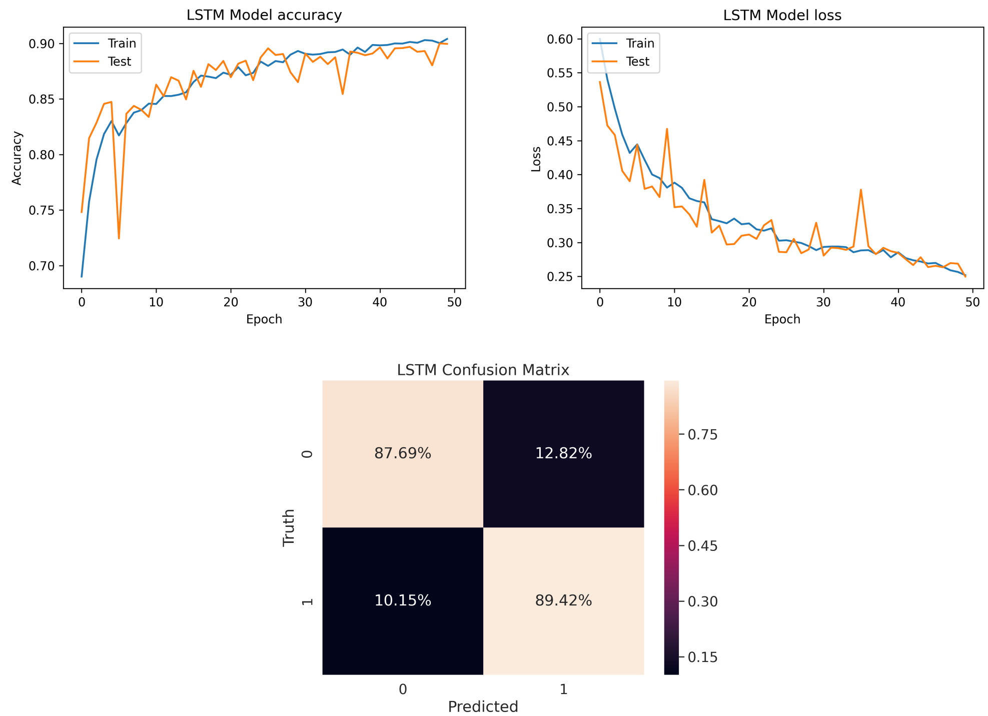

Figure 7: LSTM model accuracy vs epoch, LSTM model loss vs epoch, and LSTM normalized confusion matrix graphs with GloVe word embeddings

and being 60% faster [6].

3. XLM-RoBETAa: XLM-RoBETAa [1] a transformer-based multilingual masked language model pre-trained on text in 100 languages, which obtains state-of-the-art performance on cross-lingual classification, sequence labeling and question answering. XLM-RoBERTa (XLM-R) outperforms mBERT on cross-lingual classification by up to 23% accuracy on low-resource languages [10].

4. XLNet: XLNet [7] is a generalized autoregressive pretraining method that (1) enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order and (2) overcomes the limitations of BERT thanks to its autoregressive formulation [7].

This project used ktrain [8] library to train transformers based pretrained

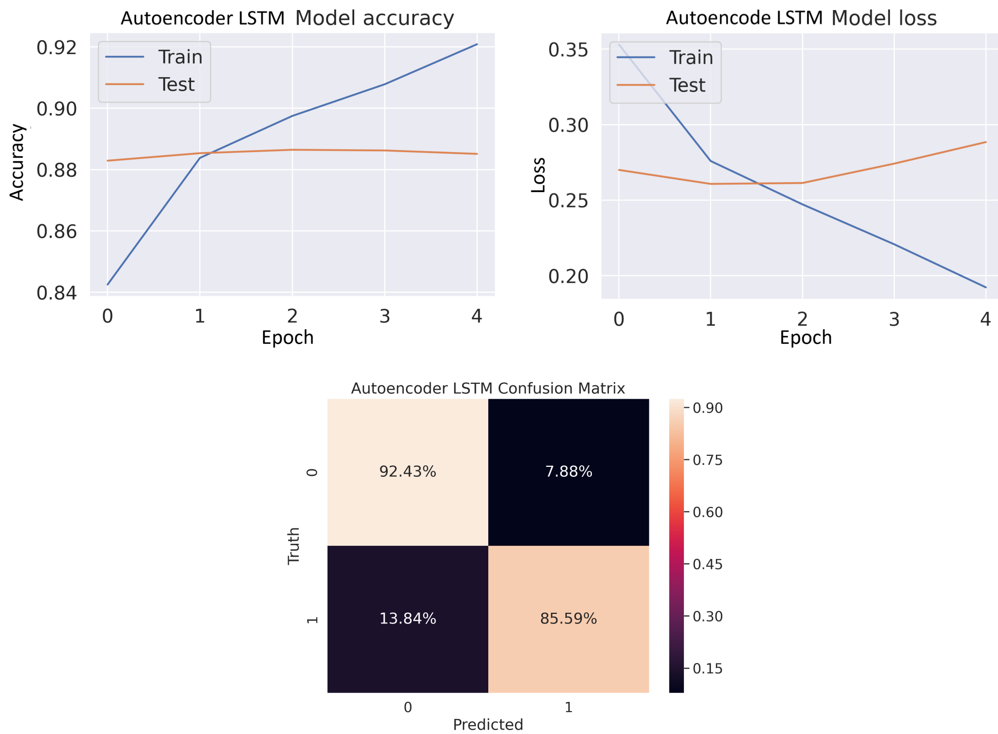

Figure 8: Autoencoder LSTM model accuracy vs epoch, Autoencoder LSTM model loss vs epoch, and Autoencoder LSTM normalized confusion matrix graphs with GloVe word embeddings

models. ktrain is low-code library for augmented machine learning. It is a wrapper to TensorFlow and many other libraries (e.g., transformers, scikit-learn, stellargraph). It is one of the latest state of the art library to train machine learning models.

To fine tune BERT, DistilBERT, XLM and XLNet classifiers, learning rate is tuned for each of them using in-built learning rate finder lr_find() function in ktrain [8]. Figure (10) shows the graph generated for loss vs learning_rate (log scale) to find best learning rate for BERT model. BERT [5] paper uses learning rate of 5e-5, 4e-5, 3e-5 and 2e-5. I choose learning rate of 2e-5 as in the figure (10), we can see the loss is going down at learning rate of 2e-5. BERT model is trained with max length of 256 and 3 epochs. BERT model achieved precision, recall, f1-score and accuracy scores of 93%.

To explore more on learning late and lr_find(), DistilBERT model was

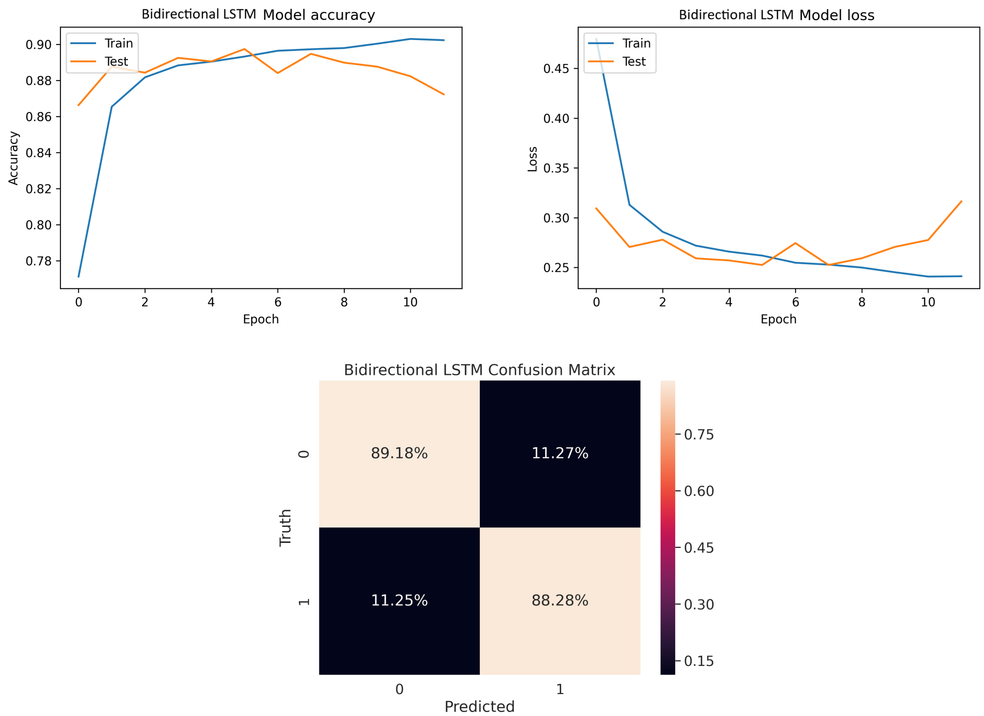

Figure 9: Bidirectional LSTM model accuracy vs epoch, Bidirectional LSTM model loss vs epoch, and Bidirectional LSTM normalized confusion matrix graphs with GloVe word embeddings

first trained with learning rate of 2e-4 and 4 epochs after observing the plot in figure (10). DistilBERT, didn't achieved good results with for max length of 500 and learning late of 2e-4 and 4 epochs. The model got overtrained and achieved accuracy of 90% on test data while 97% on training data. Then, DistilBERT model was trained on max length of 500 with learning rate of 2e-5 and 3 epochs, the model achieved pretty good precision, recall and f1-score scores of 94%. DistilBERT also achieved accuracy score of 94%.

XLM-RoBERTa model was first trained with max length of 256 and learning rate of 2e-4 and 4 epochs XLM-RoBERTa model got overtrained with 4 epoch, on training data it gave accuracy of 96% but on testing data it gave accuracy of only 90%. Then, XLM-RoBERTa model was trained on max length of 512 with learning rate of 2e-5 and 2 epochs, the model achieved pretty good precision, recall and f1-score scores of 94%. XLM-RoBERTa also

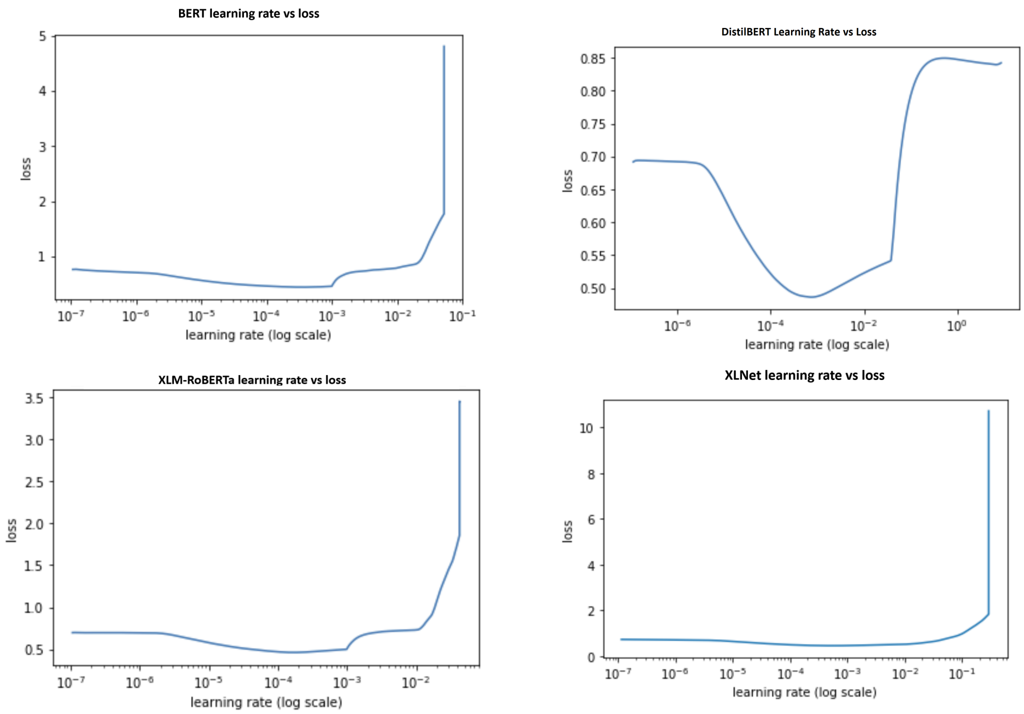

Figure 10: BERT, DistilBERT, XLM-Roberta and XLNet learning rate vs loss graphs with 1, 2 and 1 epochs respectively.

achieved accuracy score of 94%.

For XLNet, it was taking very long time to explore training on different learning rates, so to complete project on time, XLNet model with max length of 512 was trained with learning rate of 2e-5 with 3 epochs similar to DistilBERT. The best results are achieve with XLNet. XLNet model achieved precision, recall, f1-score and accuracy scores of 95%. Table (4.3) shows the scores achieved by all the three (BERT, DistilBERT and XLNet) pretrained transformers based models implemented for this project.

Table (4.3) shows the results for BERT, DistilBERT, XLM-RoBERTa and XLNet. Classification with BERT achieved precision, recall, f1-score and accuracy of 93%. Classification with DistilBERT achieved precision, recall, f1-score and accuracy of 94%. Classification with XLM-RoBERTa achieved precision, recall, f1-score and accuracy of 94%. XLNet achieved the best classification scores. XLNet achieved precision, recall, f1-score and accuracy of 95%. XLNet model achieves lowest FPR of 4.59% and lowest

FNR of 5.19% among all the models implemented for this project.

Transformers	**Precision**	**Recall**	**F1-score**	**Accuracy**
Bert	93%	93%	93%	93%
Distilbert	94%	94%	94%	94%
XLM-Roberta	94%	94%	94%	94%
XLNet	95%	95%	95%	95%

Table 3: Machine Learning models results with TF-IDF vectorization of IMDB large movie reviews dataset. These are the macro averages of Precision, Recall and F1-score.

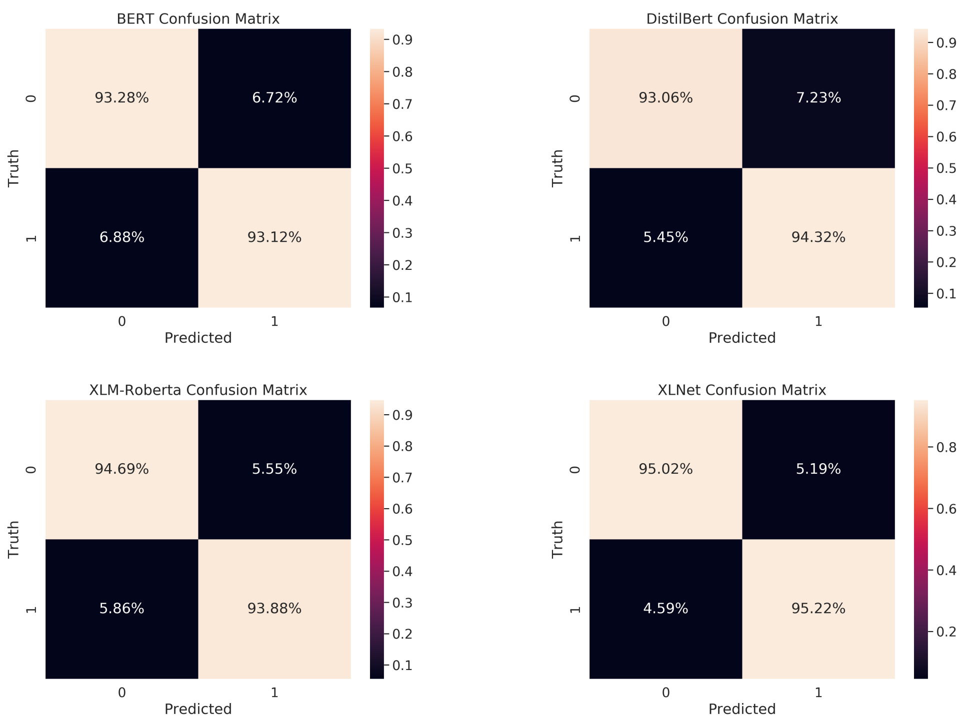

Figure 11: BERT, DistilBERT, XLM-Roberta and XLNet normalized confusion matrix on IMDB large movie reviews dataset

5 Evaluation

The main goal of the project was to learn and implement different natural language processing techniques (NLP) for sentiment analysis. The project mostly accomplishes it's goal. The only think that could have been better if the project could further explore another word embedding method and learn more about hyperparameter tuning of pretrained language models. But the given the time allotted for the course, almost all the objectives of the project are accomplished.

The other goal of the project was to measure the performance of the results of the sentiment classification model implemented in the project to Andrew Maas et al. [1]. In that sense -

1. The machine learning models trained on TF-IDF vectorized dataset didn't perform well at all. The best model among them i.e., Random Forest model gave accuracy of only 84%. It was lower than 5% points than Andrew Maas et al. [1] accuracy score of 89%.

2. The deep learning models trained on GloVe word embeddings datasets did perform as well. The best model among them i.e., CNN gave slightly better accuracy than Andrew Maas et al. [1] accuracy score of 89%. Other models i.e., LSTM, Autoencode LSTM and Bi-directional LSTM gave same accuracy score of 89% as Andrew Maas et al.

3. The transformers based pretrained models performed best. They outperformed Andrew Maas et al. [1] accuracy score by 5-6% points. XLNet model performed best and achieved accuracy of 95% compared to Andrew Maas et al. score of 89%. Other transformer based pretrained language models also performed better than Andrew Maas et al.

The transformer based language based models can be further improved by tuning hyperparameters with more GPU resources or time or both. For example, XLNet [7] model on the same movie reviews data achieved 96% accuracy. So, with little more hyperparameter tuning similar performance can be achieved.

6 Conclusion

In this project, different methods and algorithms were implemented for sentiment analysis on large movie reviews dataset [1]. From TF-IDF tokenizer, GloVe word embeddings to state of the art pretrained language model like XLNet. The best accuracy score of 95% is achieved using state of the algorithm XLNet. The main learning's from the project are to learn text tokenizing, word embeddings and transformers baser pretrained models. The main challenge was to tune hyperparatemers for transformers. To tune one hyperparameter, it used to take lot of hours. So, only parameter which was explored somewhat was learning rate for different transformers based models. The performance of transformers based pretrained language based models could have been further improved with some hyperparameter tuning as shown in Sun et al. [11]. That is one of the future tasks to explore.

This project shows how the text based sentiment analysis has come a long way during last few years through these algorithms. The project shows that transformer based pretrained language models are performing significantly better than traditional machine learning and deep learning models.

References

[1] Maas, Andrew & Daly, Raymond & Pham, Peter & Huang, Dan & Ng, Andrew & Potts, Christopher. (2011). Learning Word Vectors for Sentiment Analysis. 142-150.

[2] Pennington, Jeffrey & Socher, Richard & Manning, Christoper. (2014). GloVe: Global Vectors for Word Representation. EMNLP. 14. 1532-1543. 10.3115/v1/D14-1162.

[3] Vaswani, Ashish & Shazeer, Noam & Parmar, Niki & Uszkoreit, Jakob & Jones, Llion & Gomez, Aidan & Kaiser, Lukasz & Polosukhin, Illia. (2017). Attention Is All You Need.

[4] F. Pedregosa, et al. (2011) Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 12 (November 2011), 2825-2830.

[5] Devlin, Jacob & Chang, Ming-Wei & Lee, Kenton & Toutanova, Kristina. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[6] Sanh, Victor & Debut, Lysandre & Chaumond, Julien & Wolf, Thomas. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.

[7] Yang, Zhilin & Dai, Zihang & Yang, Yiming & Carbonell, Jaime & Salakhutdinov, Ruslan & Le, Quoc. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding.

[8] Maiya, Arun. (2020). ktrain: A Low-Code Library for Augmented Machine Learning.

[9] Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

[10] Conneau, Alexis & Khandelwal, Kartikay & Goyal, Naman & Chaudhary, Vishrav & Wenzek, Guillaume & Guzman, Francisco & Grave, Edouard & Ott, Myle & Zettlemoyer, Luke & Stoyanov, Veselin. (2020). Unsupervised Cross-lingual Representation Learning at Scale. 8440-8451. 10.18653/v1/2020.acl-main.747.

[11] Chi, Sun & Qiu, Xipeng & Xu, Yige & Huang, Xuanjing. (2019). How to Fine-Tune BERT for Text Classification?.