COMPARING THE PERFORMANCE OF LSTM (Long Short-Term Memory) and (Bidirectional Encoder Representations from Transformers) ON AMAZON KINDLE REVIEW FOR SENTIMENT ANALYSIS

INTRODUCTION

Sentiment analysis is the use of computational methods to categorize pieces of text and understand the writer's opinion. This process is made possible by Artificial Intelligence through one of its branches called natural language processing (NLP), this branch of AI allows computers to understand text and voice in a way like humans, or even better. Sentiment analysis is applied in various fields, such as market research, brand reputation management, customer feedback analysis, and social media monitoring.

Sentiment analysis is also referred to as opinion mining. Feedbacks from customers on E-commerce platforms are mined and pre-processed computationally to help formed better business decisions by business stakeholders and increase customers' satisfaction. Sentiment Analysis as an aspect of natural language processing is the new and seamless approach to business development. It is crucial to the growth of many businesses in this technological era and its processes adapts the use of different techniques from deep learning and more advanced machine learning approach.

The main research question in this topic is asking through methodologies and experiments on which Natural Language Processing model, LSTM or BERT models, will perform better when used for sentiment analysis.

The expected outcome of this comparison is to gain insights into the strengths and weaknesses of LSTM and BERT models for sentiment analysis, understand their respective advantages and limitations, and identify which model performs better in terms of performance metrics which will be discussed in the later part of this report. This report can provide valuable information for researchers, practitioners, and organizations interested in using NLP models for sentiment analysis and can contribute to the advancement of AI in the field of sentiment analysis.

BACKGROUND

The background on the comparison of LSTM and BERT using sentiment analysis on Amazon Kindle reviews will help understand the evolution and traditional methods used for sentiment analysis and how deep learning methods have significantly improved the performance of sentiment analysis models. The journey of sentiment analysis has travelled from the traditional machine learning approaches of feature engineering to neural network architectures like LSTM and BERT that can learn features from input data automatically. According to Katrekar, and AVP, (2005) in their work, from purchasing a product to choosing a school, others' opinion has a significant impact in our daily decision-making process.

Traditional approaches to sentiment analysis relied heavily on algorithms such as Naive Bayes, Random Forests, Support Vector Machines (SVM). These are machine learning solutions that are designed to conform with feature engineering which means that inputs are required to be pre-processed before being fed into them for modelling. Traditional approaches require significant domain expertise to develop a set of features with high level of accuracy that will represent a text's sentiment and the approaches were time consuming. Their algorithms also do not have capacities to capture the nuance and context of language which is crucial for sentiment analysis. According to Moraes et al. in their research, experiments indicate that SVM requires a high number of support vectors to classify reviews as expressing positive or negative opinions at document level.

As the amount of user-generated content available on the web increases, sentiment analysis has become a famous field of research and has been widely applied to product reviews where the polarity of the reviews can help businesses make informed decisions. The traditional methods of sentiment analysis have been extensively studied and they have their limitations and as a result, recent research has focused on deep learning-based approaches such as LSTM and BERT.

LSTM and BERT are neural network architectures that are configured to understand features automatically from input data without the intervention of the traditional manual feature engineering approach and this makes them suitable for sentiment analysis problem where the context in which a text appear may be used to determine its sentiment. According to research by Balakrishnan, et al., when compared with supervised machine learning, deep learning proves to be better for predicting sentiments.

LSTM (Long Short-Term Memory) has been widely used for sentiment analysis tasks due to its ability to model long-term dependencies in text which is done by incorporating a specialized memory cell that allows it to remember or forget information over time selectively. LSTM memory cell is controlled by the input gate, forget gate and the output gate respectively.

(Bidirectional Encoder Representations from Transformers) has also proven to be highly effective neural network algorithm for many NLP task including sentiment analysis. IT is designed to pretrain language models on large amounts of text data and is highly effective in capturing the context of a language.

Comparing LSTM and BERT models for sentiment analysis can provide valuable insights into the weaknesses and strength of their architectures for this task. LSTM due to its configuration may be better suited for task that involve longer texts and where long-term dependencies are important, while BERT may be more effective for task that involves capturing of more contextual information.

According to research conducted by Geetha, et al., they concluded that BERT outperforms well in accurate prediction than using the traditional feature extraction and machine learning models depending on the performance metrics used.

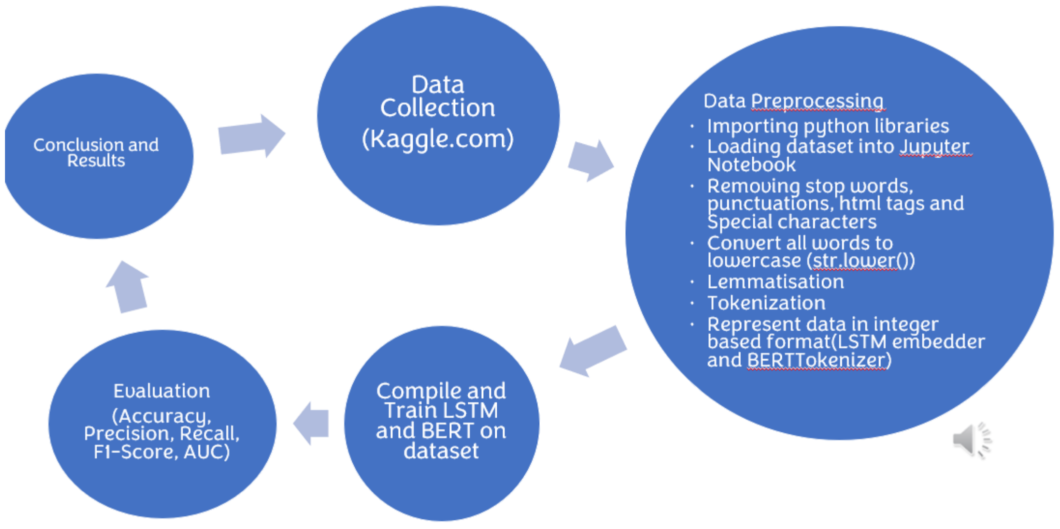
Some studies have also shown LSTM to be competitive with BERT in certain by exploring the combination of the two models to leverage the strengths of both approaches. According to a research by Aysu Ezen-Can, on comparison of LSTM and BERT on small corpus, LSTM models can achieve better performance when trained on small data than BERT model.

OBJECTIVE

The objective of this research is to preform sentiment analysis on the Amazon Kindle Book review data using LSTM and BERT in identifying customers' reactions through texts from word bags of their reviews. This work will examine the performance of the two models by comparing them and drawing inferences based on results from their Accuracy, Precision, Recall, F1-score and AUC metrics.

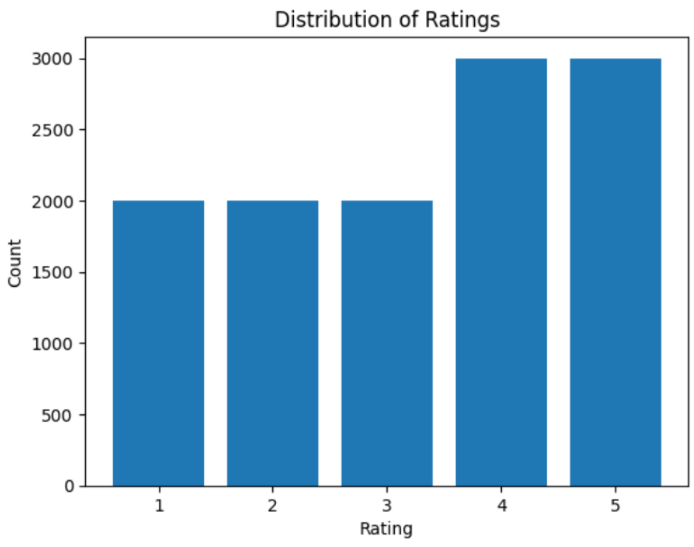
METHODOLOGY

The methodology for comparing LSTM and BERT is to evaluate their performance on a benchmark dataset for a specific natural language processing task such as sentiment analysis and analyse their results using appropriate statistical measures. A flowchart of the methodologies will be shown and discussed in this section.


Conclusion and Results
Data Collection (Kaggle.com)
Data Preprocessing
Importing python libraries
Loading dataset into Jupyter Notebook
Removing stop words, punctuations, html tags and Special characters
Convert all words to lowercase (str.lower())
Lemmatisation
Tokenization
Represent data in integer based format(LSTM embedder and BERTTokenizer)
Compile and Train LSTM and BERT on dataset
Evaluation (Accuracy, Precision, Recall, F1-Score, AUC)

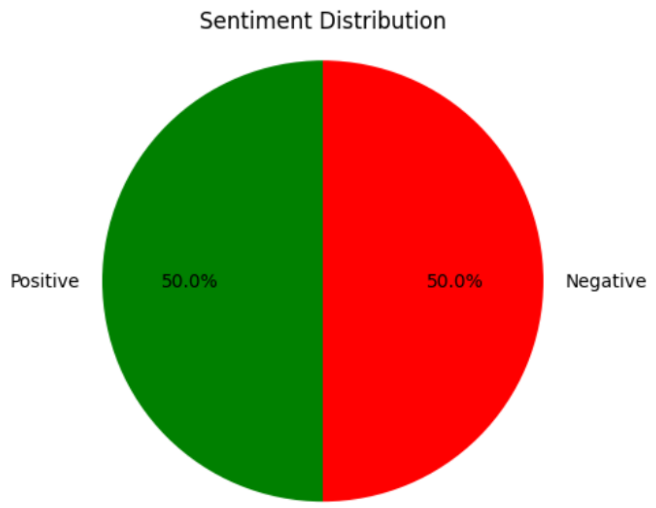
The quality and suitability of data is crucial to the success of any machine learning task. The first step to solving any machine learning problem is data collection and pre-processing. This process includes tasks such as data cleaning, feature extraction and data scaling which is sometimes referred to as data normalization.

The data for this research was collected from Kaggle website, it contains twelve thousand user reviews on the amazon kindle book product from May 1994 to July 2014. A pre-processing function was created to remove unnecessary information such as stop words, html tags, punctuation marks, numbers, and digits, the function also uses WordLemmitizer library of natural language toolkit to lemmatize the texts and classify words to their base form, this will allow words such as 'run', 'running' and 'ran' to be interpreted as one. Exploratory data analysis was done on the data. The sentiment has integers 1,2,3,4 and 5 which illustrated an imbalanced structure when plotted as seen below.

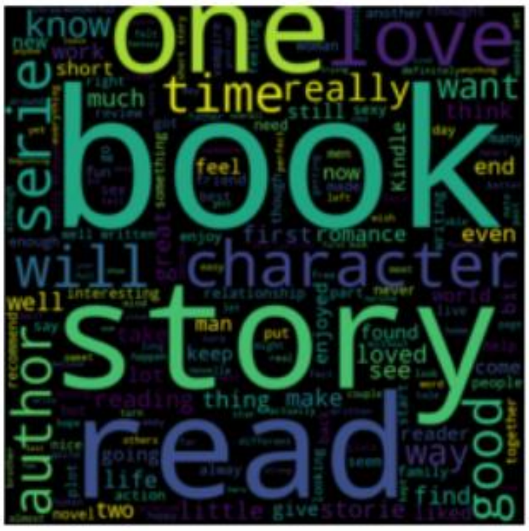


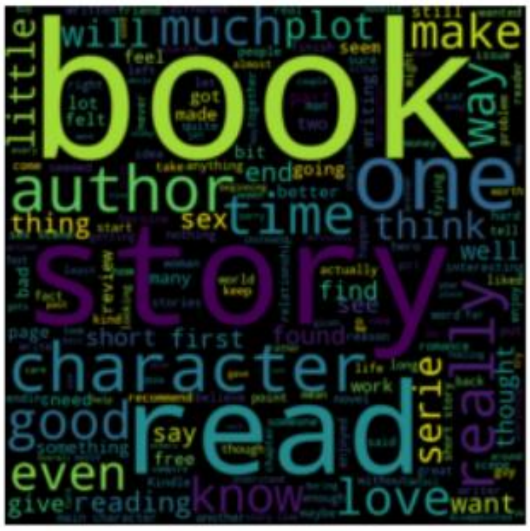
To ensure that the sentiment label data is balanced, the rating numbers were converted into binary labels by classifying numbers 1,2 and 3 as negative reviews label 0 and number 4 and 5 as positive reviews label 1. This binary classification made the sentiment label suitable for use in the models by making sure that the model can learn from an equal amount of data from each class of positive and negative reviews, this is also expected to improve the model's ability to generalize well on unseen data. An imbalanced dataset can lead to biased model that may be better at predicting the dominant class and perform poorly on the minority class.

A Pie chat was used to visualize the distribution of sentiment labels in the dataset before feeding it into our LSTM and BERT model. The data set appeared balanced after pre-processing in the visualization as seen below.



To communicate the insights from the data, the WordCloud function was imported from WordCloud library to show the size of the most frequent words in the pre-processed text. The visualization displays the most impactful words in each category of the review which can also be useful for feature engineering that was discussed in the background section of this work. This can also help users of this research to facilitate better decision making immediately.





The dataset was separated into feature and target data. The target data **y** comprising of the sentiment label is used to train the model to predict the sentiment label of a featured data **X** which comprises of the cleaned text. The target variable **y** is the desired output the model will be predicting.

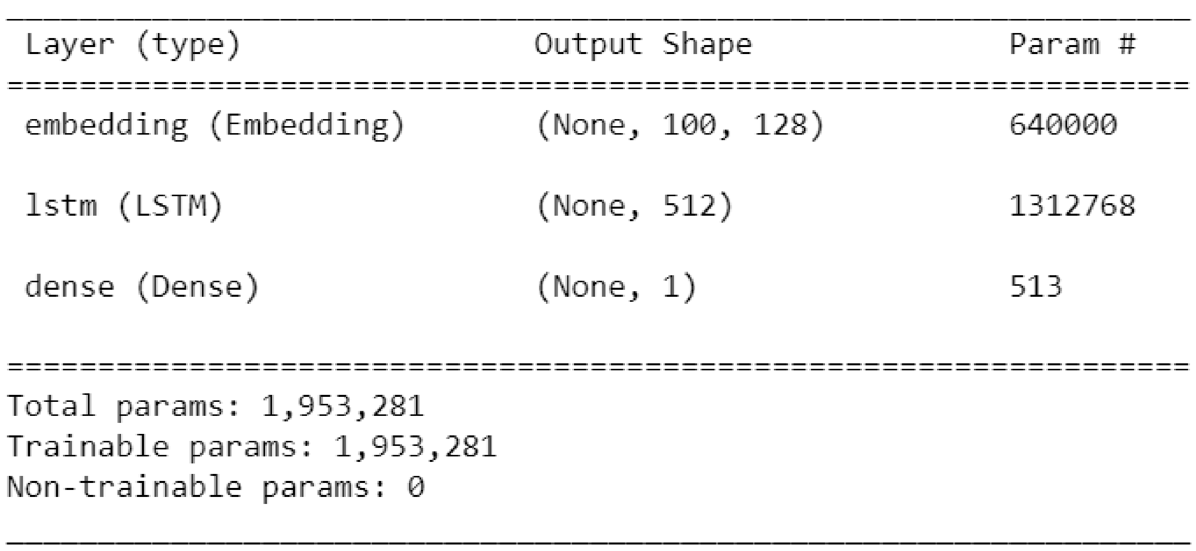
The data was divided into training and validation data to allow for the model to be trained on one set of the data and its performance to be evaluated on another set of the unseen data. This was done to prevent overfitting and to ensure that the model can generalize well to new data.

Tokenization prepares textual data for training deep learning models and allows the model to process the text data by converting it into a numerical format that can be fed into the model. The **Tokenizer** class in Keras was applied on the cleaned text before feeding it to the LSTM model. Bert on the other hand has its own tokenizer called BertTokenizer which tokenizes texts in a compatible way that the Bert architecture understands.

The pre-processed text comprises of sequences of words that can be of varying length which will not be accepted to train a neural network as it requires that all input sequence have the same length.

In LSTM, sequence padding was used to ensure that all inputs have the same length. It operates by setting a maximum length for every word, according to the article "A comprehensive study of deep learning for text categorization: A review", this method helps to truncate all sequences to the specified maximum length if they are longer than the set value or padded with zeros if they are shorter than the maximum value. This process ensures that data has a manageable size, and the neural network can be efficiently trained.

LSTM Architecture

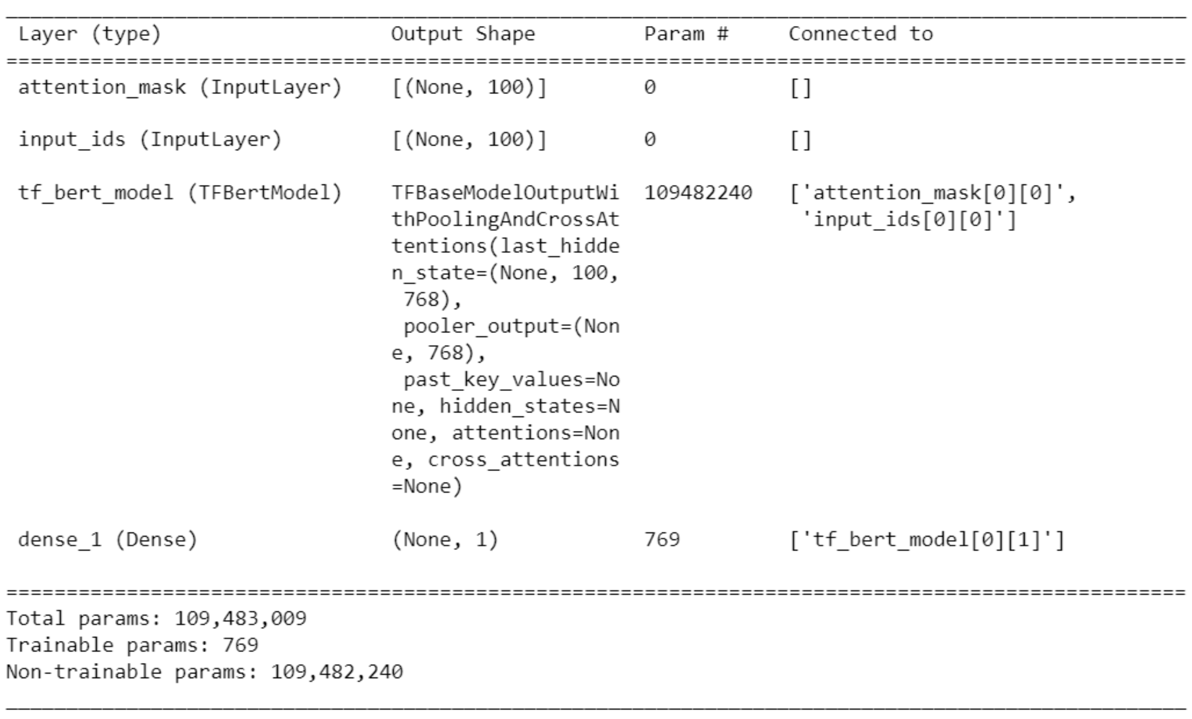


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 128)          640000

 lstm (LSTM)                 (None, 512)               1312768

 dense (Dense)               (None, 1)                 513

=================================================================
Total params: 1,953,281
Trainable params: 1,953,281
Non-trainable params: 0
_________________________________________________________________
```

Padding in BERT modelling is slightly different from other sequence models. As explained in the article "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", the padding process in BERT modelling is different from other sequence models. It is typically added to the right-hand side of the input sequence to enable it to align with the attention mask that is used in the BERT model. BERT has a pad token that is added to the input sequence as many times as needed to make all sequences in a batch have the same length. The tokenizer.encode_plus() function in the transformer library is mostly used to tokenize and pad text for BERT models.

After padding, the padded sequences are stored for further use in the neural network model.

BERT ARCHITECTURE

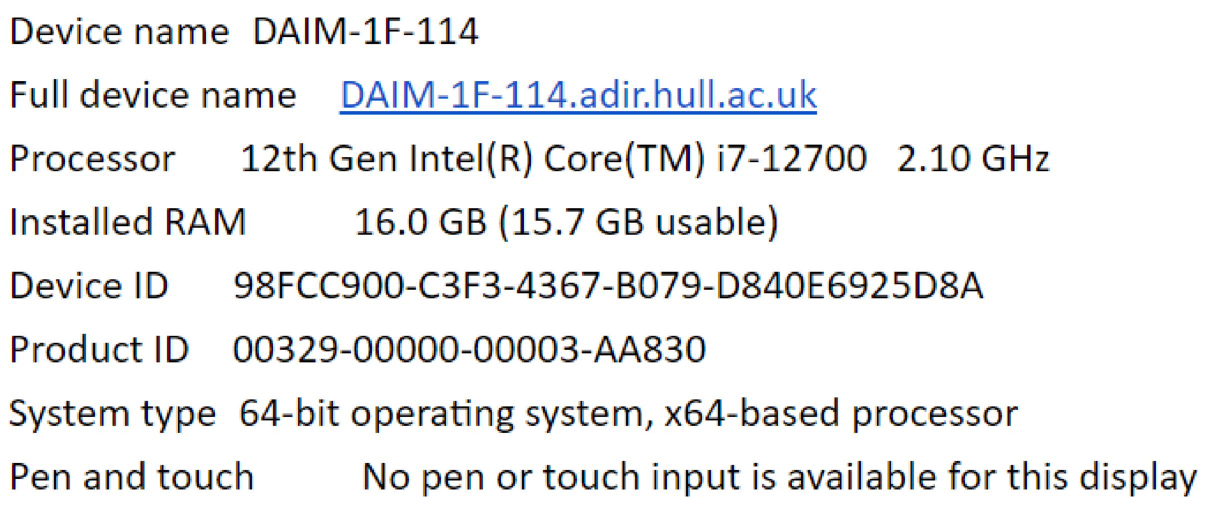


```
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 attention_mask (InputLayer)    [(None, 100)]        0           []

 input_ids (InputLayer)         [(None, 100)]        0           []

 tf_bert_model (TFBertModel)    TFBaseModelOutputWi  109482240   ['attention_mask[0][0]',
                                thPoolingAndCrossAt               'input_ids[0][0]']
                                tentions(last_hidde
                                n_state=(None, 100,
                                 768),
                                 pooler_output=(Non
                                e, 768),
                                 past_key_values=No
                                ne, hidden_states=N
                                one, attentions=Non
                                e, cross_attentions
                                =None)

 dense_1 (Dense)                (None, 1)            769         ['tf_bert_model[0][1]']

==================================================================================================
Total params: 109,483,009
Trainable params: 769
Non-trainable params: 109,482,240
__________________________________________________________________________________________________
```

The two models were compiled and trained, the sigmoid binary classification output was applied on the two models and their hyperparameters were finetuned using the training/validation set.

EXPERIMENTS

The properties of the machine used for this experiment are shown below.



Device name	DAIM-1F-114
Full device name	DAIM-1F-114.adir.hull.ac.uk
Processor	12th Gen Intel(R) Core(TM) i7-12700 2.10 GHz
Installed RAM	16.0 GB (15.7 GB usable)
Device ID	98FCC900-C3F3-4367-B079-D840E6925D8A
Product ID	00329-00000-00003-AA830
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

The dataset is divided by separating 20% of the data for validation and 80% for training using the train_test_split function from scikit-learn. The function returns four variables representing the features and labels for the training and validation sets.

The two models were experimented on the dataset based on their architectural designs and an architectural summary was printed for debugging and optimizing their architectures as well as understanding the computational resources required for training and inference.

In the LSTM, the data is tokenized using a tokenizer that has a vocabulary size of 5000 and it is padded with a maximum sequence of 100. The LSTM model consists of an embedding layer that converts each word in the input sequence into vector size of 128 which is followed by an LSTM layer set to a unit of 512 and a dropout rate of 0.2. The dropout and recurrent_dropout parameters specify the fraction of input units to drop during training to prevent overfitting. A dense layer with a sigmoid activation is added for binary classification. The model was trained using the Adam optimizer with a learning rate of 0.001 and binary cross-entropy loss function. I trained the model for 20 epochs with a batch size of 32 and early stopping is applied with a patience of 3. The hyperparameters were adjusted until a reasonable result was achieved.

In the BERT model, the data was encoded and tokenized using the BERT tokenizer with a maximum sequence of 100. The pretrained BERT model 'bert-base-uncased' was loaded and a classification layer with a sigmoid function is added on top. The BERT layers are frozen, and the Adam optimizer is also applied to the model with binary cross entropy loss function. The model is trained for 20 epochs with a batch size of 64 and early stopping of patience 3 was applied. The hyperparameters were also adjusted until a reasonable result was achieved.

A time visualization graph was plotted to visualize the time taken for each model to be trained. Confusion metrics of the two solutions was calculated on the validation set to show the following.

True positives: This shows the number of correctly predicted positive samples.

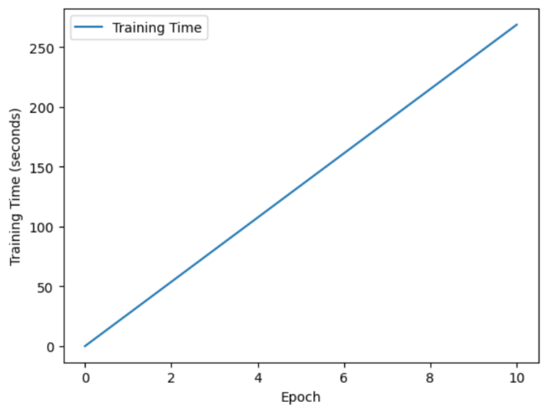
True negatives: This is the number of correctly predicted negative samples.

False positives: This is number of negative samples that were predicted incorrectly as positive.

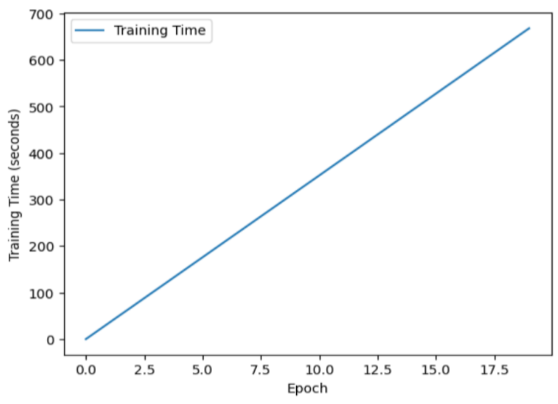
False negatives : The is the number of positive examples that were predicted incorrectly as negative.

The LSTM training time graph is compared with the BERT training time graph

LSTM training time graph

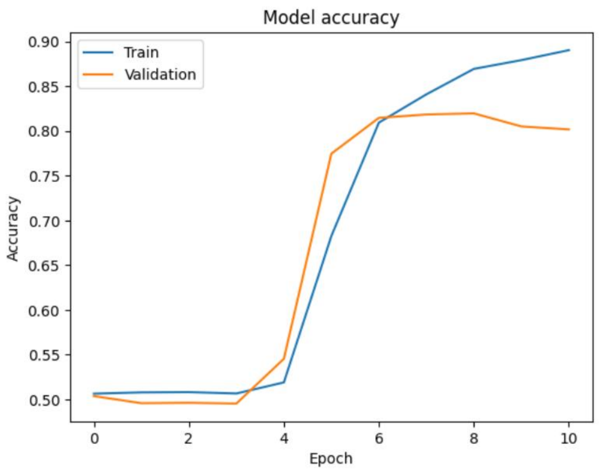


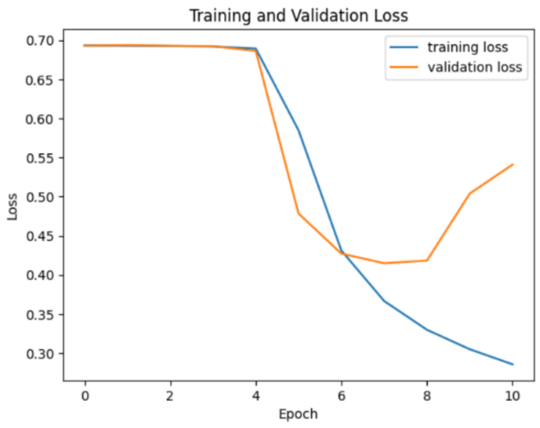
BERT training time graph



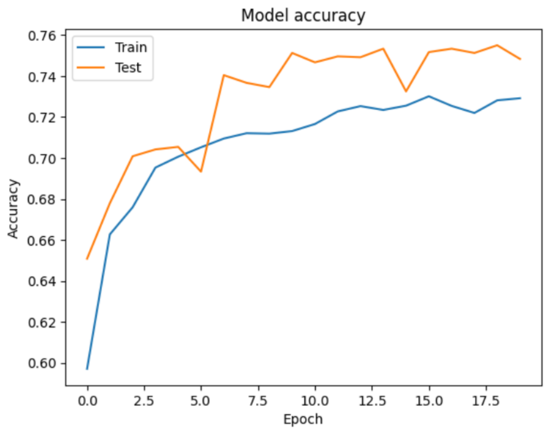
A visualization was shown to graphically illustrate how well the models trained on the validation set. The two models were evaluated based on how well they perform on their validation sets.

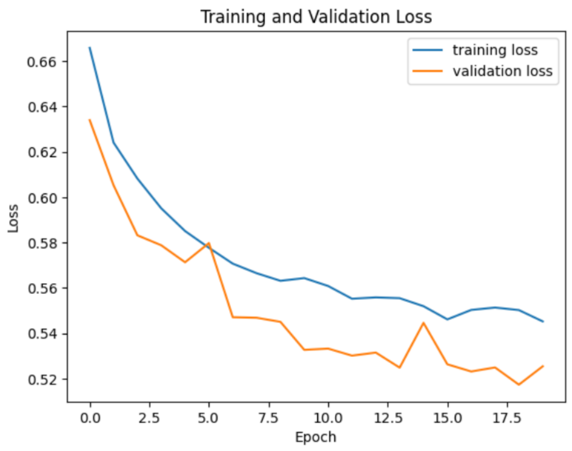
LSTM Evaluation Graph



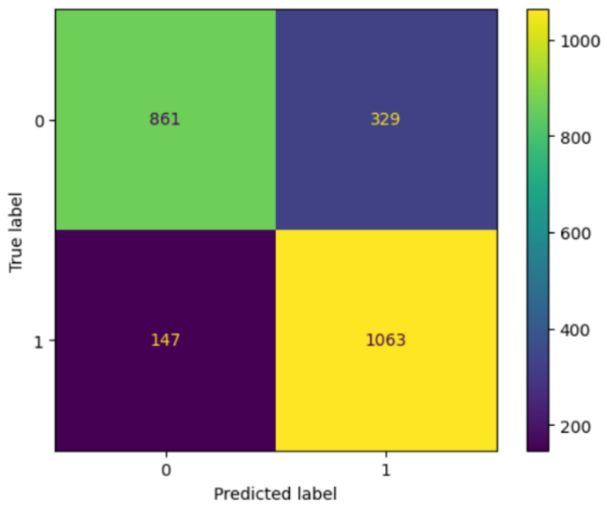


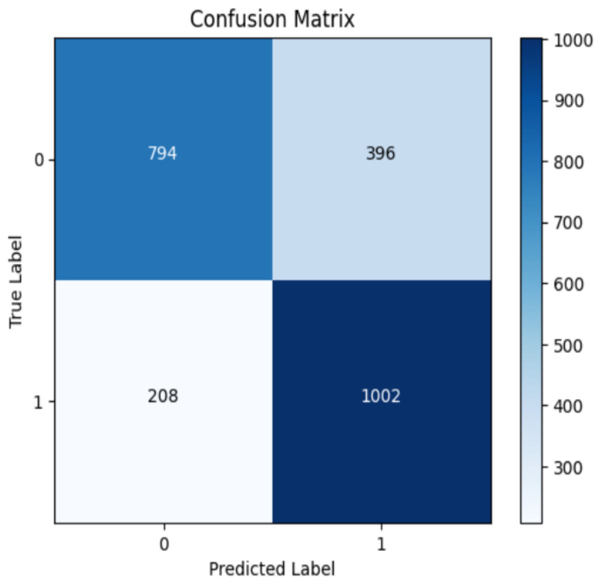
BERT Evaluation Graph





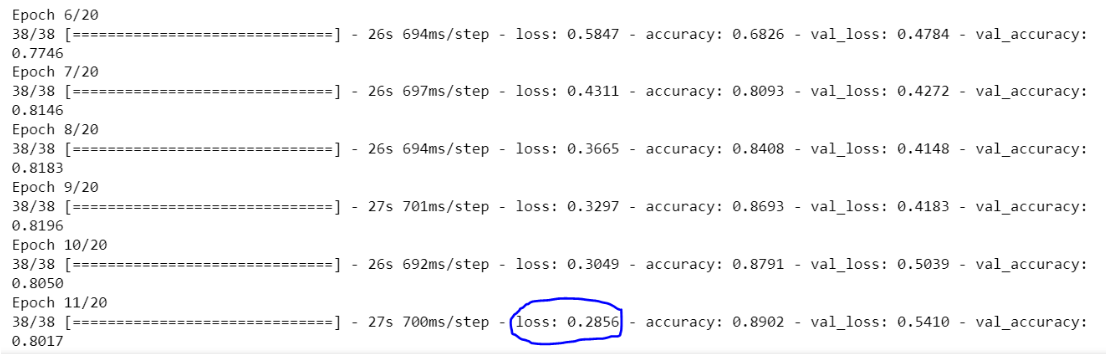
Confusion metrics of LSTM (left figure) and BERT(right figure)





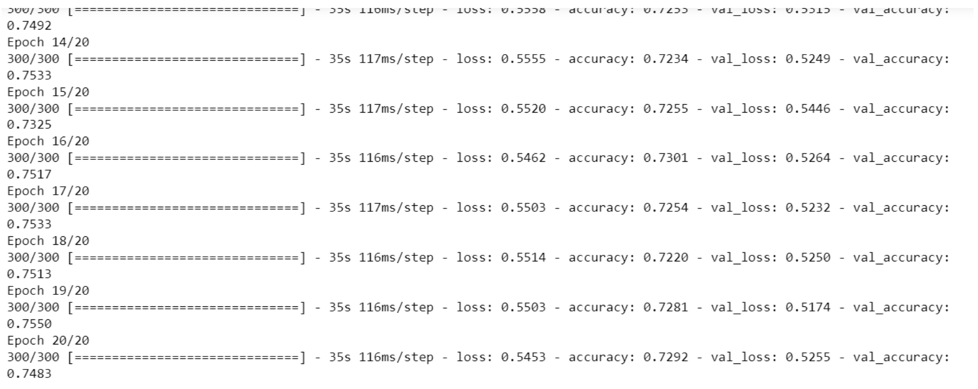
Some important aspects of the training process for the two models were captured from Jupyter notebook for further discussion in the result section as shown below.

TRAINING Process for LSTM



```
Epoch 6/20
38/38 [==============================] - 26s 694ms/step - loss: 0.5847 - accuracy: 0.6826 - val_loss: 0.4784 - val_accuracy:
0.7746
Epoch 7/20
38/38 [==============================] - 26s 697ms/step - loss: 0.4311 - accuracy: 0.8093 - val_loss: 0.4272 - val_accuracy:
0.8146
Epoch 8/20
38/38 [==============================] - 26s 694ms/step - loss: 0.3665 - accuracy: 0.8408 - val_loss: 0.4148 - val_accuracy:
0.8183
Epoch 9/20
38/38 [==============================] - 27s 701ms/step - loss: 0.3297 - accuracy: 0.8693 - val_loss: 0.4183 - val_accuracy:
0.8196
Epoch 10/20
38/38 [==============================] - 26s 692ms/step - loss: 0.3049 - accuracy: 0.8791 - val_loss: 0.5039 - val_accuracy:
0.8050
Epoch 11/20
38/38 [==============================] - 27s 700ms/step - loss: 0.2856 - accuracy: 0.8902 - val_loss: 0.5410 - val_accuracy:
0.8017
```

TRAINING Process for BERT



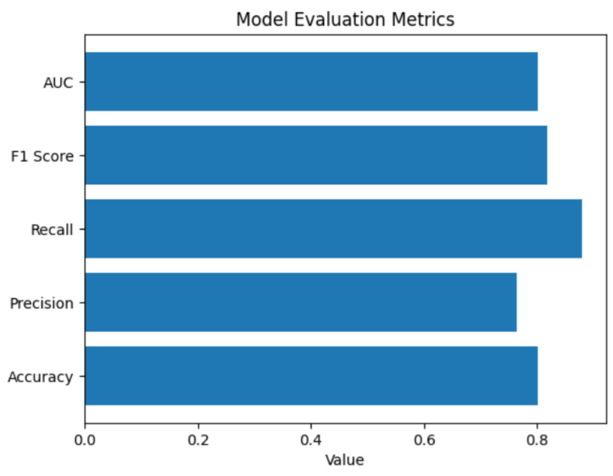
```
300/300 [==============================] - 35s 116ms/step - loss: 0.5558 - accuracy: 0.7255 - val_loss: 0.5315 - val_accuracy:
0.7492
Epoch 14/20
300/300 [==============================] - 35s 117ms/step - loss: 0.5555 - accuracy: 0.7234 - val_loss: 0.5249 - val_accuracy:
0.7533
Epoch 15/20
300/300 [==============================] - 35s 117ms/step - loss: 0.5520 - accuracy: 0.7255 - val_loss: 0.5446 - val_accuracy:
0.7325
Epoch 16/20
300/300 [==============================] - 35s 116ms/step - loss: 0.5462 - accuracy: 0.7301 - val_loss: 0.5264 - val_accuracy:
0.7517
Epoch 17/20
300/300 [==============================] - 35s 117ms/step - loss: 0.5503 - accuracy: 0.7254 - val_loss: 0.5232 - val_accuracy:
0.7533
Epoch 18/20
300/300 [==============================] - 35s 116ms/step - loss: 0.5514 - accuracy: 0.7220 - val_loss: 0.5250 - val_accuracy:
0.7513
Epoch 19/20
300/300 [==============================] - 35s 116ms/step - loss: 0.5503 - accuracy: 0.7281 - val_loss: 0.5174 - val_accuracy:
0.7550
Epoch 20/20
300/300 [==============================] - 35s 116ms/step - loss: 0.5453 - accuracy: 0.7292 - val_loss: 0.5255 - val_accuracy:
0.7483
```

RESULTS

The two models were evaluated based on accuracy, precision, recall, F1-score and AUC score. Their results are shown and discussed in this section.

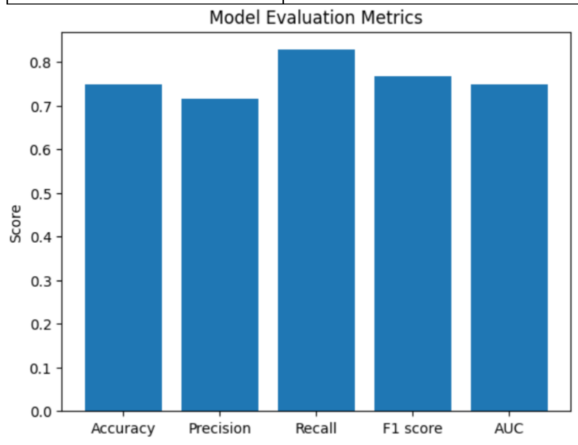
LSTM Results

Metrics	**Validation Score**
Accuracy	0.8016666666666666
Precision	0.7636494252873564
Recall	0.8785123966942149
F1-score	0.8170637970791699
AUC	0.8010209042294604



BERT Results

Metrics	Validation Score
Accuracy	0.7483333333333333
Precision	0.7167381974248928
Recall	0.828099173553719
F1-score	0.7684049079754601
AUC	0.7476630321550107



From the above results, it can be observed that the LSTM achieved an accuracy of 0.802, precision of 0.764, recall of 0.897, F1 score of 0.817 and AUC score of 0.801. The BERT model on the other hand achieved an accuracy of 0.748, precision of 0.717, recall of 0.828, F1 score of 0.768, and AUC score of 0.748.

The results from the confusion metrics shows that the LSTM model had correctly predicted 861 true negatives, 329 false positives reviews and incorrectly predicted 147 false negatives, and

1063 true positives reviews from the dataset. The BERT model on the other hand was able to correctly predict 794 true negatives, 396 false positives reviews, and incorrectly predicts 208 false negatives, and 1002 true positive reviews.

For the LSTM model, the training loss started at 0.6933 and decreased to 0.2856 by epoch 11. Similarly, the training accuracy started at 0.5065 and increased to 0.8902 by epoch 11. Also, the validation loss started at 0.6930 and decreased to 0.5410 by epoch 11, while the validation accuracy started at 0.5038 and increased to 0.8017 by epoch 11. The model seems to have made significant improvements in terms of accuracy and loss during the training process, which is a good sign.

For the BERT model, the training loss started at 0.6659 and decreased to 0.5453 by epoch 20. Similarly, the training accuracy started at 0.5971 and increased to 0.7292 by epoch 20. Also, the validation loss started at 0.6340 and decreased to 0.5255 by epoch 20, while the validation accuracy started at 0.6508 and increased to 0.7483 by epoch 20.

CONCLUSION

Based on the results, the LSTM outperformed BERT in terms of all evaluation metrics. It achieved higher accuracy, precision, recall, F1 score, and AUC than the BERT model. However, it is worth taking to account that the BERT model has potential for better performance if trained on a larger dataset.

There are more insights on the performance of the two models drawn from their confusion metrics. The BERT model shows a higher number of false negatives (208) than false positives (396), indicating that the model is not very good at correctly identifying positive reviews while the confusion matrix for the LSTM model shows a higher number of true positives (1063) than false positives (329), indicating that the model is better at correctly identifying positive reviews.

As with the LSTM model, the BERT model has made significant improvements in terms of accuracy and loss during the training process, which is a good sign. The final accuracy and loss scores are quite high and low, respectively, indicating that the model is performing well.

Sequel to the epoch information, it appears that the LSTM model performed well in the initial epochs with an accuracy of around 50%. However, its accuracy did not improve significantly over the course of training, hovering around 50% throughout the 20 epochs.

The BERT model started with a lower accuracy of around 60% but steadily improved with each epoch, reaching an accuracy of 71% by the 10th epoch.

Therefore, based on the epoch information, it can be concluded that the BERT model is better suited for this classification task than the LSTM model. Although, other factors, such as the size and quality of the dataset and the specific requirements of the task, should also be taken into consideration for the fallback experienced by the BERT model based on the performance metrics discussed in this report.

In conclusion, the LSTM model is the better choice overall for this specific Amazon Kindle book review dataset. Further investigation may be required to identify specific strengths and weaknesses of each model and how they can be utilized in different scenarios.

References

Mengting Wan, Julian McAuley. (2016) *International Conference on Data Mining (ICDM)*

https://www.kaggle.com/datasets/meetnagadia/amazon-kindle-book-review-for-sentiment-analysis Data of Access: 28/03/2023

Katrekar, A. and AVP, B.D.A. (2005) *An introduction to sentiment analysis*, GlobalLogic Inc https://www.globallogic.com/wp-content/uploads/2014/10/Introduction-to-Sentiment-Analysis.pdf. Date of Access: 04/04/2023

Aysun Ezen-Can, A. (2020) *A Comparison of LSTM and BERT for Small Corpus*, https://arxiv.org/abs/2009.05451. Date of Access: 12/04/2023

Balakrishnan V., Shi V., Law CL., Lim R., The LL., Fan Y., (2022) "*A Deep Learning Approach in Predicting Products' Sentiment Ratings: a Comparative Analysis*." The Journal of Supercomputing, vol. 78, no. 5, 2022, pp. 7206–7226., https://doi.org/10.1007/s11227-021-04169-6. Accessed 13/04/ 2023.

Moraes, Rodrigo, João Francisco Valiati, and Wilson P. Gavião Neto. "*Document-Level Sentiment Classification: An Empirical Comparison between SVM and ANN.*" Expert Systems with Applications 40, no. 2 (01.02.2013): 621-633. https://www-sciencedirect-com.hull.idm.oclc.org/science/article/pii/S0957417412009153. Accessed 18/04/ 2023.

Geetha, M.P.; Karthika Renuka,(2021) *Improving the performance of aspect based sentiment analysis using fine-tuned Bert Base Uncased model,* International Journal of Intelligent Networks, 2021, volume 2, 2121.
https://www.sciencedirect.com/science/article/pii/S2666603021000129?via%3Dihub.

Accessed 20/04/ 2023.