# COMP 551 - Assignment 4

Negin Esmaeilzadeh Kiabani

April 18, 2024

## 1 Abstract

In this assignment, we employed four machine learning models from existing libraries—Logistic Regression, Random Forest, XGBoost, and BERT—to classify the IMDB movie reviews dataset. We utilized various data preparation techniques, including tabularizing the texts for feature extraction using linear regression on highly weighted words, embedding word2vec for baseline models, and utilizing unstructured tokenized inputs for the BERT model. Fine-tuning hyperparameters for each model was conducted through grid search, and performance metrics such as accuracy and AUROC were evaluated on the test set. Additionally, we examined the attention weights in the BERT model to understand its relation to correct or incorrect sentiment predictions. Our findings showcase the effectiveness of BERT in outperforming traditional models, while also revealing the limitations of word2vec embedding in improving the performance of baseline models. Further exploration into hyperparameter fine-tuning and dimensionality reduction techniques is warranted for enhancing model performance.

## 2 Introduction

For this assignment, we used four machine learning models from existing libraries, Logistic Regression, Random Forest, XGBoost, and BERT model to classify the IMDB movie reviews dataset. We used different data preparation methods such as tabularizing the texts (feature extraction) using the highly weighted words with a linear regression, embedded word2vec for the same baseline models, and unstructured tokenized inputs used for the BERT model. We conducted a grid search to fine-tune hyperparameters for each model and measured the accuracy and AUROC as performance metrics of the models on the test set. We also examined the relation of attention weights in the BERT model with correct or incorrect prediction of the review sentiments.

### 2.1 Models

#### 2.1.1 Logistic Regression (LR)

Logistic Regression is a statistical method used for binary classification tasks, predicting the probability of an instance belonging to a particular class. It models the relationship between the independent variables and the binary outcome using the logistic function. The logistic function, also known as the Sigmoid function, maps any real-valued number to the range [0, 1].
In this assignment we used the LR method implemented in the Scikit-learn package. We ran a grid search to choose between the regularization strength and maximum iterations as hyperparameters.

#### 2.1.2 Random Forest (RF)

Random Forest is an ensemble learning method used for both classification and regression tasks. It takes advantage of using a multitude of decision trees during training and outputs the most frequent classes predicted by the individual trees.
In this assignment, we utilized the Random Forest implementation provided by the Scikit-learn library. Similar to Logistic Regression, we conducted a grid search to optimize hyperparameters, such as the number of trees in the forest and the maximum depth of the trees.

#### 2.1.3 XGBoost

Extreme Gradient Boosting is a highly efficient and scalable implementation of gradient boosting machines. At its core, XGBoost builds an ensemble of weak learners, typically decision trees, in a sequential manner. Each new learner is trained to correct the errors made by the existing ensemble. This iterative process continues until a predefined number of weak learners (trees) are created, or no further improvement can be achieved.
In our assignment, we used the build-in XGBoost model of python to explore its capabilities in comparison to Logistic Regression and Random Forest. Similar to the other methods, we conducted a grid search to fine-tune hyperparameters, such as maximum tree depth, number of estimators, and regularization strength.

#### 2.1.4 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a state-of-the-art natural language processing (NLP) model developed by Google. Unlike traditional language models that process text sequentially, BERT is bidirectional, meaning it can understand the context of a word based on both its preceding and following words. The model architecture consists of a fixed-length sequence of 512 tokens as input. Each token represents a word or subword piece in the input text, and BERT tokenizes the text using a technique called WordPiece tokenization. This tokenized input is then fed into a stack of transformer encoder layers. The original BERT-base model comprises 12 transformer layers, each with 12 self-attention heads. Within each layer, the self-attention mechanism allows the model to attend to different parts of the input sequence simultaneously, capturing various dependencies and relationships within the text. Additionally, each transformer layer includes feed-forward neural network layers for further processing. BERT also incorporates positional embeddings to encode the sequential order of tokens in the input sentences. BERT's key innovation lies in its pre-training strategy. It is pre-trained on large corpora of text data using two unsupervised learning tasks: masked language modeling (MLM) and next sentence prediction (NSP). In MLM, BERT randomly masks some of the input tokens and predicts them based on the context provided by the surrounding tokens.
In this assignment, we utilized the basic pre-trained Bert model from the Transformers library and fine-tuned this model on our text classification task. To minimize memory requirements, fine-tuning was performed only on the last layer of the BERT model. Inline with the previous methods we conducted a search over main hyperparameters such as learning rate and batch size set optimum parameters for the finalized model.

### 2.2 Datasets

#### 2.2.1 IMDB Reviews

The Large Movie Review Dataset is a comprehensive collection designed for binary sentiment classification, intended to serve as a benchmark for sentiment analysis tasks. With a total of 50,000 movie reviews, evenly split into 25,000 for training and 25,000 for testing, the dataset offers a balanced distribution of positive (score >= 7 out of 10) and negative (score of <= 4 out of 10) labels. The dataset includes bag of words (BoW) features in LIBSVM format and expected ratings for tokens. In order to prepare the data as inputs for the models to be trained we used three different methods which are explained later.

#### 2.2.2 Data Preperation

**Tabolized format (feature extraction)**: As the first method to get the non-language baseline models such as LR, RF, and XGBoost to work, we used the bag of words as features and their number of appearance times in labeled reviews. However, since the Bag of Words (BoW) contains 89,526 words, many of which are irrelevant to sentiment (such as rare and stop-words), we tried to filter out these noise features to enhance our model in terms of both speed and accuracy. In order to do this, we filtered out the words that appear in less than 1% of the documents and words that appear in more than 50% of the documents which resulted in 1744 features. Then, again to remove noisy features and reduce the computational cost, we ran a Linear Regression model and assessed the weights for each feature when predicting the ratings related to each review and selected the top 500 features. In Figure 2 you can see how the Linear Regression weight magnitude is related to the sentiment we get from 10 top words.

**Unstructured text data**: We used this method for the BERT model. Since LLMs are already trained to work with text type inputs we directly trained the BERT model on the unstructured text data without breaking them into tabular format. The only steps taken were tokenizing the text reviews, adding a class token at the beginning of each text, adding a separator token at the end of each sentence of a specific text, and padding each sentence less than the maximum length (512 tokens) with pad tokens and cutting ones more lengthy.

**Seq2Vec embedding**: As an alternate method of the tabularizing format for the baseline methods(LR, RF, XGBoost) we used wikipedia2vec package as a word-to-vector convertor to project the tokens from the IMDB review onto embedding space and then used the RF, LR, XGBoost models to classify the embedded documents and examine if in this way models can train the words more accurately like an LLM model. In order to make the text size similar we used the same padding/cutting method as we did during text tokenizing for the BERT model but instead of pad tokens we padded with spaces as we lacked many words in the wikipedia2vec package such as the word 'pad'.

## 3 Results

### 3.1 Fine-tuning hyperparameters for different models

### 3.2 BERT

The BERT model was trained for 3 epochs and evaluated with various batch sizes (16, 32, 64) and learning rates (0.1, 0.01, 0.01). Figure 1 represents how train and validation accuracy was affected using each pair of these two

hyperparameters. As it can be seen a batch size of 64 and a learning rate of 0.01 leads to the best performance. Therefore, we kept the above settings to train the model and evaluate it on the test data.

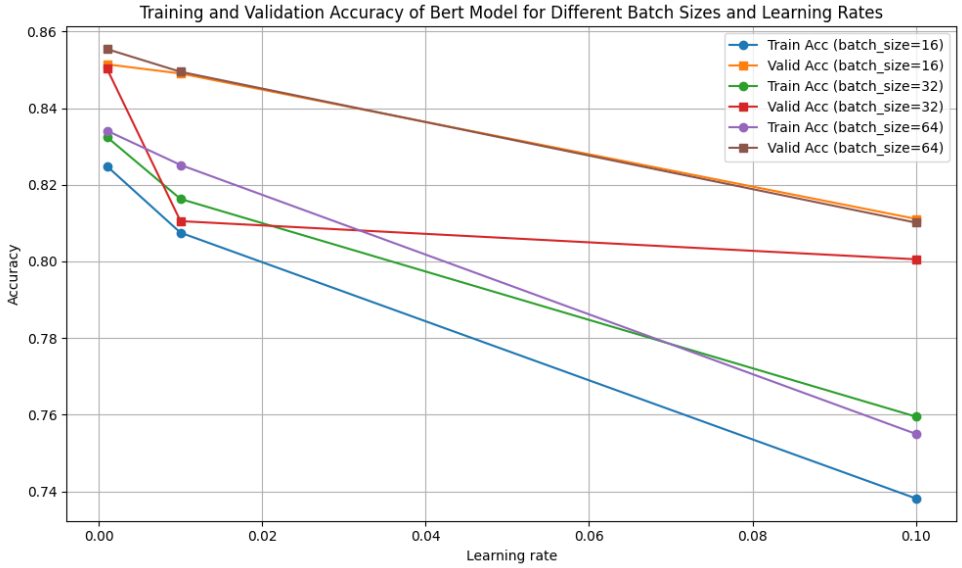


Figure 1: Monitoring the train loss and gradients calculation

#### 3.2.1 LR

We measured validation the validation accuracy (using 3 fold cross validation) for each fitted LR model with hyperparameters selection in the range of max-iter (maximum iterations) = [50, 100, 200] and C (inverse of regularization strength) = [10000, 1000, 100, 10, 1, 0.1, 0.01]. The results show that C = 0.01, and max-iter = 50 lead to higher validation accuracy, therefor we kept these settings to measure the model performance on our test set.

#### 3.2.2 RF

We measured validation accuracy (using 3 fold cross validation) for each fitted RF model with hyperparameters selection in the range of n-estimators (number of estimators) = [50, 100, 150, 200, 500] and max-depth (maximum tree depth if existed) = [None, 10, 20, 30, 50, 80, 1001]. The results show that n-estimators = 500, and max-depth = 100 lead to higher validation accuracy, therefore, we kept these settings to measure the model performance on our test set.

#### 3.2.3 XGBoost

We measured validation accuracy (using 3 fold cross validation) for each fitted XGBoost model with hyperparameters selection in the range of lambda (regularization strength) = [0.1, 1.0, 10.0, 100], max-depth (maximum tree depth) = [3, 6, 9] 50, 80, 1001] and n-estimators (number of estimators) = [50, 100, 150, 200]. The results show that lambda = 100, max-depth = 9, and n-estimators = 200 lead to higher validation accuracy therefore we kept these settings to measure the model performance on our test set.

### 3.3 Models performance

Table 1 shows the performance of each model based on train accuracy, test accuracy and AUROC scores. As it can be seen the LLM model performed the best.

| Model | Train Accuracy | Test Accuracy | Test AUROC |
|---|---|---|---|
| **LR** | 78.82% | 75.79% | 0.81 |
| **RF** | 99.97% | 74.10% | 0.79 |
| **XGBoost** | 87.38% | 75.28% | 0.80 |
| **BERT** | 77.25% | 83.84% | 0.88 |

Table 1: Train accuracy, test accuracy, and AUROC scores for baseline models and the BERT model as LLM

### 3.4 Effect of word2vec embedding

We retrained the LR, RF, and XGBoost models on the text data taken into vector embedding space using Wikipedia2Vec. Table 2 shows the results for this purpose. The results indicate a much lower performance of previous models on this newly derived data which can be due to several reasons. First, the embedding is not powerful in terms of a wide range of vocabulary. During the embedding, we watched around 1/3 of the tokens fail to be mapped to a vector. This caused

many masked words we replaced with spaces and some of which were classifying words. Second, We used the word2vec version with 100 dimensions as vector sizes for each individual word which led us to have too many features for a single review text (>1000). This causes so many noisy features and many of these simple baseline model are not very powerful not to be affected by the noise. In fact, this can be the main reson of the overfiiting sign we can see on these models performances.

| Model | Train Accuracy | Test Accuracy | Test AUROC |
|---|---|---|---|
| **LR** | 97.19% | 59.27% | 0.63 |
| **RF** | 1.00% | 59.34% | 0.58 |
| **XGBoost** | 99.96% | 56.70% | 0.59 |

Table 2: Train accuracy, test accuracy, and AUROC scores for baseline models after word2vec embedding

### 3.5 Bert attention examination

We examined the BERT attention matrix between the words and the CLS token for some of the correctly and incorrectly predicted documents. To do this, we randomly selected a head from 12 attention heads and we examined the attention weights of the last encoder block (last layer) from 12 layers of BERT. Figure 2 shows the attention weights relative to the top 20 highly attended tokens for an example of correctly predicted review sentiment and figure 3 shows the same thing for an incorrectly predicted example. As we see, when predicting correctly the model tends to attend more to some polarized words related to the label such as 'loved', 'great', and 'fantastic', whereas the same attention head in the incorrectly predicted example attended more to some stop words (probably couldn't find similar polarized words) which lead to wrong prediction.

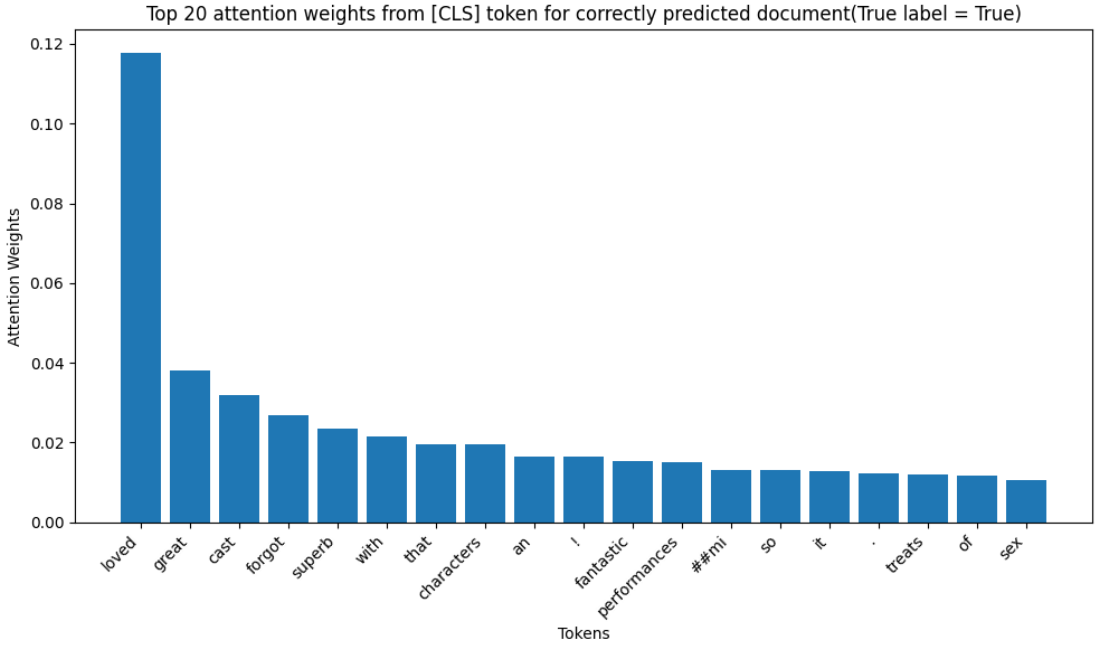


Figure 2: Top 20 attention weights from [CLS] token for correctly predicted document

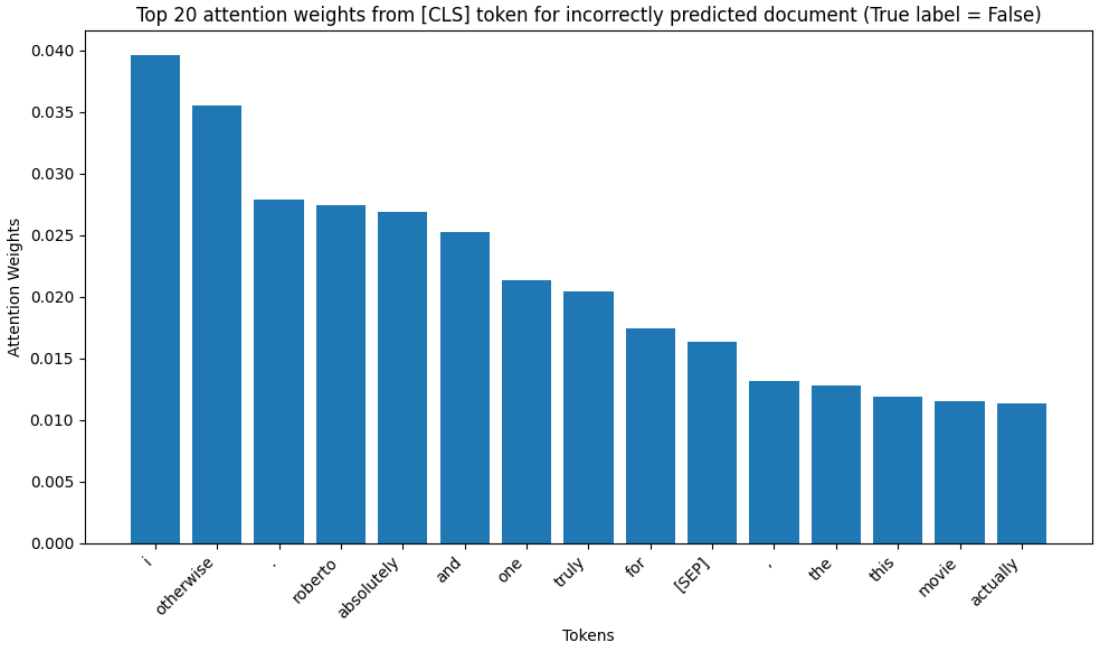


Figure 3: Top 20 attention weights from [CLS] token for incorrectly predicted document

# 4 Discussion and Conclusion

## 4.1 BERT model outperformed LR, RF, and XGBoost

The BERT model outperformed Logistic Regression (LR), Random Forest (RF), and XGBoost due to its ability to capture intricate contextual relationships within text data. Unlike traditional models like LR, RF, and XGBoost, which rely on handcrafted features and may struggle with understanding nuanced linguistic patterns, BERT utilizes a transformer-based architecture that learns contextual embeddings directly from raw text, enabling it to capture semantic meaning and context more effectively. This allows BERT to excel in tasks where understanding complex language structures is paramount, resulting in superior performance compared to traditional models.

## 4.2 Word2vec embedding couldn't improve the permonce of baseline models

The performance results of models after using wor2vec embedding show an exact sign of model overfitting. The failure could be attributed to several factors. Firstly, the embedding may lack the necessary breadth and depth to effectively capture the diverse vocabulary present in the data. Additionally, the choice of Word2vec embeddings with 100 dimensions per word may have contributed to the generation of too many features for each individual review text, leading to feature space overload. This abundance of noisy features could overwhelm simple baseline models, making them prone to overfitting and ultimately diminishing their performance. Therefore, the ineffectiveness of Word2vec embeddings in enhancing model performance underscores the importance of selecting embeddings that adequately capture the complexity and diversity of the underlying data while avoiding excessive feature dimensionality that can introduce noise and hinder model generalization. Further investigation needs to examine the effect of fine-tuning all hyperparameters of these models and finding a way to pool the information of 100 demand vectors into smaller sizes. Perhaps, dimension reduction methods such as PCA can be helpful to solve this problem.

# 5 Statement of Contributions

This work was done by a single group member.

# References

[1] Anand Jha, Atul. (2022). *BERT Testing on IMDb Dataset - Extensive Tutorial. Kaggle.* `https://www.kaggle.com/code/atulanandjha/bert-testing-on-imdb-dataset-extensive-tutorial/notebook#BERT-Model`

[2] Wikipedia2Vec. (n.d.). *Pretrained Embeddings.* `https://wikipedia2vec.github.io/wikipedia2vec/pretrained/`