UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: *Nektarios Tsimpourakis Pavlakos*
sdi: sdi2200196

Course: *Artificial Intelligence II (M138, M226, M262, M325)*
Semester: *Spring Semester 2025*

Contents

1. Abstract

Continuing the exploration of sentiment analysis techniques, this third project evaluates the performance of fine-tuned pre-trained transformer models. Where previous projects employed TF-IDF with Logistic Regression and a custom PyTorch-based neural network with Word2Vec, this assignment shifts to fine-tuning BERT and DistilBERT from the HuggingFace ecosystem using the previously utilized English-language Twitter dataset. The core objectives include implementing the fine-tuning process for each model, evaluating their performance on unseen test data, and conducting a comparative analysis of their effectiveness in sentiment classification.

2. Data processing and analysis

2.1. Pre-processing

To prepare the textual data for fine-tuning the BERT (and subsequently DistilBERT) models, a series of preprocessing steps were implemented on the text content from the Twitter dataset. These transformations were designed to standardize the input, reduce noise from social media idiosyncrasies, and align the text with the expectations of the pre-trained tokenizers, ultimately aiming for improved model performance. The preprocessing steps performed were:

- Conversion to lowercase:
  All text was converted to lowercase. This is a standard procedure, particularly when utilizing the 'bert-base-uncased' model, as its tokenizer is trained on and expects lowercase input. This ensures consistency and prevents the model from treating words with different capitalization (e.g., "Happy" vs. "happy") as distinct entities.

- Anonymization through the removal of mentions and links:
  Mentions (e.g. @username) were replaced with a placeholder token ("X"), while URLs were replaced with a common placeholder ("http"). This preserves sentence structure while simultaneously anonymizing user identifiers, which is of major importance for any artificial intelligence model.

- Removal of non-ASCII characters:
  Eliminates emojis, mojibake (encoding errors), and non-English characters irrelevant to English sentiment analysis. By doing this, we ensure the model focuses on standard English characters that are well-represented in the 'bert-base-uncased' vocabulary

- Targeted correction of certain grammatical errors:
  A set of common spelling and slang errors was manually corrected after studying the available datasets. Examples include replacing "luv" with "love", "gr8" with "great", and "omg" with "oh my god". This normalization step aims to map colloquial variations to their standard English forms, potentially improving tokenization accuracy and the model's ability to understand the intended semantics.

It is noteworthy that several preprocessing steps employed in Project 2, such as extensive punctuation removal, hashtag replacement, and reduction of repeated characters,

were deliberately omitted or modified for this project. The rationale is that BERT's WordPiece tokenizer is inherently designed to handle many of these linguistic features. For instance:

- Punctuation is often treated as separate tokens or part of subword units by the tokenizer, potentially carrying semantic value.
- Hashtags are retained as the tokenizer can often break them into meaningful sub-tokens.
- The 'bert-base-uncased' tokenizer inherently handles casing by lowercasing input, and the model itself might learn to interpret repeated characters as emphasis

2.2. Analysis

While implementing the text preprocessing, we generated analytical visualizations to actively guide and validate the process. These visualizations yielded insights that empirically justified our preprocessing rules and offered crucial feedback for optimizing the modeling pipeline. A detailed breakdown of each visualization and its significance follows:

- Class Distribution based on sentiment for Train and Validation Sets:
  The distribution of sentiment labels was found to be perfectly balanced across both the training and validation datasets, with a 50/50 split (see Figure 1). This is a highly desirable characteristic in binary classification tasks, as it minimizes bias during training and ensures that the model is not skewed toward predicting the majority class. Consequently, the metrics calculated by the classification report will be more informative and reliable.

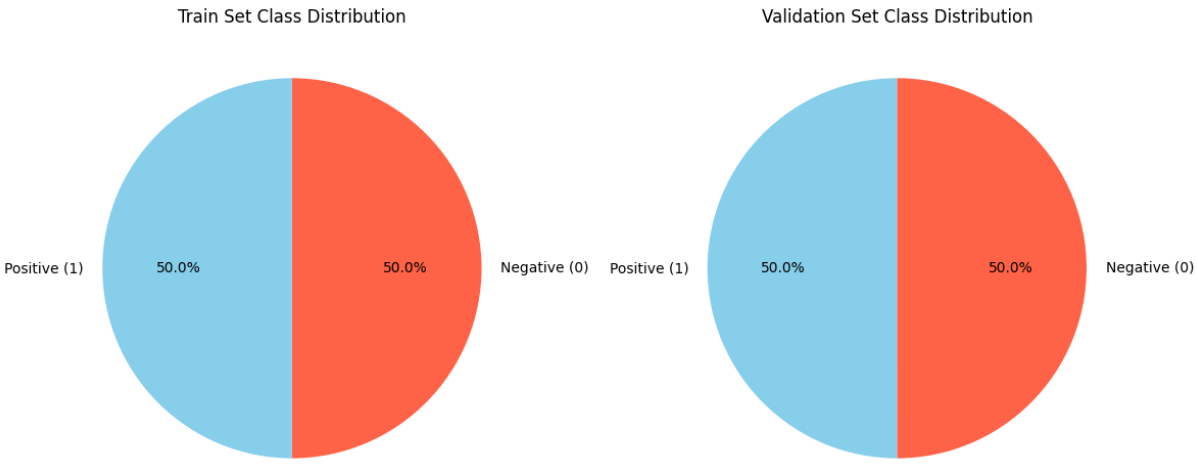

Figure 1: Class Distribution based on sentiment for Train and Validation Sets

- Average Word and Character Count per Tweet:
  This visualization (see Figure 2) compares the average number of words and characters per tweet before and after preprocessing. For the unprocessed tweets, the average word count is approximately 13.29 and the average character count is

approximately 74.77. After preprocessing, the average word count remains very similar at approximately 13.33, suggesting that our cleaning steps (like anonymization and spelling correction) do not significantly alter the number of word tokens. However, the average character count noticeably decreases to approximately 68.73. This reduction is primarily due to the removal of non-ASCII characters and the replacement of potentially long mentions and URLs with shorter, standardized placeholders ("X", "http"). This indicates that the preprocessing effectively reduces some noise and standardizes certain text elements without drastically changing the core word content, which is beneficial for subsequent tokenization by BERT and DistilBERT.

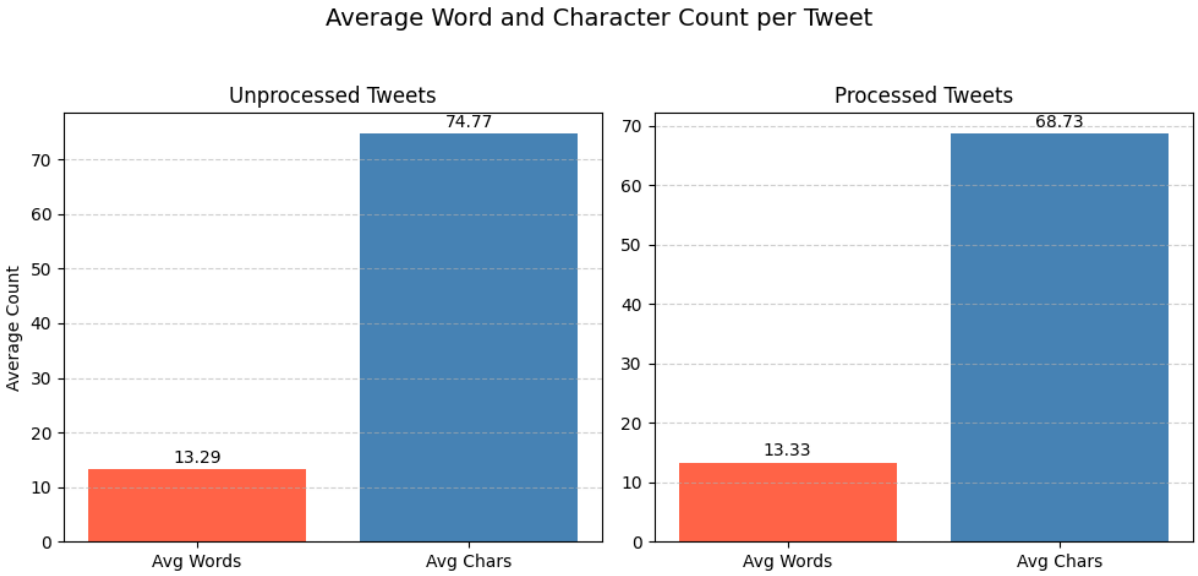

Figure 2: Average Word and Character Count per Tweet

- Set Distribution of Sequence Lengths (after BERT Tokenization):
  These histograms (see Figures 3 and 4) display the distribution of sequence lengths after tokenizing the processed text using the BERT tokenizer for both the training and validation sets. Both distributions are right-skewed, with the vast majority of sequences having between approximately 5 and 40 tokens. Crucially, almost all sequences are significantly shorter than the chosen BERT_MAX_LEN of 128, which is indicated by the red dashed line. This observation implies that very little to no information will be lost due to truncation, as nearly all tweets fit comfortably within this maximum length. While this means most sequences will be padded to reach 128 tokens, BERT's attention mechanism is designed to handle such padding effectively. The similarity in the shape of the distributions for the train and validation sets is also a positive sign, suggesting consistency in text characteristics across these datasets.

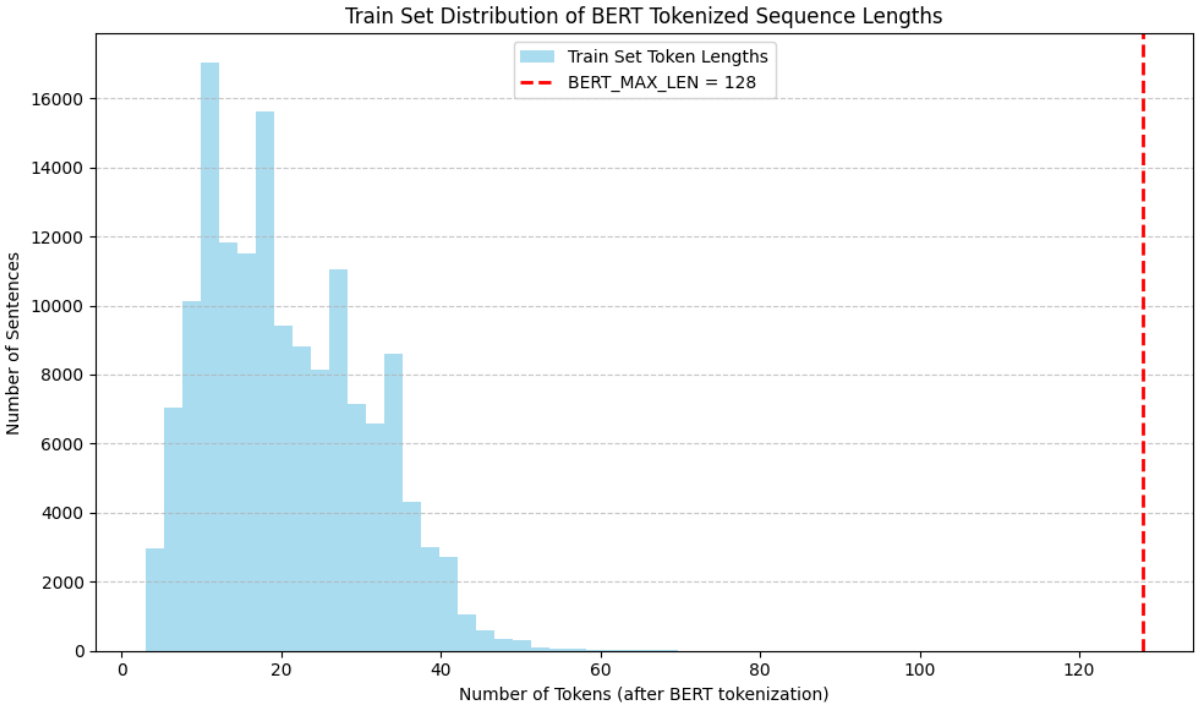

Figure 3: Train Set Distribution of Sequence Lengths

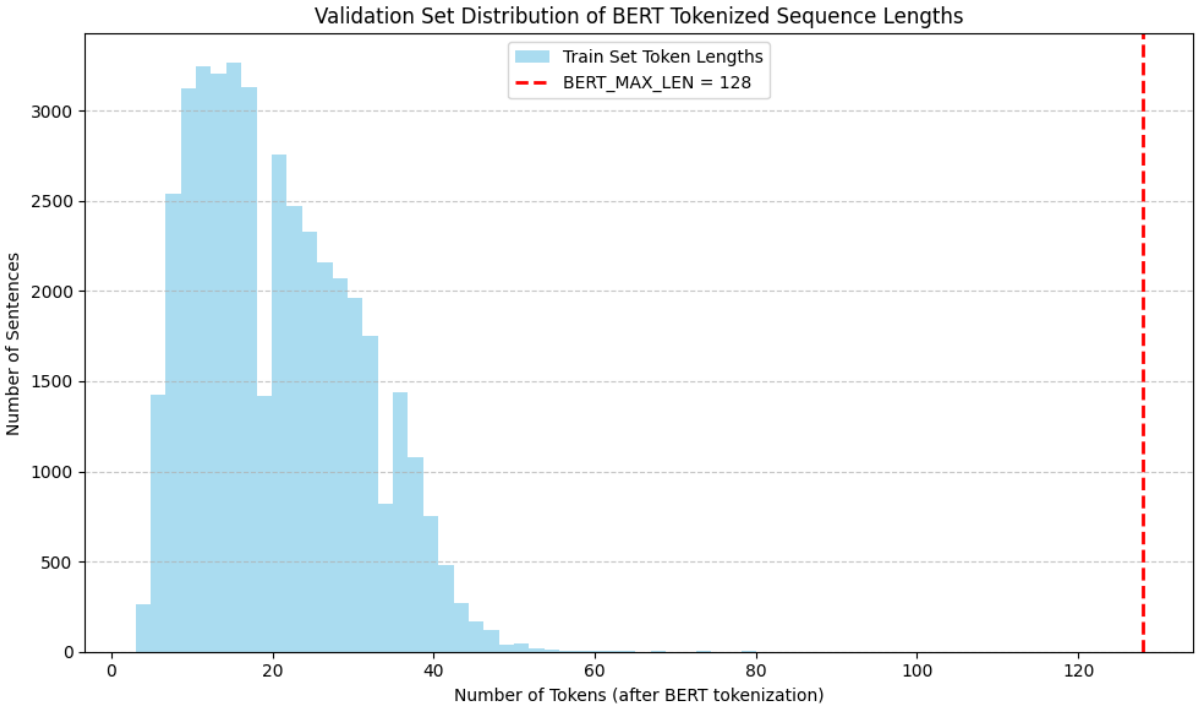

Figure 4: Validation Set Distribution of Sequence Lengths

2.3. Data partitioning for train, test and validation

The provided English-language Twitter dataset was partitioned into three distinct subsets: training, validation, and test. This division is crucial for developing a robust

sentiment classifier, allowing for model training, hyperparameter optimization, and an unbiased evaluation of its generalization capabilities on unseen data.

1. Training Set:
   The model was trained using the data in the train_df file, which contains both the text and the corresponding labels. This dataset is used to fit the parameters of the BERT and DistilBERT models, allowing it to learn the patterns and relationships between the text features and the sentiment expressed. For the final model training, the entirety of this dataset was utilized. However, during the iterative development and initial hyperparameter experimentation phases, a smaller subset, representing 30% of the full training data, was employed to facilitate faster experimentation and reduce computational overhead.

2. Validation Set:
   The val_df file, which also contains both text and labels, served as the validation set. The validation set helps in evaluating the model's performance during training while also serving as a guide for hyperparameter tuning. Overall, it acts as an intermediate stage between training and testing. Similar to the training set, while the full validation set was used for evaluating the final model, a smaller subset of 30% was used in conjunction with the training subset during preliminary model development cycles.

3. Test Set:
   The final evaluation of the fine-tuned BERT and DistilBERT models was performed on the test_df file. The models generate predictions for this set, which are then compared against the true, undisclosed labels to obtain the final performance metrics. By not exposing the test data during training or validation, we ensure a fair and unbiased estimate of the model's generalization ability.

2.4. Vectorization

Transforming the preprocessed textual data from tweets into a numerical format that can be effectively processed by transformer models like BERT and DistilBERT requires a sophisticated input preparation pipeline. This process, distinct from the static word embedding techniques used in the previous project, leverages the models' own tokenizers and adheres to their specific input structure. The following steps detail this vectorization and encoding procedure:

- Tokenization with WordPiece:
  Each preprocessed tweet was tokenized using BERT's WordPiece tokenizer. This tokenizer breaks down words into a vocabulary of common words and subword units. This subword tokenization is particularly effective for handling out-of-vocabulary (OOV) words, misspellings, and morphological variations prevalent in social media text, as it can represent rare words as sequences of more common subword pieces.

- Addition of Special Tokens:
  The tokenizer automatically prepends a [CLS] (classification) token to the beginning of each tweet's token sequence and appends a [SEP] (separator) token to the end. The [CLS] token's final hidden state is conventionally used as the aggregate sequence representation for classification tasks.

- Conversion to Input IDs:
  The resulting sequence of tokens was then mapped to their corresponding numerical indices in BERT's predefined vocabulary. These numerical sequences are referred to as input_ids and serve as the primary input to the model.

- Padding and Truncation:
  To ensure uniform input length for batch processing, all sequences were standardized to a fixed maximum length (BERT_MAX_LEN = 128 in this implementation). Sequences shorter than this maximum length were padded with a special padding token ID, while sequences longer than this were truncated.

The input_ids generated from this process provide the fixed-length numerical sequences fed directly to the BERT and DistilBERT models. These models then internally derive context-sensitive embeddings for each token during processing. This entire encoding pipeline was applied consistently across the training, validation, and test datasets.

3. Algorithms and Experiments

3.1. Experiments

This section details the entire investigation undertaken to develop and optimize the BERT and DistilBERT models for the text classification task, building upon the initial baseline model derived from the provided tutorial. My methodology involved a process where key components of the model and training pipeline were sequentially modified and evaluated. Each experiment aimed to isolate the impact of a specific change. This process allowed for beneficial decisions at each stage, guiding the model's evolution towards improved evaluation metrics before final hyperparameter tuning.

1. Baseline Model:
   The baseline implementation for both BERT and DistilBERT, closely followed the structure provided in the tutor's tutorial, establishing a foundation for further experimentation. This initial setup included a very basic preprocessing function that only removed mentions, links, and non-ASCII characters from the text. This decision was made because BERT and DistilBERT models are inherently capable of handling many other linguistic features and complexities within the text. Other aspects, such as the choice of optimizer and learning rate scheduler, also closely mirrored the tutorial's recommendations. The BERT baseline model completed training in 11 minutes and 30 seconds, achieving a validation accuracy of 0.8238, precision of 0.8355, recall of 0.8066, and an F1-score of 0.8208 on the validation set. In contrast, the DistilBERT baseline was considerably faster, training in just 5 minutes and 13 seconds, and yielded an accuracy of 0.8190, precision of 0.8295, recall of 0.8035, and an F1-score of 0.8163 on the validation set. This initial comparison highlighted DistilBERT's significant training time advantage with only a marginal difference in performance.

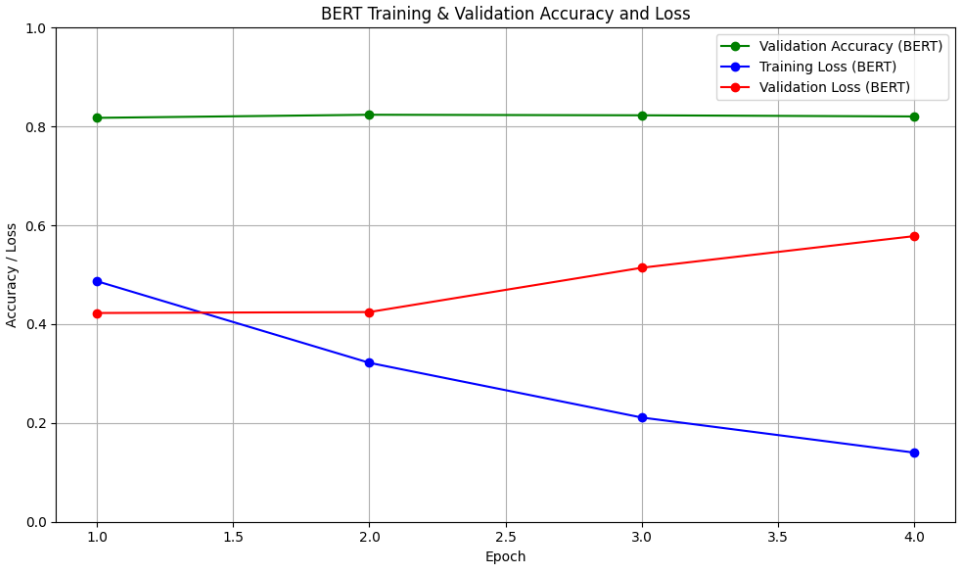

Figure 5: BERT - Baseline Model

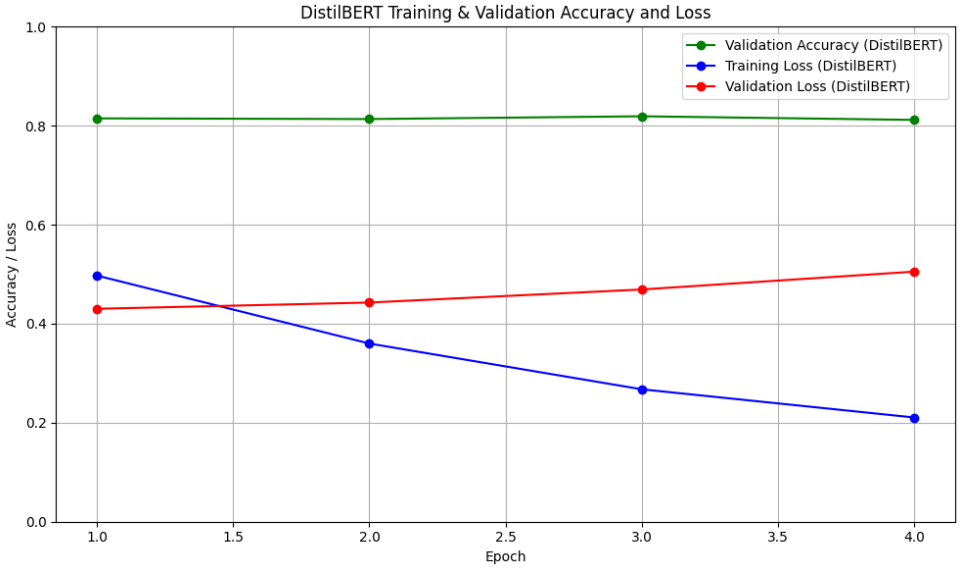

Figure 6: DistilBERT - Baseline Model

2. Optimizer and Weight Decay:
   The AdamW optimizer was selected for model training. AdamW enhances the standard Adam optimizer by decoupling weight decay from the gradient updates, which can lead to better model generalization and is a common choice for transformer models like BERT and DistilBERT due to its proven effectiveness over alternatives in such contexts. A weight decay rate of 0.1 was applied. This regularization technique helps prevent overfitting by adding a penalty to the loss function for large weights, encouraging the model to use simpler representations. The default epsilon (eps) value of 1e-8 was maintained throughout this experiment. This small constant, added to the denominator during adaptive learning rate calculations to ensure numerical stability by preventing division by zero, proved to be ideal and did not require further adjustment. Using this opti-

mizer setup, the BERT model achieved a validation accuracy of 0.8183, precision of 0.7974, recall of 0.8538, and an F1-score of 0.8246 on the validation set. The DistilBERT model, with the same optimizer configuration, yielded an accuracy of 0.8238, precision of 0.8291, recall of 0.8160, and an F1-score of 0.8225 on the validation set.

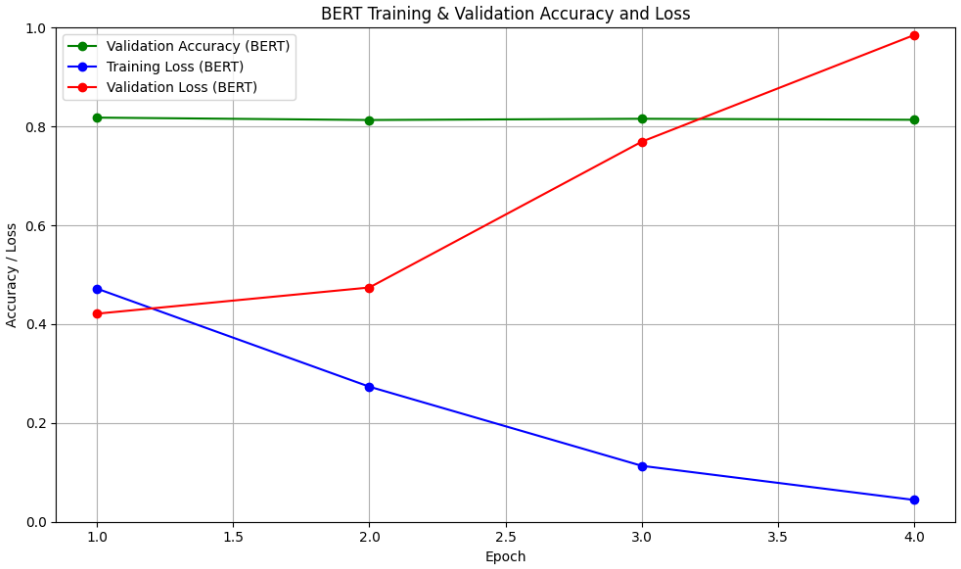

Figure 7: BERT - Optimizer and Weight Decay

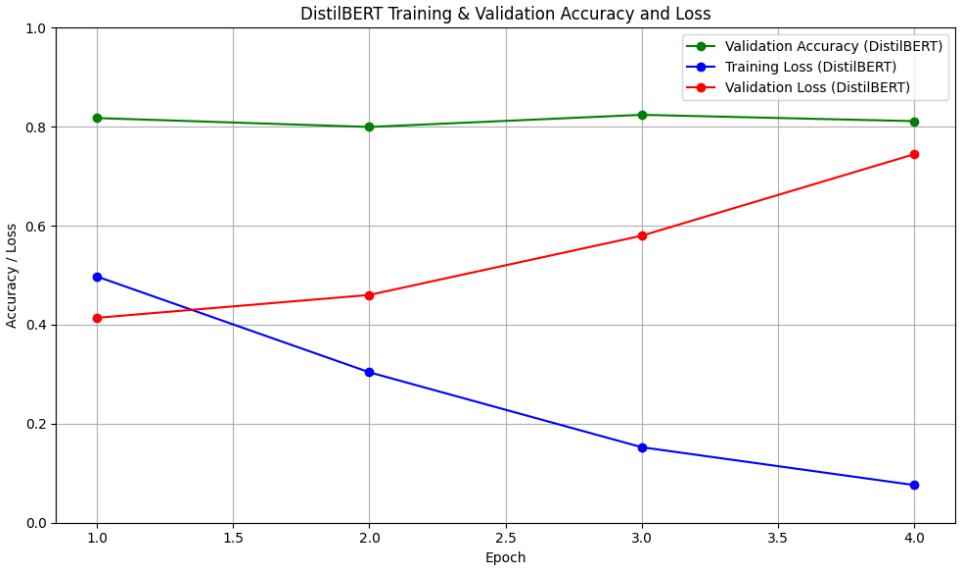

Figure 8: DistilBERT - Optimizer and Weight Decay

3. Early Stopping, Scheduler and Warmup Ratio:
   For managing the learning rate, the get_linear_schedule_with_warmup scheduler was employed. This scheduler begins by gradually increasing the learning rate from a very low value for an initial period. In this case, a warmup ratio of 5% of the total training steps was used which helps stabilize the model's learning in

the early stages. After the warmup phase, the learning rate is then decreased linearly towards zero. This approach is often ideal for fine-tuning large pre-trained models like BERT and DistilBERT as it prevents large disruptive updates at the beginning and allows for more precise adjustments later in training. Alongside this, early stopping was implemented, configured to forcefully terminate any training trial that did not show an improvement in validation metrics over the last two consecutive epochs. While early stopping's impact was not particularly significant in these experiments due to the short total training duration of only four epochs, it serves as a valuable mechanism to prevent wasted computation on trials that are unlikely to yield further improvements. With these settings, the BERT model achieved a validation accuracy of 0.8308, precision of 0.8379, recall of 0.8208, and an F1-score of 0.8292 on the validation set. The DistilBERT model, under the same conditions, attained an accuracy of 0.8183, precision of 0.8149, recall of 0.8239, and an F1-score of 0.8194 on the validation set.

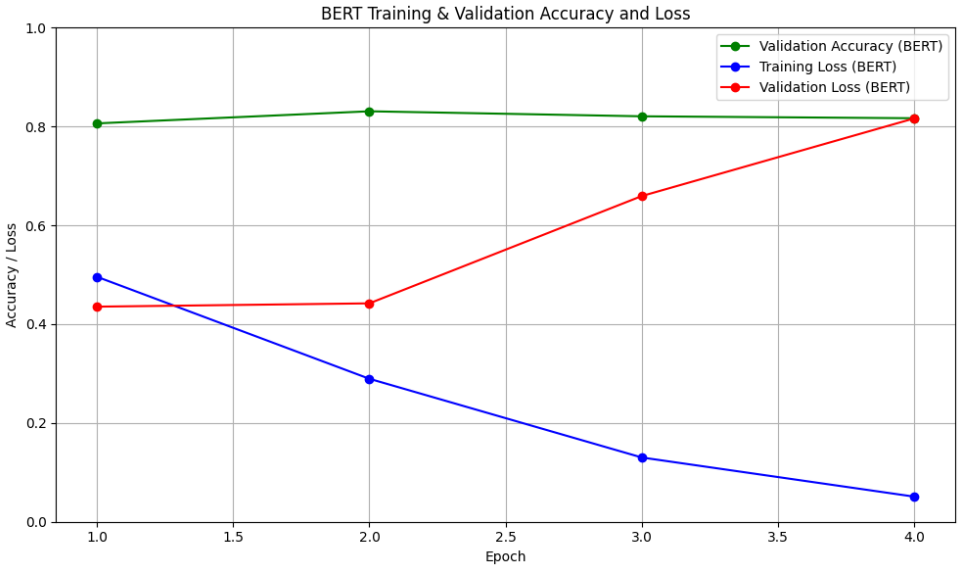

Figure 9: BERT - Early Stopping, Scheduler and Warmup Ratio

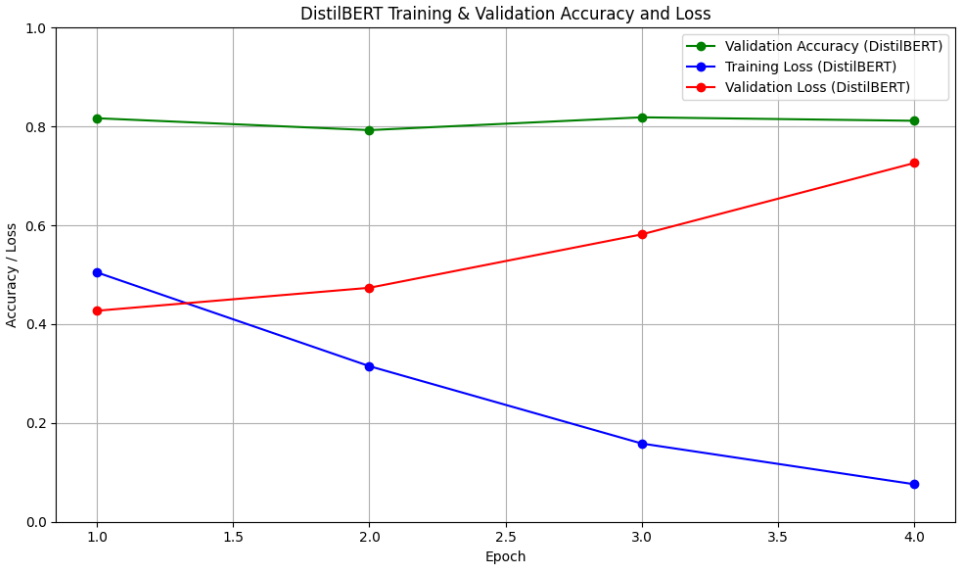

Figure 10: DistilBERT - Early Stopping, Scheduler and Warmup Ratio

4. Dropout:
   Dropout is a regularization technique used to prevent overfitting by randomly setting a fraction of neuron activations to zero during training, forcing the network to learn more robust features. For the BERT model, two specific dropout probabilities were tuned: hidden_dropout_prob and attention_probs_dropout_prob. The hidden_dropout_prob applies dropout to the outputs of the fully-connected layers within the Transformer's feed-forward networks and the pooled output, while attention_probs_dropout_prob applies dropout to the attention probabilities themselves. Initially, values of [0.05, 0.1, 0.15, 0.2] were manually tested for hidden_dropout_prob, with 0.15 yielding the best performance on the external test file as observed in a Kaggle competition submission. Subsequently, the same range of values was tested for attention_probs_dropout_prob, where the default value of 0.1 proved to be optimal. These optimized values were then also applied to the DistilBERT model, setting its general dropout to 0.15 and attention_dropout to 0.1. With these dropout configurations, the BERT model achieved a validation accuracy of 0.8183, precision of 0.8209, recall of 0.8145, and an F1-score of 0.8177 on the validation set. The DistilBERT model, using the corresponding dropout rates, obtained an accuracy of 0.8253, precision of 0.8265, recall of 0.8239, and an F1-score of 0.8252 on the validation set.

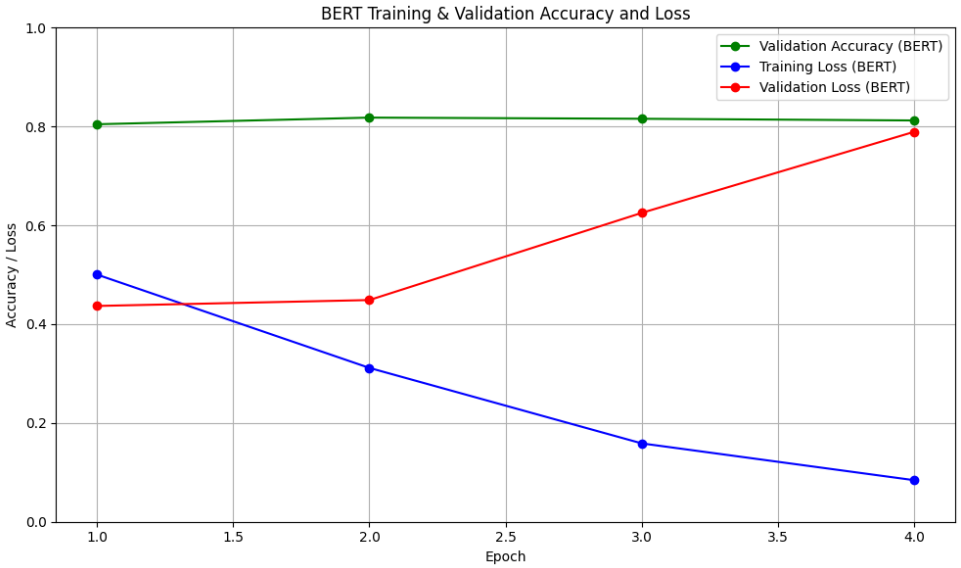

Figure 11: BERT - Dropout

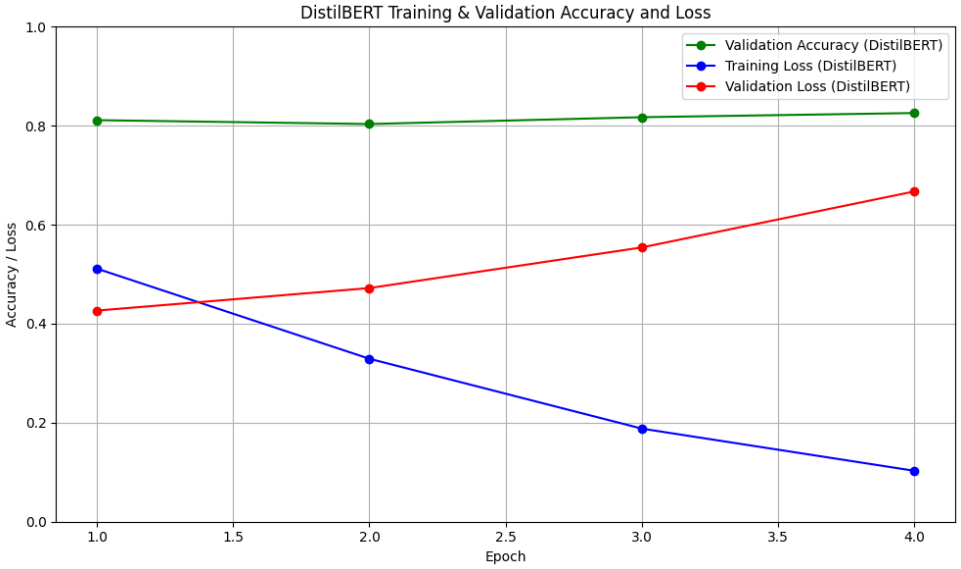

Figure 12: DistilBERT - Dropout

5. Preprocessing Adjustments:
   The final preprocessing function, which was applied to the data for these experiments, is described in detail in a preceding section of this report. It's worth reiterating that a key principle of this final preprocessing strategy was to intentionally leave much of the raw text data untouched, particularly concerning elements like punctuation and repeated characters. This approach was adopted because models like BERT and DistilBERT are specifically designed and pre-trained to understand and leverage such nuanced linguistic features, and aggressive cleaning might strip away valuable contextual information. After applying this refined preprocessing, the BERT model achieved a validation accuracy of 0.8206, precision of 0.8228, recall of 0.8176, and an F1-score of 0.8202. The DistilBERT model, with the same preprocessing, yielded an accuracy of 0.8112, precision of 0.7861,

recall of 0.8553, and an F1-score of 0.8193.

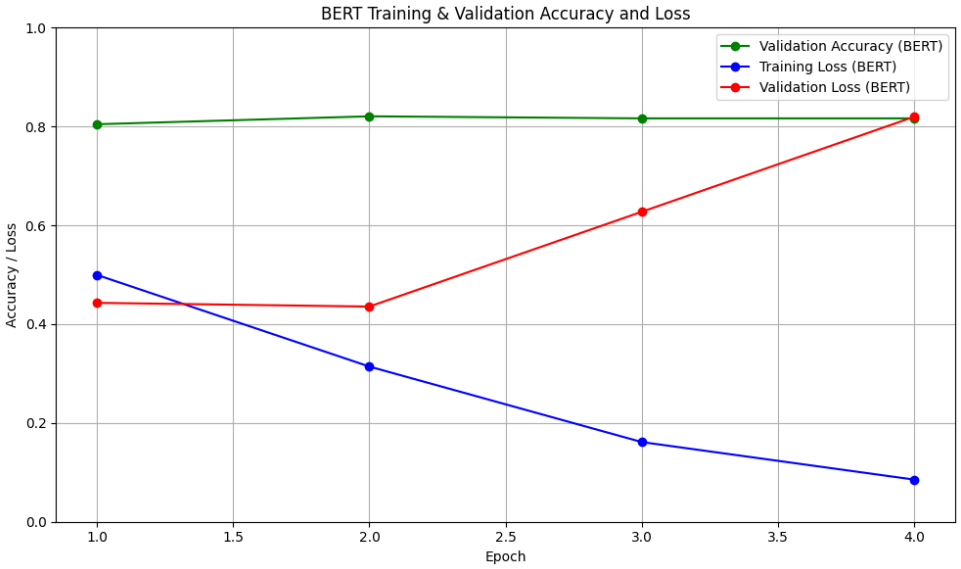

Figure 13: BERT - Preprocessing Adjustments

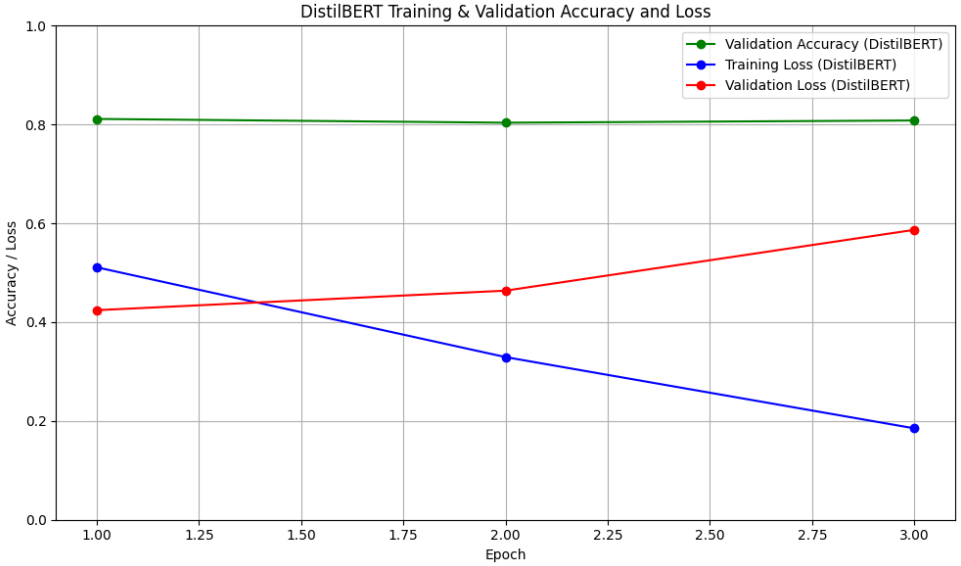

Figure 14: DistilBERT - Preprocessing Adjustments

6. Tuned Hyperparameters:
   Following the individual component optimizations, a final hyperparameter tuning step was performed using Optuna to identify optimal values for key training parameters. This process led to the selection of a learning_rate of approximately 2.96e-05 and a batch_size of 64. The learning rate dictates the step size at which the model adjusts its weights during training; this specific value is notably close to the 3e-5 learning rate often recommended by the original BERT authors for fine-tuning tasks. The batch size determines the number of training examples utilized in one iteration (or one forward/backward pass) of the model; a larger batch size can lead to more stable gradient estimates but requires more memory.

With these Optuna-derived hyperparameters, the BERT model achieved a validation accuracy of 0.8285, precision of 0.8245, recall of 0.8349, and an F1-score of 0.8297. The DistilBERT model, using the same tuned learning rate and batch size, obtained an accuracy of 0.8183, precision of 0.8358, recall of 0.7925, and an F1-score of 0.8136.

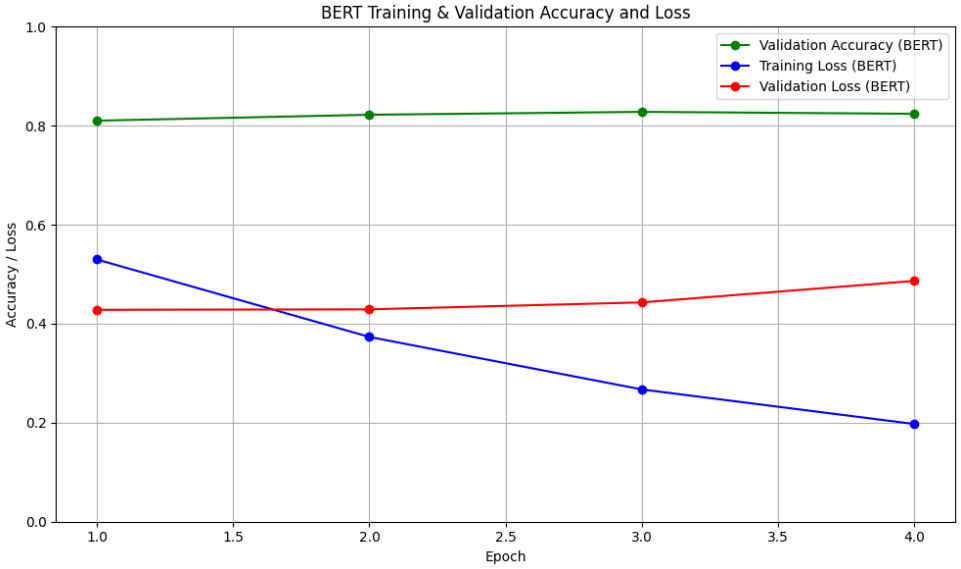

Figure 15: BERT - Tuned Hyperparameters

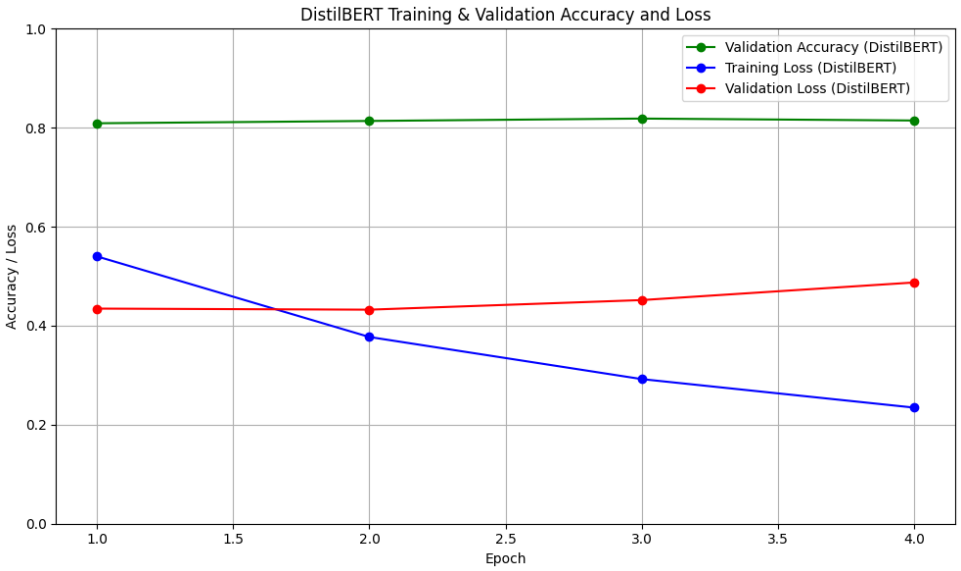

Figure 16: DistilBERT - Tuned Hyperparameters

7. Final Model:
   After iteratively optimizing individual components and hyperparameters, the final models for both BERT and DistilBERT were trained using the entire available training dataset, rather than a smaller subset (such as the 30% sometimes used for quicker initial experimentation or hyperparameter searches). This full dataset training aimed to leverage all available information to achieve the best

possible generalization and performance. The BERT model, trained on the complete dataset with the optimized configuration, achieved a validation accuracy of 0.8563, precision of 0.8612, recall of 0.8495, and an F1-score of 0.8553. Similarly, the final DistilBERT model yielded an accuracy of 0.8482, precision of 0.8623, recall of 0.8288, and an F1-score of 0.8452 on the validation set.

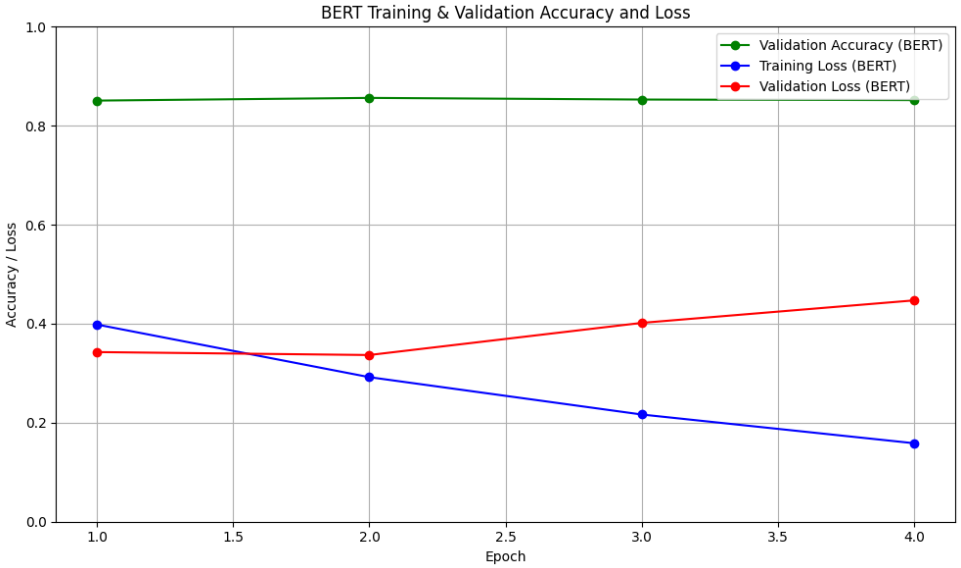

Figure 17: BERT - Final Model

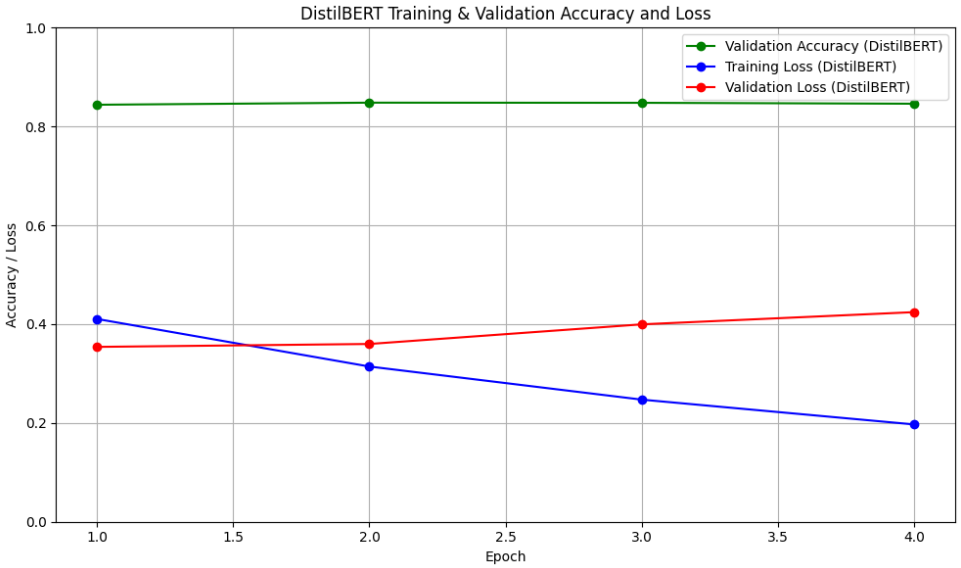

Figure 18: DistilBERT - Final Model

3.1.1. Table of trials. The tables of trials show a summary of the results from the experiments thoroughly analysed above. Each row highlights the main change we tested at that specific step. The accuracy scores listed are for the complete model setup at that point, including the new change.

Trial	Description - Analysed in above section	Validation Accuracy
1	Baseline Model	82.38
2	Optuna - AdamW, lr= 5e−5, weight_decay= 0.1	81.83
3	Early Stopping, Scheduler, warmup_ratio= 0.05	83.08
4	Optuna - batch_size= 32 (Changed later on)	82.61
5	Manual - hidden_dropout= 0.15, attention_dropout= 0.1	81.83
6	Preprocessing Adjustments	82.06
7	Optuna - lr= $2.9609614936542493e-05$, batch_size= 64	82.85
8	Final Model - Testing with entire datasets	85.63

Table 1: Trials for BERT Model

Trial	Description - Analysed in above section	Validation Accuracy
1	Baseline Model	81.90
2	AdamW, lr= 5e−5, weight_decay= 0.1	82.38
3	Early Stopping, Scheduler, warmup_ratio= 0.05	81.83
4	Manual - dropout= 0.15, attention_dropout= 0.1	82.53
5	Preprocessing Adjustments	81.12
6	Tuned Hyperparameters	81.83
7	Final Model - Testing with entire datasets	84.82

Table 2: Trials for DistilBERT Model

3.2. Hyper-parameter tuning

The model configuration and training process involved the following key components and hyper-parameters:

- BERT and DistilBERT base model (uncased):
  The foundation of the models was either the bert-base-uncased or distilbert-base-uncased pre-trained architectures. These were chosen for their strong performance on a wide range of natural language understanding tasks and because "uncased" models simplify the vocabulary by converting all input text to lowercase, which was found to be beneficial.

- 0.15 Dropout and 0.1 Attention Dropout:
  Specific dropout rates were applied to regularize the models and prevent overfitting. A hidden_dropout_prob of 0.15 was used for the outputs of the fully-connected layers, and an attention_probs_dropout_prob of 0.1 was applied to the attention scores. These values were selected after manual iterative testing, where 0.15 for hidden dropout showed the best results on an external test set, and 0.1 (the default) was optimal for attention dropout.

- 64 Batch Size:
  A batch size of 64 was used during training. This hyperparameter, which dictates the number of training samples processed before the model's weights are updated, was identified through Optuna hyperparameter optimization as providing a good balance between gradient stability and computational resource utilization.

- AdamW Optimizer:
  The AdamW optimizer was selected for training. This optimizer is an adaptation of Adam that improves weight decay regularization by decoupling it from the gradient updates, which is was effective for training these transformer models. A learning rate of 2.9609614936542493e-05 and a weight decay rate of 0.1 were used, in combination with the default epsilon value of 1e-8 was maintained for numerical stability, as it proved ideal.

- 5% Warmup Ratio:
  A learning rate warmup was configured for 5% of the total training steps. During this initial phase, the learning rate gradually increases from a very small value up to its target initial rate. This helps to stabilize training, particularly in the early stages of fine-tuning large pre-trained models.

- get_linear_schedule_with_warmup Scheduler:
  This learning rate scheduler was employed to manage the learning rate throughout training. After the initial warmup period, it linearly decreases the learning rate from its initial value down to zero over the remaining training steps. This is a common and effective strategy for fine-tuning transformers, allowing for more aggressive learning initially and finer adjustments later.

- 4 Epochs with Early Stopping:
  Training was conducted for a maximum of 4 epochs. An early stopping mechanism was implemented, set to terminate training if there was no improvement in validation metrics over the last two consecutive epochs. While the overall training duration was short, making early stopping less frequently triggered, it serves as a safeguard against overfitting and unnecessary computation in scenarios where the model might converge sooner.

Overfitting and Undefitting

Overfitting means creating a model that matches (memorizes) the training set so closely that the model fails to make correct predictions on new data. It's not uncommon to observe overfitting relatively quickly, even within just a few epochs, particularly when fine-tuning powerful, pre-trained models like BERT or DistilBERT. These models have a vast number of parameters and can rapidly adapt to the nuances of the specific training dataset, especially if the dataset size is modest relative to the model's capacity. However, the overfitting observed within these four epochs (see Figure 21 and Figure 22) isn't extreme suggesting the models are beginning to specialize rather than having lost their generalization ability.

Underfitting occurs when a model is too simple to capture the underlying patterns in the data, resulting in poor performance on both the training and validation sets. The training loss for both BERT (see Figure 21) and DistilBERT (see Figure 22) consistently decreases, demonstrating they are effectively learning the training data. Crucially, the validation accuracy also starts at a reasonably high level (around 0.84-0.85) and stays there, which isn't characteristic of underfitting where both training and validation performance would remain poor and show little improvement.

3.3. Optimization techniques

Several optimization techniques were employed to tune the model and improve training efficiency and overall performance:

- Manual Tuning and Component Selection:
  Initial decisions for core components were based on established best practices for transformer models and insights from the project's tutorial. This included selecting bert-base-uncased and distilbert-base-uncased, opting for the AdamW optimizer, and choosing the get_linear_schedule_with_warmup scheduler. Dropout rates and preprocessing strategies were also iteratively refined.

- Automated Hyperparameter Optimization (Optuna):
  The Optuna framework was utilized for a systematic and automated search for optimal hyperparameters, specifically for fine-tuning the learning rate and batch size, guiding the selection of values that enhanced model performance.

- Learning Rate Scheduling with Warmup:
  Implemented the get_linear_schedule_with_warmup scheduler, incorporating an initial warmup phase (5% of steps) followed by a linear decay of the learning rate. This technique optimizes training by stabilizing updates in the early epochs and allowing for finer adjustments as training progresses.

- Early Stopping:
  An early stopping mechanism monitored validation metrics and halted training if no improvement was observed for two consecutive epochs. This prevented overfitting and saved computational resources.

- GPU Acceleration:
  GPU acceleration via PyTorch's CUDA and Kaggle's GPU T4×2 capabilities was leveraged to significantly expedite training and experimentation cycles, allowing for more extensive iterative development.

3.4. Evaluation

To assess the performance of our fine-tuned BERT and DistilBERT models, several standard classification metrics were used: accuracy, precision, recall, and F1-score. These metrics provide comprehensive insights into each model's effectiveness in predicting sentiment labels on the validation set.

Evaluation Metrics

- Accuracy: Measures the proportion of correctly classified instances out of the total instances.
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
  The validation accuracy achieved with BERT was 0.8563, meaning the model correctly predicted the sentiment for approximately 85.63% of the samples. For DistilBERT, the accuracy was 0.8482 (84.82%).

- Precision: Measures the accuracy of positive predictions
$$Precision = \frac{TP}{TP + FP}$$

BERT achieved a precision of 0.8612, indicating that when this model predicted a positive sentiment, it was correct 86.12% of the time. DistilBERT's precision was 0.8623 (86.23%).

- Recall: Measures the ability to capture all relevant instances

$$Recall = \frac{TP}{TP + FN}$$

The recall for BERT was 0.8495, meaning the model correctly identified 84.95% of all actual positive sentiment samples. For DistilBERT, this value was 0.8288 (82.88%).

- F1 Score: Balances precision and recall into a single metric

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The F1-score for BERT was 0.8553 (85.53%), reflecting a robust and balanced performance between its precision and recall. DistilBERT achieved an F1-score of 0.8452 (84.52%).

Validation Set Performance

The detailed classification reports for both classes are as follows:

Class	Precision	Recall	F1-Score	Support
0 (Negative)	0.85	0.86	0.86	21197
1 (Positive)	0.86	0.85	0.86	21199
accuracy			0.86	42396
macro avg	0.86	0.86	0.86	42396
weighted avg	0.86	0.86	0.86	42396

Table 3: BERT - Classification Report

Class	Precision	Recall	F1-Score	Support
0 (Negative)	0.84	0.87	0.85	21197
1 (Positive)	0.86	0.83	0.85	21199
accuracy			0.85	42396
macro avg	0.85	0.85	0.85	42396
weighted avg	0.85	0.85	0.85	42396

Table 4: DistilBERT - Classification Report

3.4.1. ROC curve. The ROC curves for both BERT and DistilBERT exhibit strong discriminative ability, as evidenced by their steep rise from the origin and significant deviation above the diagonal no-skill line, indicating a high True Positive Rate is achieved while maintaining a low False Positive Rate across various thresholds. Notably, the performance is strikingly similar between the two, with both models achieving an excellent Area Under the Curve (AUC) of 0.93 and visually almost identical curves. This near-identical discriminative power suggests that DistilBERT offers a highly compelling balance of strong performance and computational efficiency for this task.

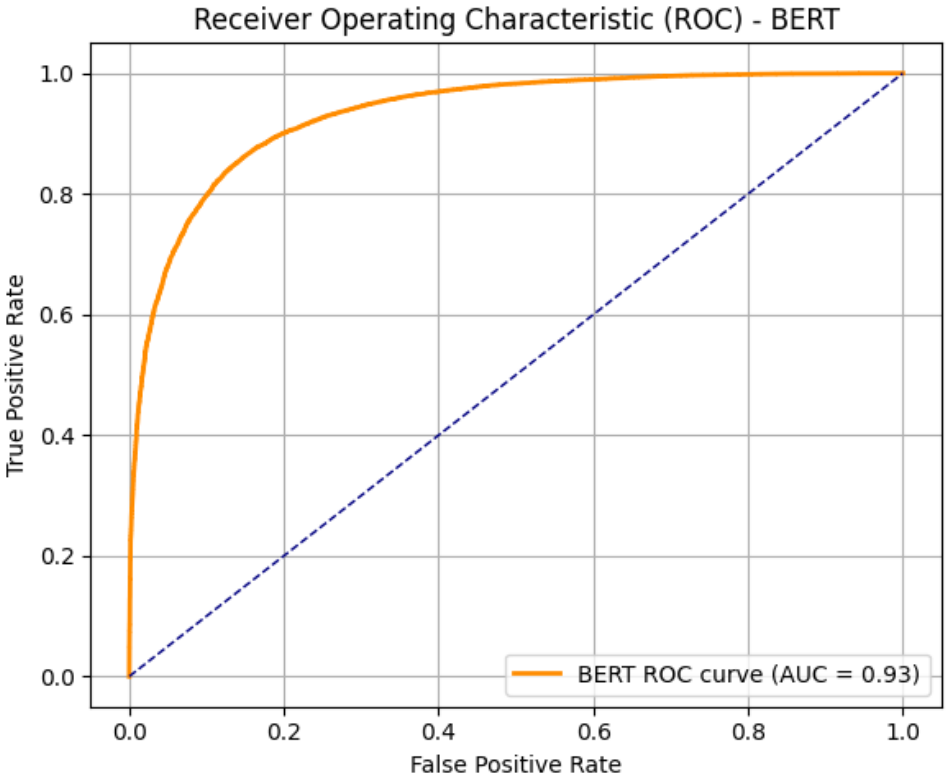

Figure 19: BERT - ROC curve

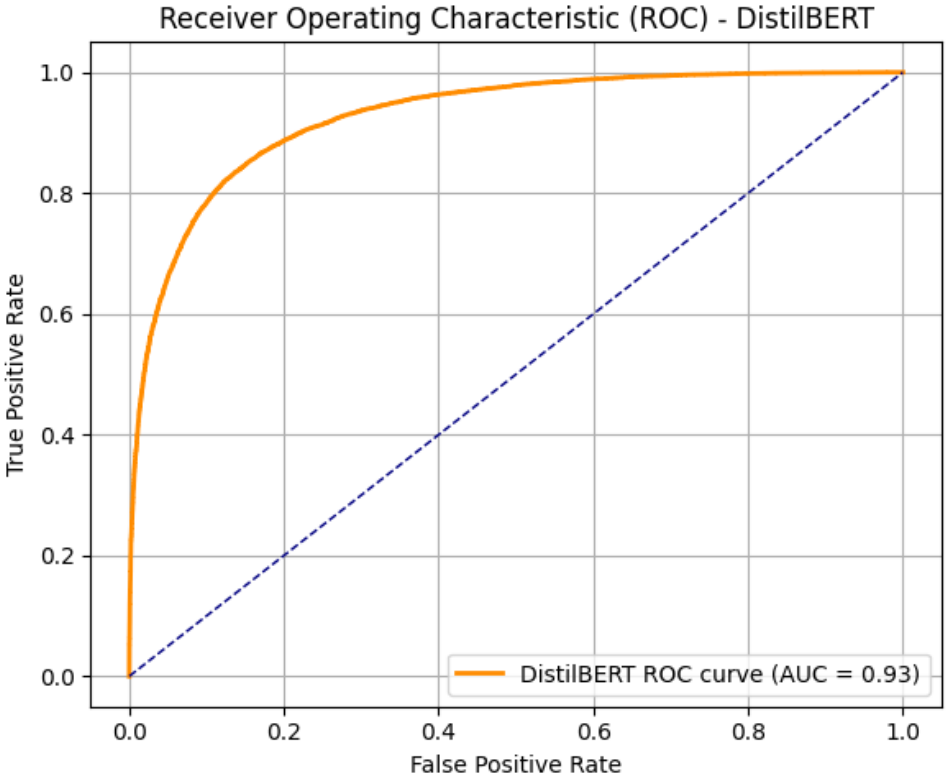

Figure 20: DistilBERT - ROC curve

3.4.2. Learning Curve. Both models exhibit very similar training dynamics. They quickly achieve their peak validation accuracy and then subsequently show increasing signs of overfitting as training continues into later epochs. BERT achieves a slightly lower final training loss, suggesting it might be fitting the training data slightly more closely, and its validation loss also ends slightly higher, which could indicate it's overfitting a bit more intensely by epoch 4 compared to DistilBERT.

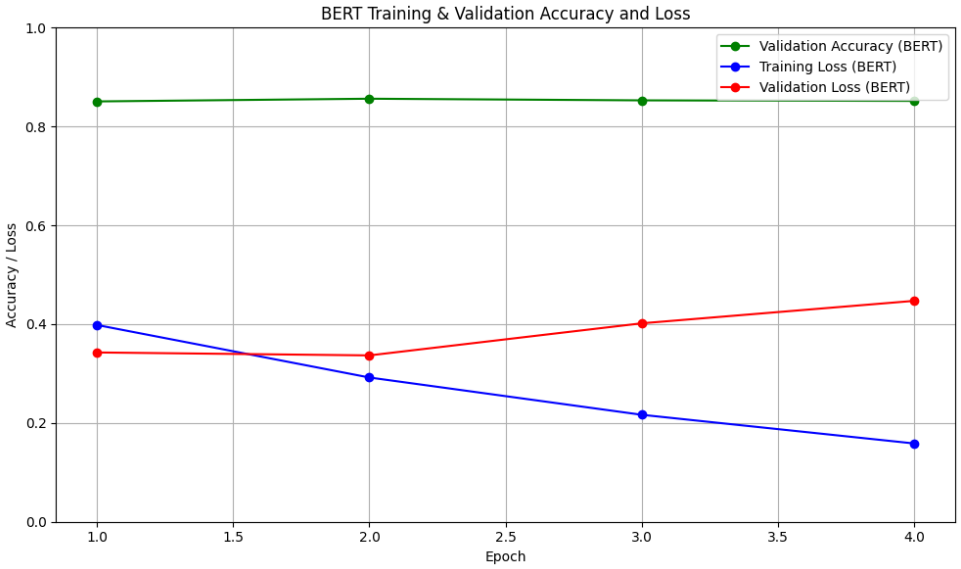

Figure 21: BERT - Learning Curve

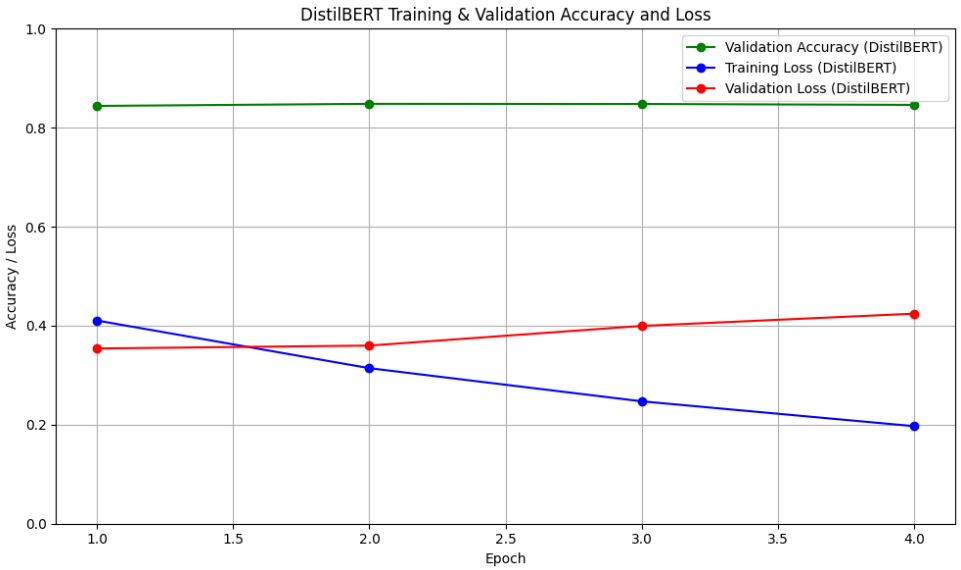

Figure 22: DistilBERT - Learning Curve

3.4.3. Confusion matrix. Both confusion matrices reveal strong classification performance for BERT and DistilBERT, with high counts along the main diagonal indicating a large number of correct predictions for both positive and negative sentiments. A closer look shows BERT correctly identifies more true positive instances (18008 vs.

DistilBERT's 17570) and consequently has fewer false negatives (3191 vs. 3629), suggesting a slightly better ability to capture actual positive cases. DistilBERT, on the other hand, registers marginally more true negatives (18392 vs. BERT's 18294) and slightly fewer false positives (2805 vs. BERT's 2903). Overall, both models demonstrate a high degree of accuracy in distinguishing between the classes, with BERT showing a slight advantage in correctly classifying positive instances and DistilBERT slightly better at identifying true negatives.

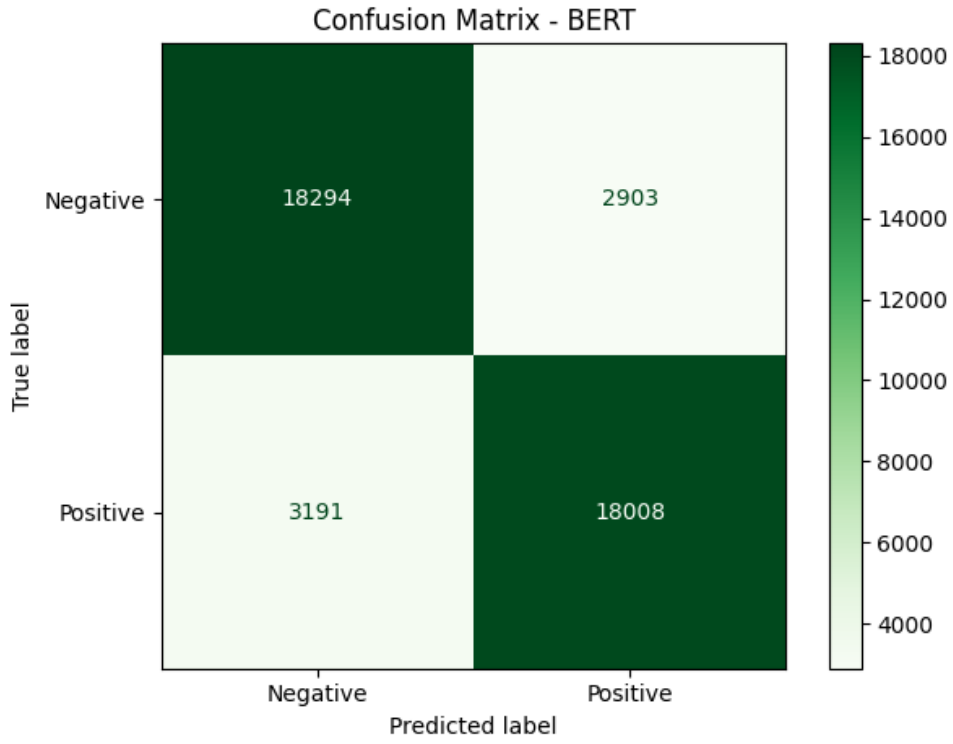

Figure 23: BERT - Confusion matrix

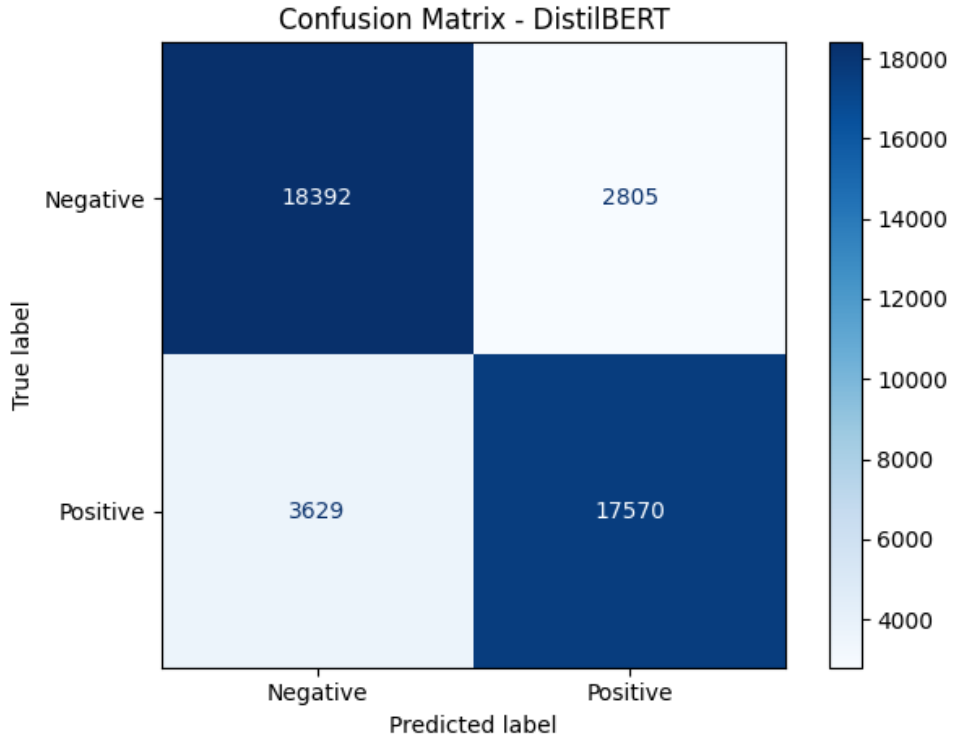

Figure 24: DistilBERT - Confusion matrix

4. Results and Overall Analysis

4.1. Results Analysis

The primary objective of this project was to develop and optimize BERT and DistilBERT models for the text classification task, aiming to achieve high accuracy on unseen data. Through iterative experimentation with model components, preprocessing, and hyperparameter tuning, the final models achieved the following key performance metrics:

	BERT Model:	**DistilBERT Model:**
Validation Set Accuracy	85.63%	84.82%
Test Set Accuracy (Kaggle)	85.65%	84.75%

Table 5: BERT and DistilBERT - Accuracy Metrics

Analysis:

- These results indicate that both models learned effectively from the training data and generalized exceptionally well to unseen validation and test sets. The validation and test accuracies were very similar (for example, BERT got 85.63% on validation and 85.65% on the test set). This close alignment suggests that the validation set was a reliable guide for final test performance and that the models were not significantly overfitting.

- Even though early training showed initial tendencies towards overfitting, the final selected model configurations successfully mitigated severe overfitting, yielding strong generalization. This was analysed further in a previous section which also included the related graphs.
- Achieving test accuracies of 85.65% for BERT and 84.75% for DistilBERT demonstrates highly capable classifiers, significantly outperforming many simpler approaches similar to the ones implemented in previous assignments. The slightly superior performance of BERT is expected given its larger architecture, but DistilBERT's performance, being only marginally lower, highlights its efficiency and effectiveness as a smaller, faster alternative.

There are several additional experiments and improvements that could be explored to further enhance the model's performance such as:

- Automated Hyperparameter Optimization (Optuna) for DistilBERT:
  While BERT underwent Optuna tuning for learning rate and batch size, a similarly extensive automated hyperparameter search for DistilBERT was not performed due to constraints with Kaggle's limited free accelerator time.
- Extended Training on Full Datasets:
  Although the final models were trained on the entire available training dataset for 4 epochs, exploring more extended training (e.g., 5-10 epochs) with robust early stopping or further experiments with the entire datasets could be beneficial. This was computationally prohibitive under the current project constraints but remains a valid thought for improvement.

4.1.1. Best trial.

- BERT - Performance Metrics
  - Test Accuracy (Kaggle): 85.65%
  - Validation Accuracy: 85.63%
  - Validation Precision: 86.12%
  - Validation Recall: 84.95%
  - Validation F1-Score: 85.53%
- DistilBERT - Performance Metrics
  - Test Accuracy (Kaggle): 84.75%
  - Validation Accuracy: 84.82%
  - Validation Precision:86.23%
  - Validation Recall: 82.88%
  - Validation F1-Score: 84.52%
- Models
  - Sentiment classifiers using pre-trained transformer architectures, specifically bert-base-uncased and distilbert-base-uncased, for the English-language Twitter dataset using PyTorch.

- Best Hyperparameters
  - BERT and DistilBERT base model (uncased)
  - 0.15 Dropout and 0.1 Attention Dropout
  - 64 Batch Size
  - AdamW Optimizer
  - 5% Warmup Ratio
  - get_linear_schedule_with_warmup Scheduler
  - Epochs with Early Stopping

4.2. Comparison with the first project

Comparing the current transformer-based models with the first project, which employed a Logistic Regression classifier with TF-IDF features, reveals a substantial advancement in performance and efficiency. The first project achieved a validation accuracy of 80.49% and an F1-score of 80.59% on the full dataset and traditional machine learning techniques. In stark contrast, both BERT (85.63% validation accuracy, 85.53% F1-score) and DistilBERT (84.82% validation accuracy, 84.52% F1-score) from the current project demonstrated a significant leap when trained on the full dataset. Notably, even when fine-tuned on just 5% of the training dataset, these transformer models were able to achieve performance levels comparable to the Logistic Regression model, highlighting their remarkable data efficiency. This improvement can be attributed to the sophisticated contextual understanding and rich feature representations inherent in pre-trained transformer architectures, which far surpass the capabilities of models like TF-IDF.

4.3. Comparison with the second project

Comparing the current transformer-based models with the second project, which involved a custom Deep Neural Network (DNN) that utilized pre-trained static word embeddings (specifically GloVe-twitter-200), illustrates another significant advancement. The second project achieved a test accuracy of 78.98% and a F1-score of 78.94% on the full dataset. The current BERT and DistilBERT models represent another significant step forward from this approach. When trained on the complete dataset, BERT achieved a test accuracy of 85.65% and DistilBERT reached 84.75%, marking an improvement of approximately 6-7 percentage points over the DNN with static embeddings. The performance of the second project was somewhat lower than initially anticipated for a deep learning approach with static embeddings. Consequently, the current transformer-based models prove to be a much better fit for tackling this sentiment analysis task. This overall better performance can be attributed to their advanced architecture, which allows them to capture complex linguistic patterns and contextual nuances far more effectively.

5. Bibliography

References

[1] Hugging Face. Bert base model (uncased).

[2] Hugging Face. Distilbert base model (uncased).

[3] Dan Jurafsky and James H. Martin. *Speech and Language Processing*.

[4] Manolis Koubarakis. Bert, 2025.

[5] Manolis Koubarakis. Machine translation, 2025.

[6] Manolis Koubarakis. Transformers, 2025.

[7] 3Blue1Brown or Grant Sanderson. Neural networks course, 2017-2025.

[8] Despina-Athanasia Pantazi. Homework tutorials, 2025.

[5] [6] [4] [8] [7] [1] [2] [3]