Sentiment Analysis on IMDB Movie Reviews

Team 13
Stelly Tomy 1001871041
Sriya Mishra 1001879636
Sujeeth Nilakantan 1001855274

Abstract –

The goal of the paper is to assess the performance of machine learning and deep learning algorithms on various sentiment-labeled reviews from the IMDB Large Movie datasets. Preprocessing is the first and most important step in a Natural Language Processing (NLP), which plays a significant impact on the accuracy of algorithms. The positive and negative reviews are the basis on which classification will occur to which is further subjected to a variety of algorithms. The work is based on the IMDb dataset, which contains movie reviews as well as the labels that go with them (positive or negative). The conclusion is to find the model with the best accuracy.

Introduction –

Movie reviews from users can provide insight into the film and help determine whether it makes it a good watch or not. In order to get quick analysis on the movie reviews you need to make automatic classifications in a world full of movie reviews. Sentiment analysis is a branch of machine learning that focuses on extracting subject information from textual reviews using natural language processing and text mining. We can determine the viewers and decide the overall polarity of the review. With the help of movie reviews, it can quantitatively highlight on the favorability or likelihood of a film; and provide insight into the film.
This task purely relies on an analysis of human language, having a positive word in a review or a negative connotation. We will perform Sentiment classification on movie reviews using machine learning models and deep learning algorithms on the "Movie Review DB."

Dataset –

The movie review dataset is provided by Stanford.edu. The dataset has 50,000 movie reviews split evenly in training and testing datasets. It includes 50,000 unlabeled documents for unsupervised learning. We have set a negative review having a score <= 4 out of 10, and a positive review has a score >= 7 out of 10. This collection consists of no more than 30 reviews are allowed for any given movie because reviews for the same movie tend to have correlated ratings. We neglected neutral ratings in the train/test sets.
Folders and files are included in the Dataset are: test/, train/, imdb.vocab, imdbE.txt and Readme.txt

Project Description
1 Description –

Movie reviews are vital to gauge the performance of a movie. Users provide a numerical/star rating to a movie. Multiple reviews give a better understanding of the movie's success or failure and help in the recommendation process. Sentiment Analysis in machine learning aids to extract subjective information from the agglomerative textual reviews. The field of sentiment of analysis is closely tied to natural language processing and text mining. Determines the attitude of the reviewer and overall polarity of review. Using sentiment analysis, figure the state of mind of the reviewer whilst providing the review if the person was "happy", "sad", "angry". In this project we aim to use Sentiment Analysis on a set of movie reviews given by reviewers and try to understand their overall reaction to the movie. We first extract the seed words, define the polarity and text sentiment prediction.

2 Main references used for your project –

We took the reference from Apply Word Vectors for Sentiment Analysis of APP Reviews by Xian Fan, Xiaoge Li, Feihong Du, Xin Li and Mian Wei. The aim is to investigate the effectiveness of word vector representations for the problem of Sentiment Analysis. The paper aims to build sentiment lexicon and predict text sentiment. The three sub-tasks namely sentiment words extraction, polarity of sentiment words detection, and text sentiment prediction. The dataset from APP store is tokenized, then word vectors are used to build sentiment lexicon on App reviews domain by word2vec. Prominent sentiment seeds are extracted from comments. Further, they calculate the distance between all words containing sentiment seeds to forms them into groups following which they predict the polarity of sentiment words. In the end, they classify the sentiment of users' reviews using classical naive Bayes.

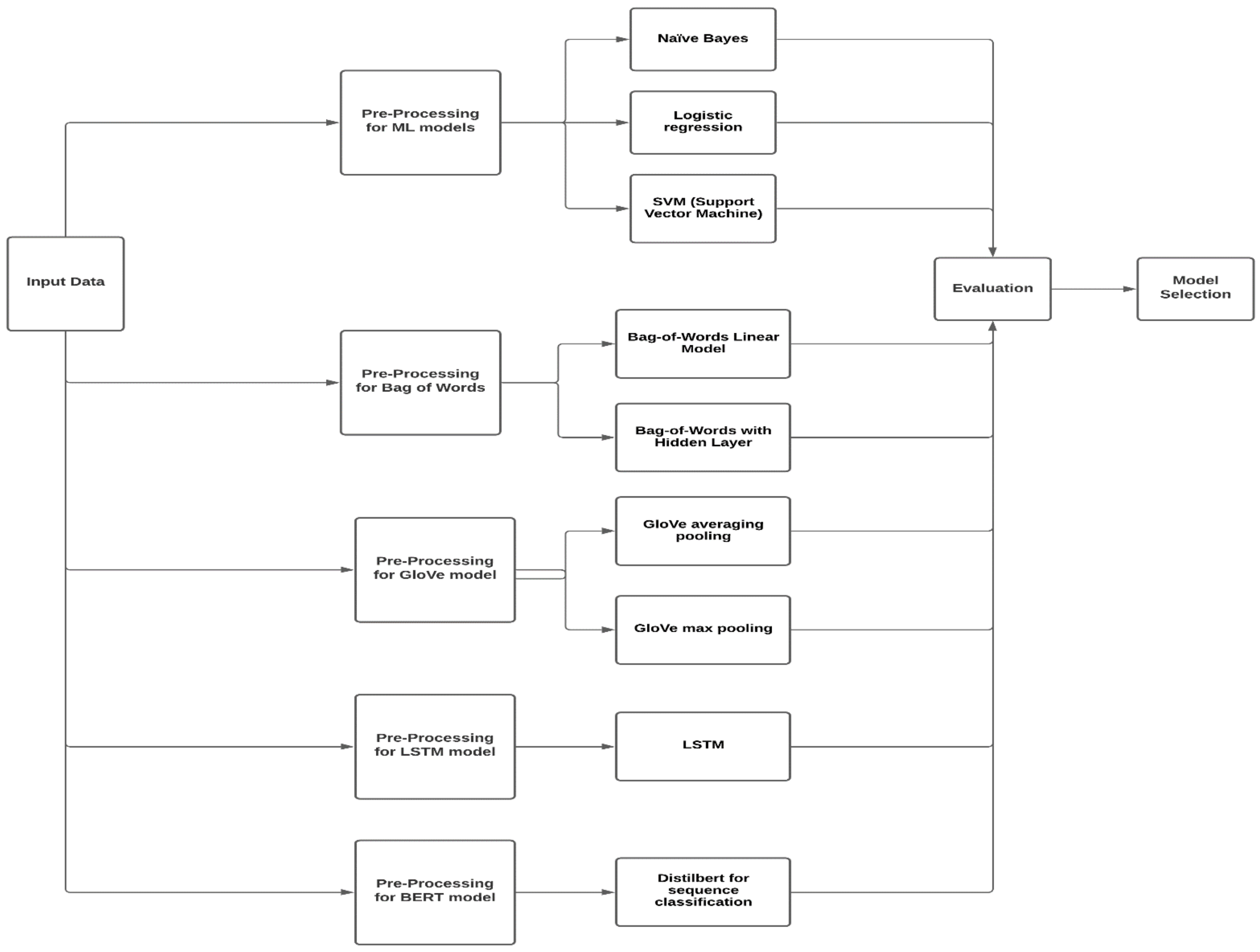

Input Data
Pre-Processing for ML models
Naïve Bayes
Logistic regression
SVM (Support Vector Machine)
Pre-Processing for Bag of Words
Bag-of-Words Linear Model
Bag-of-Words with Hidden Layer
Pre-Processing for GloVe model
GloVe averaging pooling
GloVe max pooling
Pre-Processing for LSTM model
LSTM
Pre-Processing for BERT model
Distilbert for sequence classification
Evaluation
Model Selection

Fig: Architecture

3 Difference in APPROACH/METHOD between your project and the main projects of your references –

We first pre-processed the data with the nltk library using regular expressions and lemmatization. Then we incorporated three different machine learning models and three deep learning models.

3.1 Pre-processing for text corpus:

For pre-processing we use the nltk to pre-process the text corpus. First all the characters of the corpus were converted to lower case. Secondly, we converted all the tabs to spaces. Third step was removing all punctuation, special characters. Further on we covert the corpus to tokens and eliminate stop words from the corpus. *Lemmatization* refers finding the stem words and deleting any extra additional words attached to the word to get the root word extracted. This process returns the base or dictionary form of a word, which is known as the *lemma*.

3.2 Machine Learning Models Used:

a) Naïve Bayes –

Naive Bayes classifier is a probabilistic machine learning model. It is used for classification task. The formula for the classifier is based on the Bayes theorem.

$$\mathrm{P(A|B)} = \frac{P(B|A).P(A)}{P(B)} .$$

The Multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification) and hence we used this in our implementation. The hyperparameters used were α (learning rate) = 1.0. The accuracy achieved is 83.75% for the test data.

b) Logistic regression –

The probability of a discrete outcome given an input variable uses logistic regression. The algorithm is used for classification purpose exclusively. This algorithm generates a binary outcome, that takes two values such as true/false, yes/no. Multinomial logistic regression is used with more than two discrete outcomes. Logistic regression wants to know if a new sample belongs to one of the classification categories.

$$\text{Logistic Function} = \frac{1}{1+ e^{x}}$$

The Logistic Regression model got us a validation accuracy of almost 85.16%.

c) SVM (Support Vector Machine) –

The support vector machine algorithm is to find a hyperplane in an N-dimensional space to define classifies the data points. There are numerous hyperplanes that could be used to separate the two classes of data points. Our goal is to find a plane with the strongest margin, i.e., the maximum distance between data points from both classes. Maximizing the margin distance provides allows future data points to be classified with greater certainty. The loss function we used to maximize to increase the margin is hinge loss.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

The SVM model got validation accuracy of almost 82.13 %.

3.3 Deep Learning Models –

a) Bag Of Words –

The approach is very simple and flexible and can be used in a myriad of ways for extracting features from documents. A bag-of-words is a entails text that counts the occurrence of words within a document. It involves two things:

1. A vocabulary of known words.
2. A measure of the presence of known words.

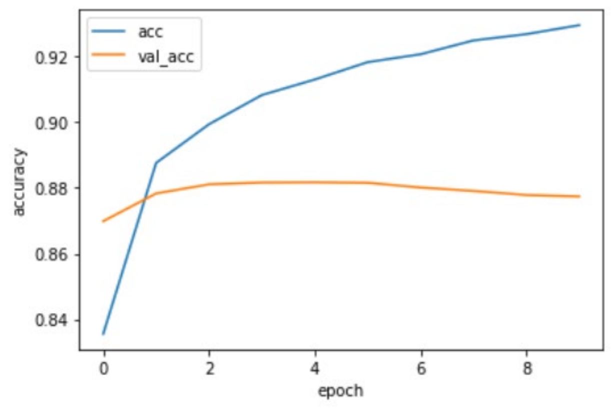
Another factor involved is term frequency and TFID to count the number of times the word is present in the corpus. It is called a "*bag*" of words because any information about the words is present. The model is only concerned with whether known words occur in the document, not where in the document.

i. Bag-of-Words Linear Model –

We wanted to see the performance when we change the hyper parameters and the below are different implementations of this model. So, we have used just 2 layers. The first layer is the input layer, and it sends the 5000 inputs to the next layer. The next layer is the bag of words layer with just one node. It then sends one output to the next layer which is sigmoid layer or activation function.

$$\text{Sigmoid} = \text{S}(\text{x}) = \frac{1}{1 + e^{-x}}$$

The sigmoid function will give us a number between 0 or 1 which will in turn help us determine whether the review is negative or positive. Further to compile the model we use the Adam optimizer, binary cross entropy loss and accuracy as the metrics. After compiling the model, it is trained for 10 epochs. After predicting the output, it shows the training accuracy of 88.24% at epoch 5. The training history is plotted as follows.



ii. Bag-of-Words with Hidden Layer –

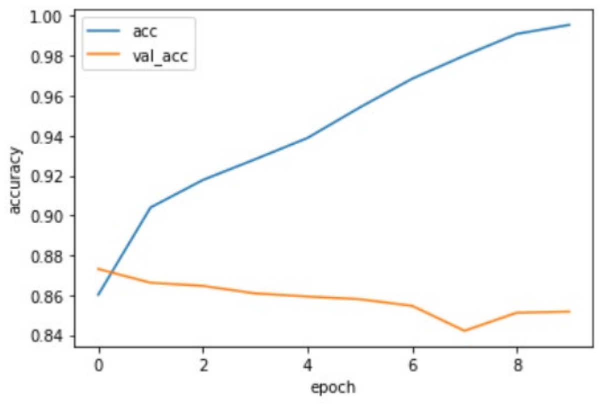
In this model we have added a hidden layer. The first layer is the input layer and it sends the 5000 inputs to the next layer. The next layer is the hidden layer with 64 nodes. It then sends outputs to the next layer which is tanh layer or activation function.

$$\text{Tanh} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

The next layer is the bag of words layer with just one node. It then sends one output to the next layer which is sigmoid layer or activation function

$$\text{Sigmoid} = S(x) = \frac{1}{1 + e^{-x}}$$

The sigmoid function will give us a number between 0 – 1 which will in turn help us determine whether the review is positive or negative. The model is now created. Further to compile the model we use the Adam optimizer, binary cross entropy loss and accuracy as the metrics. After compiling the model, it is trained for 10 epochs. After predicting the output, it shows the training accuracy of 87.56% at epoch 1. The training history is created and plotted as follows.



b) GloVe –

GloVe is an unsupervised deep learning algorithm for obtaining vector representations for words. We opted for this as it gives better results 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 GloVe algorithm consists of following steps:

1. Collect word co-occurence statistics in the form of an X-shaped word co-occurrence matrix. Each Xij element of such a matrix represents the frequency with which word i appears in the context of word j. Typically, we scan our corpus as follows: for each term, we look for context terms within a window size before the term and a window size after the term. Also, we give less weight to words that are further away, typically using the following formula:

$$\text{decay}=1/\text{offset}$$

2. Define soft constraints:

$$w^t{}_i w_j + b_i + b_j = \log(X_{ij})$$

Here wiwi - vector for the main word, wjwj - vector for the context word, bibi, bjbj are scalar biases for the main and context words.

3. Cost function –

$$J = \sum_{i=1}^{V}\sum_{j=1}^{V} f(X_{ij})(w_i^T w_j + b_i + b_j - \log X_{ij})^2$$

Here f is a weighting function which help us to prevent learning only from extremely common word pairs. The GloVe authors choose the following function:

$$f(X_{ij}) = \begin{cases} (\frac{X_{ij}}{x_{max}})^\alpha & \text{if } X_{ij} < XMAX \\ 1 & \text{otherwise} \end{cases}$$

For this project we are using the pre trained GloVe embedding model. Before we create a model, we load the GloVe embedding. The words which are not found in embedding index are all zeros using nparray. The sentences are then converted to tokens for training.

i. GloVe averaging pooling model –

In this model we have used 4 layers. The first layer is the input layer. The next layer is the embedding layer layer with 50 nodes. It then sends one output to the next layer which is Global Average Pooling 1Dwith 50 nodes. After pooling the output is passed to the last dense layer with 1 output and activation function being Signmoid. The sigmoid function will give us a number between 0 – 1 which will in turn help us determine whether the review is positive or negative. The model is now created. Further to compile the model we use the Adam optimizer, binary cross entropy loss and accuracy as the metrics. After compiling the model, it is trained for 10 epochs. After predicting the output, it shows the training accuracy of 69.38% at epoch 10. The validation accuracy is 68.77%.

i. GloVe max pooling model –

In this model we have used 4 layers. The first layer is the input layer. The next layer is the embedding layer layer with 50 nodes. It then sends one output to the next layer which is Global

Average Pooling 1Dwith 50 nodes. After pooling the output is passed to the last dense layer with 1 output and activation function being Signmoid. The sigmoid function will give us a number between 0 – 1 which will in turn help us determine whether the review is positive or negative. The model is now created. Further to compile the model we use the Adam optimizer, binary cross entropy loss and accuracy as the metrics. After compiling the model, it is trained for 10 epochs. After predicting the output, it shows the training accuracy of 86.88% at epoch 10. The validation accuracy is 83.33%.

c) LSTM –

The control flow of an LSTM is similar to that of a recurrent neural network. It processes data and forwards information as it propagates. These operations allow the LSTM to remember or forget information. The cell state and its various gates are at the heart of LSTMs. The cell state serves as a highway for relative information to be transferred all the way down the sequence chain. In theory, the cell state can carry relevant information throughout the sequence's processing. As a result, information from earlier time steps can travel to later time steps and fulfil the long-term dependency requirement. As the cell state travels, information is added or removed from the cell state via gates. The gates can learn what information is relevant to keep or forget during training.

In this model we have used 3 layers. The first layer is the embedding layer and it sends top words, embedding vector length and the maximum review length as input to the next layer. The next layer is the LSTM layer with 100 nodes. The next layer is the Dense layer with just 1 node. It then sends one output to the next layer which is sigmoid layer or activation function.

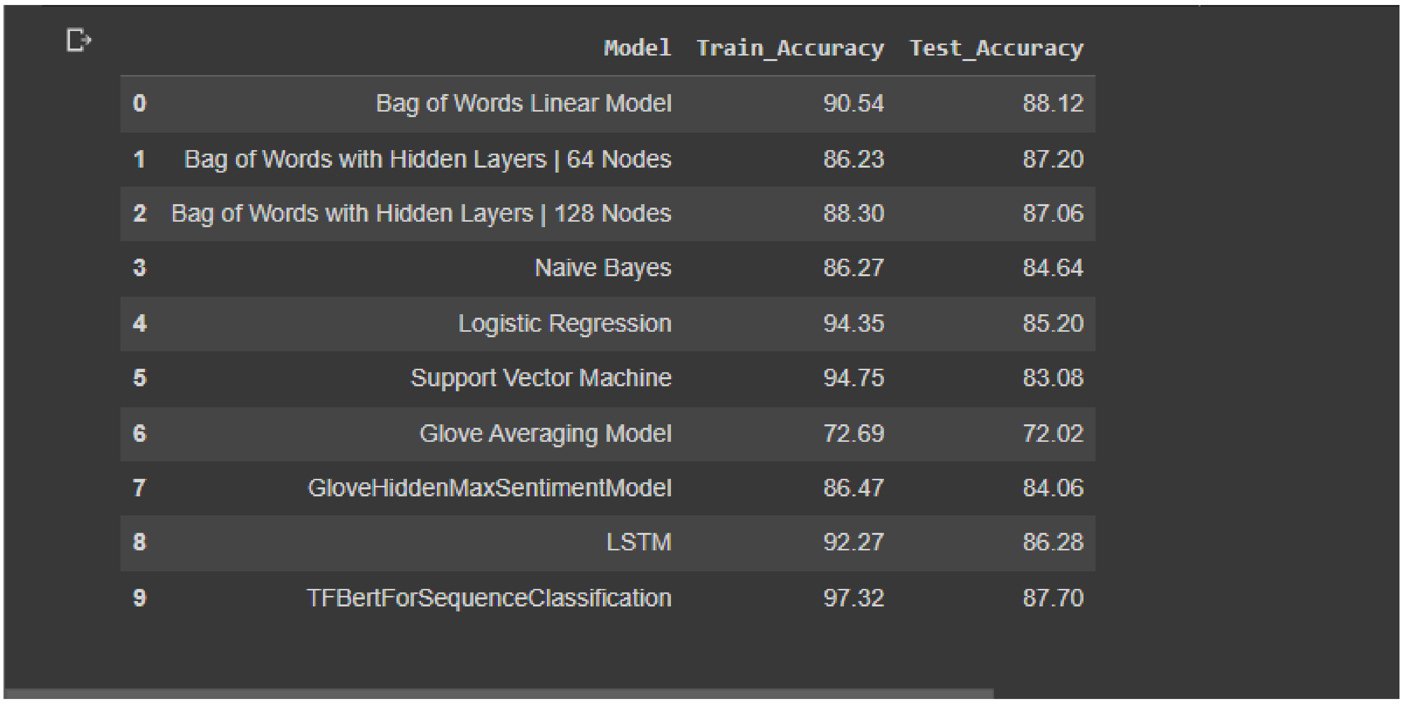
$$\text{Sigmoid} = S(x) = \frac{1}{1 + e^{-x}}$$

The sigmoid function will give us a number between 0 – 1 which will in turn help us determine whether the review is positive or negative. The model is now created. Further to compile the model we use the Adam optimizer, binary cross entropy loss and accuracy as the metrics. After compiling the model, it is trained for 5 epochs. After predicting the output, it shows the training accuracy of 92% at epoch 5. But the testing accuracy score is 87.32%.

d) TFBert for sequence classification –

In this project first we used the BertTokenizer form pretrained bert-base-uncased to tokenize the sentences. Then we encoded the dataset so that it corresponds to TFBertForSequenceClassification methods. The model built was the TF Bert For Sequence Classification.from_pretrained.
For compilation we used the Adam Optimizer and Sparse Categorical Crossentropy loss with metrics set to accuracy. The model was then fit on dataset with total 3 epochs. After predicting the output, it shows the training accuracy of 97.38% at epoch 2. The validation accuracy is 88.44%.

3.4 Evaluation –



	Model	Train_Accuracy	Test_Accuracy
0	Bag of Words Linear Model	90.54	88.12
1	Bag of Words with Hidden Layers | 64 Nodes	86.23	87.20
2	Bag of Words with Hidden Layers | 128 Nodes	88.30	87.06
3	Naive Bayes	86.27	84.64
4	Logistic Regression	94.35	85.20
5	Support Vector Machine	94.75	83.08
6	Glove Averaging Model	72.69	72.02
7	GloveHiddenMaxSentimentModel	86.47	84.06
8	LSTM	92.27	86.28
9	TFBertForSequenceClassification	97.32	87.70

Table – Models and their accuracy.

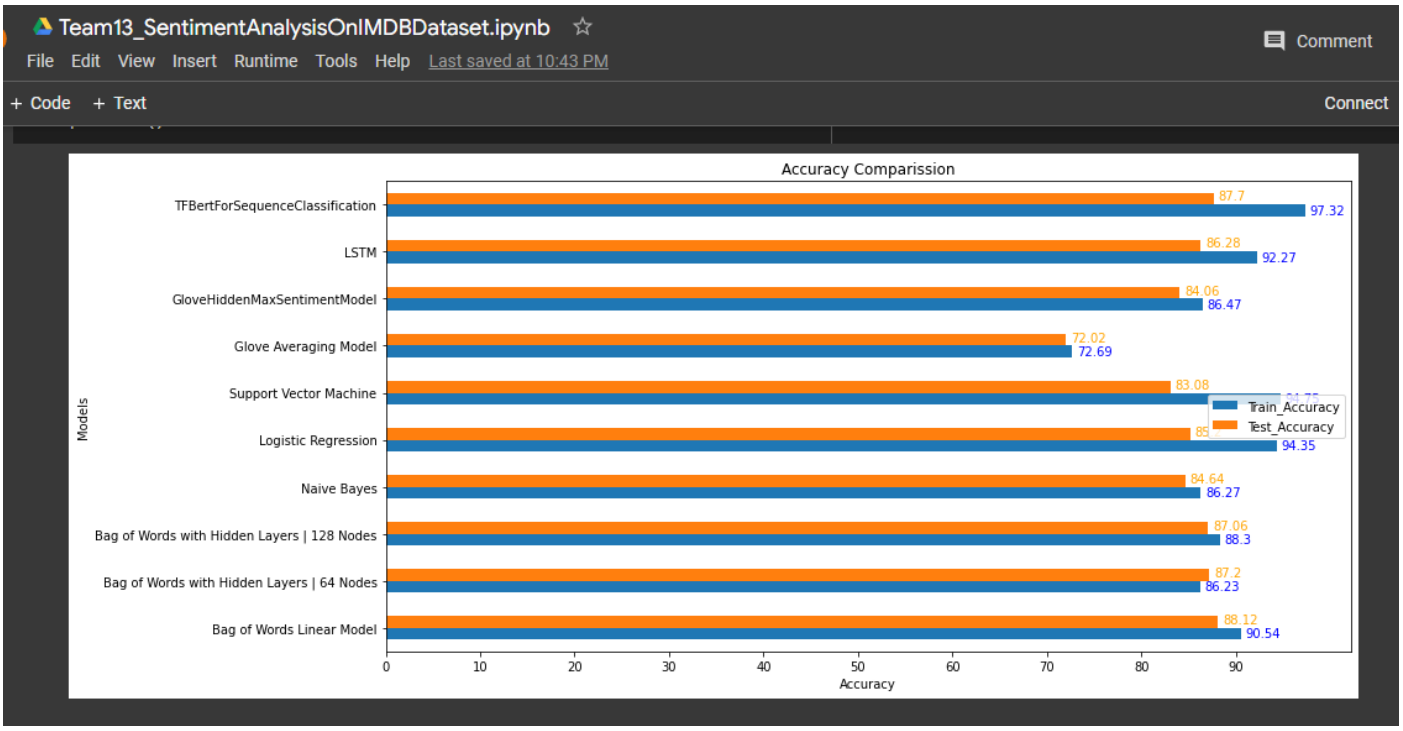


Fig – Models vs Train-Test Accuracy

This concludes that TFBert for sequence classification has the highest training and testing accuracy of about 97.38% and 88.44% respectively.

4 Difference in ACCURACY/PERFORMANCE between your project and the main projects of your references

Analysis

1 What did I do well?

In the reference paper, we used BERT and LSTM models rather than Nave Bayes and Word2Vec. This enabled us to achieve 97 percent accuracy, which was higher than the paper's 87 percent.

2 What could I have done better?

If we had a faster GPU and more processing power, we could have improved the model's accuracy by adjusting hyperparameters like learning rate and epochs. In addition, we could have used dropout layers or CNN layers in the LSTM model.

3 What is left for future work?

In future we will work on more modelssuch as EFL (Entailment as Few-Shot Learner), GraphStar,GRU, Nyströmformer to check the best suited model for this project.

Conclusion –

1) We found out that the highest accuracy was yield by **TFBert for sequence classification** with a score of 97.38%.
2) The results show that for machine learning models, the train accuracy is greater than the test accuracy. This demonstrates that the models are overfitting the data
3) We compared machine learning models and deep learning models based on the accuracies.
4) For deep learning, we can see that there isn't much of a difference between training and testing accuracy.
5) Using pre-trained models to reduce computation and achieve greater accuracy because it understands the context of the reviews.
6) The best results are as follows: BERT, LSTM, Bag of words Linear Model, Bag of with Regularization, Bag of with Hidden Layers, GloveHiddenMaxSentimentModel, Glove Averaging Model, SVM, Logistic Regression, and Naïve Bayes.

Enhancements –

- The main purpose of our paper is to highlight that deep learning algorithms show better accuracy when compared to machine models. This is clearly stated in the accuracy and table posted above.

- In our approach, we first pre-processed the data with the NLTK library using the Lemmatization technique.

- Then we used three different machine learning models to train on the IMDB dataset and similarly used deep learning algorithms like LSTM and Bag of Words.

- For ML models, the results show that the train accuracy > test accuracy.

- We have followed our reference paper to implement the existing project.

- Further enhancements include implementing Glove Encoding Deep Learning models and predicting which deep models best serve the purpose of sentiment analysis of our Movie review recommendation system.
- Using Bert large model will have better accuracy in fetching the sentiment of a sentence. It has 24 encoding layers and 340 million parameters.

- As per the performance analysis of the models, GloVe with max pooling has better performance than the model with average pooling on the same dataset.

References –

1. Xian Fan, Xiaoge Li, Feihong Du, Xin Li, Mian Wei – "Apply Word Vectors for Sentiment Analysis of APP Reviews"

2. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit – "Attention is all you need"

3. Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning

4.Has¸im Sak, Andrew Senior, Franc¸oise Beaufays – "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling"

5. A. J. Robinson, G. D. Cook, D. P. W. Ellis, E. Fosler-Lussier, S. J. Renals, and D. A. G. Williams, "Connectionist speech recognition of broadcast news,"

6. Jose Camacho-Collados , Mohammad Taher Pilehvar – "On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation

Study on Text Categorization and Sentiment Analysis”

7. Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures.

8. Yazhen Li , Xiaoge Li, Gen Yu. Research of Sentiment Classification
Based on the Chinese Stock Blog. Journal of Wuhan University (Natural
Science Edition)